

**2D Object Recognition by  
Adaptive Feature Extraction and  
Dynamical Link Graph Matching**

**Kenneth Flaton**

**CENG Technical Report 92-24**

**Department of Electrical Engineering - Systems  
University of Southern California  
Los Angeles, California 90089-2562  
(213)740-4484**

2D OBJECT RECOGNITION BY  
ADAPTIVE FEATURE EXTRACTION AND  
DYNAMICAL LINK GRAPH MATCHING

by

Kenneth Flaton

---

A Dissertation Presented to the  
FACULTY OF THE GRADUATE SCHOOL  
UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the  
Requirements for the Degree  
DOCTOR OF PHILOSOPHY  
(Electrical Engineering)

May 1992

Copyright 1992 Kenneth Flaton

## Acknowledgements\*

I would begin by thanking my friends and colleagues, with whom, over the past four years, I have probably had as much conversation, collectively, as with my own wife. Those to whom I owe the greatest thanks are: Steve Alexander, Bill Betts, Joachim Buhmann, Jörg Lange, Martin Lades, Walter Tackett, and Rolf Würtz.

I wish also to thank Professor Jean-Luc Gaudiot for his direction and support during my entire doctoral candidacy. His energy and our numerous conversations were key to the development and execution of my research.

Professor Gerard Medioni was exceedingly helpful in getting me to see my work in terms of the general vision problem. His insight and knowledge of the computer vision field were central to my development of the proof-of-concept experiments described in the dissertation.

Through Professor Christoph von der Malsburg, I have had the pleasure of working for that rare combination of mentor and friend. His effect on my scientific philosophy and outlook are evident in the dissertation. Academically and personally, I am a richer and better person for having known him.

The raising of children is often a thankless task, a condition that I would at least partially rectify here. It was my parents, Peter and Sylvia, who instilled in me the desire, discipline, and ethic that enabled me to perform this work. Without their continuous encouragement and support, the path to this point would have been considerably longer.

By far, however, my greatest debt of gratitude must go to Noelle who has daily supported me emotionally and logistically. Without her steadfast love and care, this degree would likely have been unattainable.

---

\* This work was partially funded by a Hughes Doctoral Fellowship.

# Contents

<b>Acknowledgements</b>	<b>iii</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xi</b>
<b>Abstract</b>	<b>xiii</b>
<b>Chapter 1 - Introduction</b>	<b>1</b>
1.1 Problem Statement .....	1
1.2 Background .....	1
1.3 Approach .....	3
1.4 Related Work .....	4
1.5 Scope of Work .....	6
1.6 Summary of Results .....	7
1.7 Organization of the Dissertation .....	7
<b>Chapter 2 - Adaptive Feature Extraction</b>	<b>9</b>
2.1 Morlet Wavelet Decomposition .....	9
2.1.1 Gabor Functions .....	10
2.1.2 Wavelet Decomposition .....	11
2.1.3 Laplacian Preprocessing .....	14
2.1.4 Real vs. Complex Decompositions .....	15
2.1.5 Orientation Column Tuning .....	18
2.1.6 Saliency and Jet Magnitudes .....	19
2.1.7 Algorithm Implementation .....	20
2.2 Adaptive Vector Quantization .....	20
2.2.1 Learning .....	22
2.2.2 Quantization .....	23
2.2.3 Algorithm Implementation .....	23

<b>Chapter 3 - Graph Formation</b>	<b>25</b>
3.1 Node Generation .....	25
3.2 Edge Generation.....	27
3.3 Implementation .....	28
<b>Chapter 4 - Dynamical Link Graph Matcher</b>	<b>29</b>
4.1 Overview .....	29
4.2 Architecture.....	33
4.2.1 Neural Planes .....	33
4.2.2 Links.....	34
4.3 Dynamics .....	41
4.3.1 Neural Dynamics.....	41
4.3.2 Link Dynamics .....	42
4.3.3 System Dynamics.....	46
4.4 Other Considerations.....	49
4.4.1 Mixing Factors .....	49
4.4.2 Uniqueness .....	50
4.5 Implementation .....	51
<b>Chapter 5 - Results</b>	<b>53</b>
5.1 Processing Module Results .....	53
5.1.1 Morlet Wavelet Decomposition .....	53
5.1.2 Adaptive Vector Quantization .....	60
5.1.3 Graph Formation .....	64
5.1.4 Dynamical Link Graph Matching .....	66
5.2 Recognition in Toyland.....	72
5.3 Summary of Results .....	95
<b>Chapter 6 - Conclusion</b>	<b>97</b>
6.1 Summary .....	97
6.2 Observations, Limitations, and Extensions .....	99
6.2.1 Multiview Recognition .....	99
6.2.2 Negative Recognition.....	100
6.2.3 Expectation Driven Perception .....	100
6.2.4 Implementation Issues.....	101
6.2.5 Scale and Rotation Robustness .....	101

6.2.6	Computational Complexity .....	102
6.2.7	Gallery Scaling Issues .....	104
6.2.8	Dealing with Complex Presentations .....	105
6.2.9	Fusion .....	106
<b>Appendix A - Data Structures for the Feature Extractor</b>		<b>A-1</b>
<b>Appendix B - System Flowchart</b>		<b>B-1</b>
<b>Appendix C - Program Flowcharts for GTVQ</b>		<b>C-1</b>
<b>Appendix D - Program Flowcharts for AVQ</b>		<b>D-1</b>
<b>Appendix E - Program Flowcharts for SGMaker</b>		<b>E-1</b>
<b>Appendix F - Program Flowcharts for DyLink GM</b>		<b>F-1</b>
<b>Appendix G - Dialog Boxes for GTVQ</b>		<b>G-1</b>
<b>Appendix H - Dialog Boxes for AVQ</b>		<b>H-1</b>
<b>Appendix I - Dialog Box for SGMaker</b>		<b>I-1</b>
<b>Appendix J - Dialog Boxes for DyLink GM</b>		<b>J-1</b>
<b>References</b>		<b>R-1</b>

## List of Figures

Figure 2.1 - 2D complex Gabor function .....	10
Figure 2.2 - Image representation using a jet .....	12
Figure 2.3 - Alternate Gabor-filtered image representations .....	13
Figure 2.4 - Misalignment of Morlet wavelet components in a Morlet pyramid.....	13
Figure 2.5 - Laplacian mask shape.....	14
Figure 2.6 - Wavelet gain resulting from Laplacian edge enhancement .....	15
Figure 2.7 - Feature ambiguity problem using real wavelets.....	16
Figure 2.8 - Nature of feature ambiguity problem with real Morlet wavelets .....	17
Figure 2.9 - Complex kernel solution to the feature ambiguity problem .....	18
Figure 2.10 - Orientation column tuning by convolution with $\{-1,2,-1\}$ .....	19
Figure 2.11 - Feature content of a jet .....	21
Figure 2.12 - Elemental features extractable by vector quantization of Morlet jets ...	21
Figure 4.1 - Schematic representation of the DyLink graph matcher .....	30
Figure 4.2 - Main system dynamics .....	31
Figure 4.3 - Overview of the matching process .....	32
Figure 4.4 - Feature label confusion .....	37
Figure 4.5 - Repetitive Labeling Problem scenario .....	38
Figure 4.6 - Repetitive Labeling Problem.....	39
Figure 4.7 - Spotlight Halo Problem .....	44
Figure 4.8 - Graphic depiction of Equation 4.25, Inverse Parabolic Function .....	45
Figure 4.9 - Graphic depiction of Equation 4.26, Asymmetric Anti-Halo Function ..	45
Figure 5.1 - Standard MWD of Stickville polygon.....	55
Figure 5.2 - Jet magnitudes (saliency) for Stickville cross and polygon .....	56
Figure 5.3 - Jet magnitudes (saliency) for infrared truck and greyscale face .....	57
Figure 5.4 - Response of varying $s$ on jet magnitudes .....	58
Figure 5.5 - Results of FFT convolution with and without zero padding .....	59
Figure 5.6 - Jet images for five features.....	59
Figure 5.7 - Jet continuity images.....	60
Figure 5.8 - Self-organization of a vector map .....	61
Figure 5.9 - Quantization error versus iteration for vector map .....	61
Figure 5.10 - Quantization results using trained vector map .....	62
Figure 5.11 - Feature images using model vectors from Stickville .....	63
Figure 5.12 - The node selection process.....	64

Figure 5.13 - Nodes of a labeled, locally connected graph are formed by sampling..	65
Figure 5.14 - Behavior of the Rail Capture dynamic .....	67
Figure 5.15 - Stickville objects manually segmented into their features .....	68
Figure 5.16 - Graph matching results when the Polygon is the input .....	69
Figure 5.17 - Graph matching results when the Letter A is the input .....	69
Figure 5.18 - Graph matching results when the Triangle is the input.....	70
Figure 5.19 - Graph matching results when the Cross is the input .....	70
Figure 5.20 - Graph matching results when the Square is the input .....	71
Figure 5.21 - Graph matching results when the Window is the input .....	71
Figure 5.22 - Model objects used to form graph matcher database .....	72
Figure 5.23 - 5x5 Laplacian of the model objects.....	73
Figure 5.24 - Jet magnitudes of the model objects .....	74
Figure 5.25 - 15x15 vector map trained using 1080 jets from 39 images.....	75
Figure 5.26 - Running average of quantization error over iteration .....	76
Figure 5.27 - 2D histogram of trained vector map.....	76
Figure 5.28 - Feature images of the model objects .....	77
Figure 5.29 - Object graphs of biplane at 0°, biplane at 90°, and car .....	78
Figure 5.30 - Object graphs of the boat and the helicopter .....	79
Figure 5.31 - Plots of g-cell activity for the boat and biplane 0° models .....	80
Figure 5.32 - Plots of g-cell activity for the biplane 90° and car models .....	81
Figure 5.33 - Plots of g-cell activity for the helicopter and elephant models .....	82
Figure 5.34 - Plots of g-cell activity for the football and phone models .....	83
Figure 5.35 - Plot of g-cell activity for the model shoe .....	84
Figure 5.36 - Aspect test inputs .....	85
Figure 5.37 - System response to aspect inputs .....	86
Figure 5.38 - Test inputs distorted .....	87
Figure 5.39 - Biplane match results under elevation perspective distortion .....	88
Figure 5.40 - Biplane match results under variable lighting direction.....	89
Figure 5.41 - Biplane match results under scale and rotation distortion.....	90
Figure 5.42 - Car match results for perspective distortion.....	91
Figure 5.43 - Test inputs of helicopters distorted by partial occlusion .....	92
Figure 5.44 - Helicopter match results under partial occlusion .....	93
Figure 5.45 - Helicopter match results under partial occlusion .....	94
Figure 6.1 - Data densities for the system as implemented .....	102
Figure A.1 - Graphic jet depictions.....	A-2

## List of Tables

Table 4.1 - General relationship between neural correlation and link dynamics.....	43
Table 5.1 - Parameters used to generate Figure 5.14 .....	67
Table 5.2 - Absolute pattern mixing factors .....	95
Table 6.1 - Observed convergence behavior of the graph matcher .....	104

## Abstract

We present a mid-level system for 2D object recognition, comprised of an adaptive feature extractor and a dynamical link graph matcher. The system is robust to variations caused by translation, scale, perspective, lighting, and partial occlusion. The feature extractor performs a Morlet wavelet decomposition that results in a feature vector for each point in the image. The feature vectors are quantized to a set of learned model vectors and labeled. Using a saliency measure derived during the decomposition, an object graph is formed and compared to graphs stored in the associative memory of the dynamical link graph matcher. The graph matcher that has formed the basis for our work comes from [von der Malsburg and Bienenstock, 1987]. That system, not intended as a robust recognition system, was the first to use dynamical links in graph matching and demonstrated a rudimentary invariant recognition capability. We have improved on this graph matcher by providing the capacity for a richer (than binary) feature set and by increasing the processing speed through improved dynamics and a reduction in the number of nodes needed to represent an object. Our system uses a number of biologically plausible mechanisms (Gabor filters, temporal correlations, neural networks) and is in other ways guided by biological principles (local connections, saliency, internal focus of attention, self-organizing topological maps, orientation column tuning). The approach is that of an *unstructured* system, that is, a system in which no domain knowledge is initially embedded but which the system may learn; consequently, the system may be applied to a variety of domains with little or no alteration in architecture or dynamics. Modest extensions to the current system may include multi-modal recognition (fusion), expectation-driven perception, and 3D object recognition.

# Chapter 1 - Introduction

## 1.1 Problem Statement

Consider a child in search of his favorite toy. He enters the nursery and surveys the scene, easily recognizing each familiar object despite a multitude of distortions that affect those objects. Each toy will almost certainly be translated, rotated, and scaled on his retinae with respect to any previous exposure. In addition, the appearance of the objects are likely to be different as a result of changes in perspective and lighting direction. If the toy is of the GI Joe variety, it is likely that its configuration has changed. (Indeed, if the child is normal, there will likely be several formerly rigid objects whose configurations have also been changed.) And finally, if the room is in its normal state, the favorite toy may well be partially occluded. Despite all these fundamental distortions (translation, rotation, scale, perspective, lighting, configuration, and occlusion), the child is easily able to identify the object of his desire.

The ability to visually recognize an object given a variable image representation is a central theme of object recognition. Another highly important aspect of the problem concerns domain. Ten years later, the former child will be driving cars, shopping, writing checks, and scoping babes. Objects must be recognized in these and in countless other domains using the same machinery.

A variety of computational mechanisms for these tasks have been proposed and span the space from low-level iconic recognition to mid-level object recognition to high-level inference- and reason-driven cognition. Indeed, the human visual system is strongly believed to use numerous mechanisms over several levels [Treisman, 1986]. While the development of an artificial vision system that is competitive with the human visual system is an unrealistic goal, it is nonetheless feasible to develop a visual subsystem that performs recognition in a highly invariant manner. It is the purpose of this dissertation to describe such a system.

## 1.2 Background

Computer vision may be characterized as being afflicted with numerous issues. As a group, they have remained largely unsolved due to the ill-posed (i.e., under-specified) nature of the vision problem and its resultant complexity. A major thrust of computer vision research, therefore, has been to solve basic recognition problems such as translation, rotation, scale, and

occlusion distortions at a reduced computational complexity. The literature is thus replete with indexing schemes based on extracted or derived image attributes that endeavor to pare down the space of possible solutions. For example, [Lambdan and Wolfson, 1988] extract *interest points* (e.g., lines of high curvature) for a geometric hashing type of indexing scheme. Non-collinear triplets of these points are used as affine bases for computing the coordinates of all other model interest points with the resulting coordinates stored in a hash table. [Kalvin et al., 1986] extract boundary parts called *footprints*, also for a geometric hashing scheme. [Bolles and Cain, 1982] extract three feature types, along with their sizes and orientations, from binary images: round holes, convex 90° corners, and concave 90° corners. During model acquisition, a unique topological arrangement of features is determined and a *focus feature* is selected. During recognition, the object is searched for the focus feature of each model. When found, the surrounding topology is verified and, if successful, the model is hypothesized to exist at that location. Finally, hypothesized model templates are rotated and translated and matched to the input object.

A multitude of computer vision taxonomies also exist. A system may be described by its ability to handle objects of various complexities: objects representable by lines and vertices (2D) or polygons (3D), objects representable by curves (2D) or curved surfaces (3D), and objects representable only by higher order approximations such as fractals. A system capable of performing recognition on rigid objects may well be found wanting when it comes to flexible or articulate objects, thus another taxonomical dimension. Feature representations may be global, local, or hierarchical and may vary in complexity from simple to complex (e.g., point, line, specific feature, complex feature, functional description). A system may be classified in terms of its computation complexity and may therefore take on descriptors such as exponential (indicating poor performance), polynomial, linear, and sublinear (indicating high performance). Given the system complexity, additional description may be provided in terms of the number of models that it can handle. Viewpoint invariance forms another taxonomy and includes classifications such as translation, rotation, scale, occlusion, and perspective. Much of the significance of our work, however, is not easily measured in the aforementioned terms and so in this section we describe two additional taxonomies.

The first categorization deals with how (and how much) knowledge is embedded in the system; it is a description of the degree of *knowledge structure* imposed by the designer. The two ends of this spectrum are typified by rule-based systems at one extreme and self-organizing systems at the other.

In highly structured systems, the knowledge structure is often in the form of a knowledge base or heuristic that is explicitly programmed by the designer to solve a particular set of problems. Partially because structured systems are often designed for specific applications, their development tends to be easier and their performance better than many adaptive systems. In addition, structured systems have a longer history with a greater level of man-effort involved and so are better understood. The difficulty with structured systems is their tendency to show limited performance outside their specific domains. The result is that new applications tend to require the development of newly handcrafted structure.

Unstructured systems are generally defined by a simple architecture and set of dynamics that, when applied to the architecture, facilitate learning. These adaptive systems, therefore, are initialized with little or no knowledge, but with the ability to accumulate it. Once acquired,

the stored knowledge is a faithful understanding of the environment in which the system was trained and is not subject to the biases and domain understanding of the designer.

The second taxonomy attempts to describe a recognition paradigm in terms of the level of its involvement and relates to a flexibility hierarchy. Specifically, we may speak of low-level iconic recognition, mid-level object recognition, and high-level cognition. Iconic recognition is deemed a low-level process because it works close to the level of the original data and is essentially a template-matching or single-feature-extracting process. It is an expensive, fast, and rigid method that is particularly suited for recognizing objects that are viewed often. High-level cognition is likely to be symbolic in nature and makes use of contextual cues, inference, and reasoning. Cognition is a slow but highly flexible process that is useful when the lower level approaches fail. In between iconic and cognitive recognition is object recognition which is characterized by the aggregation of features resulting in a recognition. This approach is slower than iconic but is more efficient in requiring fewer resources per stored object and provides a good flexibility compromise between high- and low-level paradigms. A well-balanced system might be expected to utilize all three recognition levels.

### 1.3 Approach

The object recognition system described in this dissertation consists of an adaptive feature extractor and a graph matcher. (Appendix B contains a flow diagram for the entire object recognition system.) The function of the system is patterned after the "what" parvocellular pathway found in humans and other primates [Davidoff, 1991] [Mishkin et al., 1983] and is not concerned with object location or number.

The objective of the feature extractor is to reduce an input image to a limited number of salient features. The features, hierarchically representing local image information, are formed from wavelet decompositions of training images and are learned using a self-organizing vector map. Input images, expressed in terms of these learned features, are transformed into object graphs which are used to topologically represent the input object. A dynamical link graph matcher acts as an associative memory and classifies the object graphs relative to a set of graphs stored in its database. For these reasons, we characterize the system as a mid-level recognition paradigm that was described in the preceding section.

The processes used throughout the system are highly non-specific and have no explicit knowledge base designed into them, i.e., they are unstructured. Because the feature extractor is adaptive and because the classifier operates on a common, well-defined representational structure, the system is extremely flexible in terms of object domain. For these same reasons, both the feature extractor and the classifier may be easily used as modular subsystems in other artificial vision paradigms.

The design philosophy that underlies the system derives its inspiration from biology. While the system is in no way a biological model, certain organizations and solutions found in Nature were applied to the development of our system. Sometimes the applications were quite direct, sometimes they were loosely analogous. An example of the former is our use of a

simple edge enhancement technique that mimics the center-surround function of retina [Cornsweet, 1970]. Similarly, the optimal Gabor functions, used for primitive feature extraction in our system, are found in Nature [Jones and Palmer, 1987] [Burr, Morrone, and Spinelli, 1989] and are believed to perform a similar role there [Daugman, 1988]. The use of dynamical links for binding in the graph matcher is also a plausible function of biological nervous systems [von der Malsburg, 1981] [Crick, 1984]. Our system makes use of a low-level saliency measure to improve efficiency by processing features on a salience-priority basis. In addition, saliency is used to both select features for inclusion in the representational graphs and to guide the graph matching process in a micro-saccadic manner thereby causing shifts in the internal focus of attention [Treisman and Gelade, 1980] [Crick, 1984]. The saliency measure that is employed behaves in a manner consistent with the results of psychophysical experimentation [Atteneave, 1954] and is also consistent with the information theoretic view of salience [Gonzales and Wintz, 1987].

Perhaps the most obvious relationship between our system and the biological motivation is the neural style of much of the architecture. In addition to the edge enhancement and Gabor filtering, the system employs a self-organizing vector map, vector quantization, and dynamical link graph matching. All of these are either neural paradigms or, at least, easily cast into a neural framework. The use of neural networks in our system is not driven entirely by a sense of aesthetics but more by the fact that the neural approach provides good engineering alternatives to existing methods. These benefits are widely discussed in the literature and include parallelism, eventual direct hardware implementation, fault tolerance, learning, and a natural style of solution that often yields good answers, rather than best answers, at a substantially improved algorithmic rate.

Our system is not put forth as the solution to the object recognition problem. It does, however, provide a powerful method for doing mid-level object recognition in which the object image may be subject to a variety of distortions. In particular, the system is robust with respect to translation, scale, perspective, lighting direction, and occlusion. When compared with other methodologies, our system shows deficits specific to its categorization as a mid-level recognizer. In particular, low-level iconic recognition processes are considerably simpler and faster whereas high-level cognition systems may make use of powerful contextual cues. We do not view these methodologies as competing approaches but rather as complements. While our system may function well as a stand-alone recognizer, it would be advantageous, in many applications, to include it as a part of a larger system incorporating low-level iconic recognition and high-level inference-driven cognition. As a stand-alone recognizer, however, it provides an appropriate tradeoff between speed and flexibility and between simplicity and power.

## 1.4 Related Work

Object recognition systems abound with many possessing unique combinations of traits. One common thread, however, is that they may be generally broken into two functions: feature extraction and classification. Our system employs an adaptive feature extractor, consisting of Morlet wavelet decomposition and adaptive vector quantization, and a classifier that performs

graph matching. In this section we discuss only recognition paradigms that have specific commonality with ours.

2D Gabor functions for feature extraction have become increasingly popular of late. One of the most prolific users of Gabor functions, though more for image compression than feature extraction, is Daugman who has demonstrated their near orthogonality [Daugman, 1989] and their superiority in compression tasks [Daugman, 1987]. It should be noted that orthogonality is less of an issue in feature extraction than it is in data compression. The latter is concerned with image reconstruction and the ease that orthogonality provides in that reconstruction. Feature extraction is concerned with orthogonality only in the sense of redundancy and, thus, the efficiency of feature encoding.

The use of Gabor functions for feature extraction has been demonstrated by [Buhmann, Lange, and von der Malsburg, 1989]. In their system, 48 Gabor kernels of 6 orientations and 8 spatial frequencies (with half-octave separation) are convolved with an input image. The resulting components are collected into feature vectors, a subset of which are selected for object representation using a fixed sampling grid. The chosen feature vectors are then compared to stored sets of feature vectors (where each set corresponds to a stored pattern) one at a time using an energy minimizing diffusion algorithm. The pattern with the lowest resulting energy is deemed to be the closest match. Using the same recognition scheme, [Buhmann, Lades, and von der Malsburg, 1990] demonstrated that scale invariance may be achieved by shifting the feature vector elements that correspond to spatial frequency, until a low energy match is found.

Boundary detection on greyscale and texture images was demonstrated by [Manjunath and Chellappa, 1991] using Gabor functions. Their approach was highly motivated by studies of cat visual cortex and involved local scale and orientation interactions to produce hypercomplex [Hubel and Wiesel, 1965] or end-stopping cells.

The second half of our system uses dynamical link graph matching to perform classification. Because procedural graph matching algorithms are so plentiful and because they are not particularly relevant to our work, we review here only neural graph matchers.

[Kuner, 1989] demonstrates that graph matching using a Hopfield network is a natural outgrowth of Ullmann's definition of subgraph isomorphism [Ullmann, 1976]. Although Kuner suggests that his graphs may be construed as image representations, the work is general and successfully finds subgraph isomorphisms in randomly generated graphs.

Another application of Hopfield to graph matching is demonstrated by [Parvin and Medioni, 1991] in which edge and region features of synthetic images are extracted. These features comprise graph nodes that are connected by short and long range edges. The intent of the system is to form complete boundaries with the aid of local (short range edges) and global (long range edges) constraints.

A system for learning and matching 3D object graphs was demonstrated by [Reiser, 1991]. In that work, a 3D *multiview fusion graph* was learned by the system as an object was slowly rotated in front of the camera. At each frame, surface features were extracted and added to the forming graph. Features that were noisy or otherwise unreliable died off whereas stable features remained and became permanent. Graph matching employed feature and neighborhood

similarity measures as opposed to our use of correlated neural activity and employed a modified Hopfield approach. The features used by Reiser were predominantly region-based in contrast to our system which uses locally salient, Gabor-derived features.

It should be noted that the Hopfield approach differs significantly, and at a fundamental level, from the dynamical link approach that we have used. In general terms, the objective of both approaches is to bind subgraphs from an input plane to subgraphs of a model plane. Connections of some kind must therefore exist between both planes. In the case of dynamical link graph matchers, the connections take the form of variable strength links. Input to model plane bindings occur by increasing dynamical link strengths as a function of the temporal correlation of neural activity. In the Hopfield approach, an array of "binding" neurons takes the place of the dynamical link matrix with each binding neuron requiring application-specific connections to other binding neurons. The advantage of the Hopfield approach is in its parallelism and increased degree of freedom. The advantage of the dynamical link approach is in the simplicity of the architecture and its relatively reduced computational complexity.

The dynamical link graph matcher that has formed the basis for our work comes from [von der Malsburg and Bienenstock, 1987] and [Bienenstock and von der Malsburg, 1987]. Its architecture consists of two neural planes and has connections within and between the planes. One plane is termed the input plane and contains the input graph that is to be matched. The second plane is termed the object plane and contains a database of stored object graphs. The link dynamics are implemented as two Hamiltonians that are minimized in a Monte Carlo style. The system, not intended as a robust recognition system, was the first to use dynamical links in graph matching and demonstrated a rudimentary invariant recognition capability. It was our intention to improve on this graph matcher by providing the capacity for a richer (than binary) feature set and to increase the processing speed by improving the dynamics and reducing the number of nodes needed to represent an object. This dissertation will demonstrate that we have done both.

## 1.5 Scope of Work

The main purpose of the work is to develop an unstructured object recognition system, i.e., a mid-level system with little or no domain knowledge incorporated into it. Such a system should be capable of learning the relevant features of a specific domain and thus be adaptive over various environments and sensors.

The system should perform robust 2D object recognition. Distortions under which the system should perform include translation, scale, perspective, lighting, and partial occlusion.

The system should be complete in the sense that, with minimal external processing, it should be capable of acting as a stand-alone recognizer. At the same time the system should form a solid foundation for extension and should be complementary to higher- and lower-level vision processes. In particular, the system should be amenable to extension by data fusion processes and top-down guidance. Finally, the system should find a natural placement in a complex vision system comprising low-, mid-, and high-level paradigms.

## 1.6 Summary of Results

The system was tested with a database of nine objects derived from greyscale images. Undistorted objects and objects distorted by scale, perspective, rotation, partial occlusion, and lighting aspects were presented to the system. All were correctly recognized except one. The exception was the rotationally distorted object; in that case, the correct model came in second. Enhancement of the storage capacity of the system, by the sharing of resources (nodes), was also tested. Under the highest possible sharing criteria, undistorted object recognition remained completely intact while distorted object recognition was significantly degraded.

## 1.7 Organization of the Dissertation

Chapters 2 through 4 contain the theory and description for each of the processing modules. Chapter 2 covers the adaptive feature extraction that is composed of Morlet wavelet decomposition and adaptive vector quantization. Chapter 3 describes the method by which an array of features, produced by the feature extractor, is transformed into an object graph. Chapter 4 describes the classifier, i.e., the dynamical link graph matcher. Chapter 5 shows the computational results from two problem sets. The first set is derived from Stickville and assorted real images and is intended to provide insight into the mechanisms involved in each processing module. The second problem set consists of a nine-object greyscale gallery and forms the proof of concept for the dissertation. The dissertation concludes with Chapter 6 which summarizes the system and the results and examines some limitations and their solutions. Following Chapter 6 are a series of appendices that contain functional flowcharts for each of the processing modules as well as information regarding the parametric options available to each processing module.

## Chapter 2 - Adaptive Feature Extraction

The purpose of feature extraction in artificial vision is the transformation of an input image into a representational set of elements that facilitate classification or recognition. The definition of that representational set, and therefore of the transform itself, is often highly dependent upon such considerations as the object classes and their interrelationships, the nature of the clutter and noise processes, the image modality, and the subsequent recognition processes.

A portion of our work has resulted in a feature extraction process that, when coupled with the specific graphic processes described in Chapters 3 and 4 of this dissertation and applied to greyscale imagery, provides a set of features that is robust with respect to changes in translation, perspective, partial occlusion, scale, and lighting direction. Perhaps more interesting is that the feature set is adaptive with respect to the objects and imagery with which the system is expected to perform.

Adaptive feature extraction proceeds in two phases: learning (adaptation) and extraction. During the learning phase, the system generates a rich but discrete set of features that are later used during the actual extraction process. The feature learning and extraction processes are similar. (See the Algorithm Flow Diagram in Appendix B.) In both phases, processing begins with edge enhancement and is followed by Morlet wavelet decomposition. In the adaptive phase, the computational results from a large number of decompositions are used to train a self-organizing vector map. In the extraction phase, the image decomposition results are compared with the learned vectors and are transformed into features representing the specific input data.

This chapter discusses the mechanics of the adaptive feature extractor<sup>1</sup> by its constituent parts: edge enhancement, wavelet decomposition, and vector quantization (adaptive and extractive).

### 2.1 Morlet Wavelet Decomposition

The purpose of the *Morlet wavelet decomposition (MWD)* module is to extract primitive image features. It is accomplished by decomposing the image into a number of elements per pixel, where each element is a primitive feature describing the existence of discontinuities of specific orientation and spatial frequency at or near the image pixel point. The MWD is a transform based on 2D Gabor functions. Combining the primitive features into Morlet jets provides both

---

<sup>1</sup> It is recommended that the reader review Appendix J for an introduction to data structures fundamental to the feature extractor.

a powerful representational structure and a means to determine local saliency. The saliency measure is used in later processing to improve efficiency and to guide the graph matching process.

### 2.1.1 Gabor Functions

Gabor functions have a number of interesting and desirable properties including extended spatial locality, directional preference, optimal joint uncertainty in space and frequency, and biological plausibility.

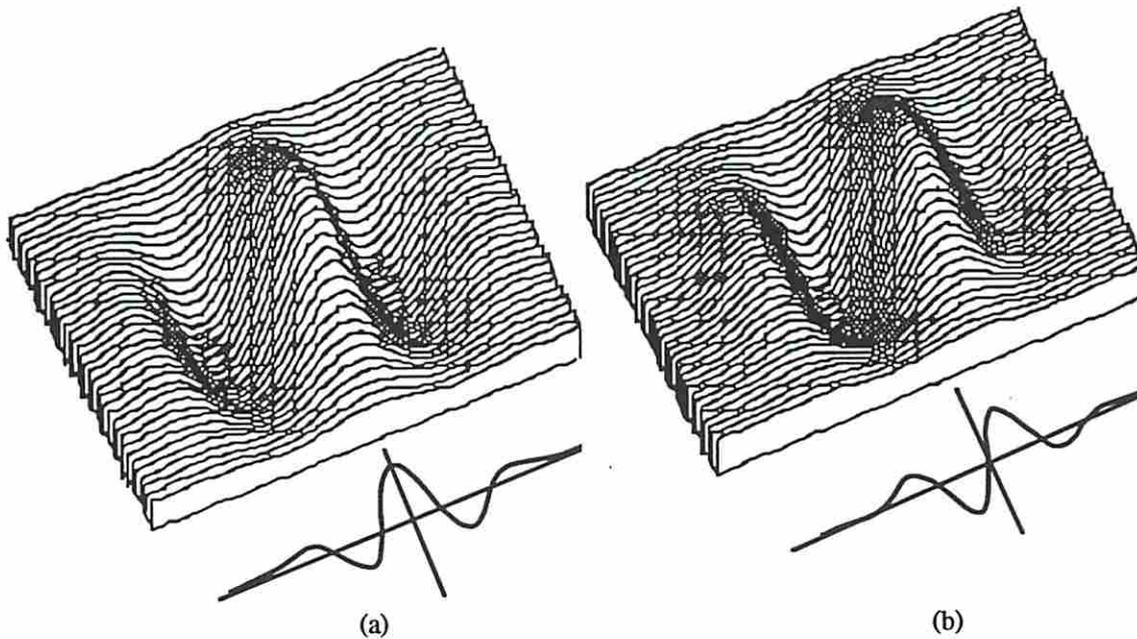


Figure 2.1 - 2D complex Gabor function; (a) real (cosine) part, (b) imaginary (sine) part.

Gabor functions were originally introduced in the context of uncertainty theory for information [Gabor, 1946]. Gabor showed that a signal has a theoretical lower limit for joint uncertainty in time and frequency when the 1D sampling functions have the form of complex sinusoids with Gaussian envelopes. Later, the 1D Gabor function was extended to two dimensions in an attempt to capture the neurobiological variables of a neuron's orientation and spatial frequency preference [Daugman, 1980]. Figure 2.1 shows the real and imaginary parts of a 2D Gabor function. Gabor functions have recently reemerged in partial response to the debate of whether early vision involves local space-domain feature detection or more global, Fourier-like spatial-frequency decomposition. The Gabor formulation provides a suitable compromise by virtue of its extended locality property<sup>2</sup>. In addition, it is a biologically

---

<sup>2</sup> Extended locality refers to the fact that the Gabor function is non-global but at the same time may cover extensive portions of the image.

plausible feature extraction mechanism; it fits extremely well with empirical studies of visual receptive fields in the cat striate cortex [Jones and Palmer, 1987] and is consistent with psychophysical evidence of even- and odd-symmetric feature detectors in the human visual system [Burr, Morrone, and Spinelli, 1989]. In addition, these functions are amenable to group transformation operations (especially rotation and dilation) thereby making them candidates for wavelet kernels.

The 2D Gabor function,  $\Psi$ , consists of the product of Gaussian and sinusoid terms:

$$\Psi_{\mathbf{k}}(\mathbf{x}) = \exp\left(-\frac{|\mathbf{x}|^2}{2\Sigma^2} + j\mathbf{x}^t\mathbf{k}\right) \quad (2.1)$$

where  $\mathbf{x}$  is the  $(x,y)$  position,  $\mathbf{k}$  is the two dimensional wave vector, and  $\Sigma = \sigma / |\mathbf{k}|$  regulates the size of the Gaussian window. Orientation and spatial resolution of the filter are determined by  $\mathbf{k}$ : a larger  $|\mathbf{k}|$  corresponds to a higher frequency whereas orientation is specified by the relative values of the components of  $\mathbf{k}$ , that is,  $k_x$  and  $k_y$ . The spatial extent of the filter is governed by  $\Sigma$ : a larger sigma increases the size of the Gaussian window. The Gabor filter, thus, has directional specificity and extended locality.

In our system, Gabor functions are used for primitive feature extraction. The function is used to generate a kernel of specific orientation and spatial frequency that is then convolved with an input image. The computation yields complex Morlet wavelet components for each point in the image. Each component corresponds to the existence of Gabor shaped features in the image, i.e., an oriented line (real part of the kernel) and/or an oriented edge (imaginary part of the kernel).

### 2.1.2 Wavelet Decomposition

A 2D wavelet transform may be described as one that changes a variable of two dimensions (spatial) to one of four dimensions (two spatial and two frequency). The transform is accomplished with a family of self-similar basis functions that are different from each other only by group transformation, in our case, dilation and rotation. For Morlet wavelets, the basis functions are Gabor functions, that is, a family of Gabor functions of varying orientation and spatial frequency form the convolution kernels of the transformation. Each Gabor function is generated from an original by scale and rotation transformations and are self-similar. The self-similarity is preserved by maintaining the Gaussian window of the Gabor function as a function of the spatial frequency; in our case,  $\Sigma = \sigma / |\mathbf{k}|$  in Equation 2.1. The Morlet wavelet transform should not be confused with the Gabor transform in which the window has a fixed size over all spatial frequencies, i.e.  $\Sigma$  is constant.

A Morlet wavelet transformation is accomplished by convolving a 2D Morlet wavelet, parameterized by  $\mathbf{k}$ , with the image:

$$W_M(\mathbf{k}, \mathbf{x}_0) = \iint \Psi_{\mathbf{k}}(\mathbf{x}_0 - \mathbf{x}) \cdot I(\mathbf{x}) d^2\mathbf{x} = \Psi_{\mathbf{k}} * I \quad (2.2)$$

where  $I(\mathbf{x})$  is the array of grey-level pixels representing the 2D image. A Morlet wavelet decomposition (MWD) is a family of Morlet wavelet transformations (parameterized by  $\mathbf{k}$ ) of an image. The computational results of the MWD are 2D arrays of components, one array for each transform. Each element in an array may be viewed as the response of a Gabor filter to the image from that image point.

A MWD imposes a hierarchy of resolutions. In our case, each resolution is separated from the next by one octave. For each of  $S$  spatial resolutions there are  $O$  different orientations so that a family of Morlet wavelet components (from the same point) is formed with of a total of  $S*O$  transforms. The hierarchical nature of the MWD facilitates the stacking of components from a single image point into feature vectors called *jets*. See Figure A.1a (Appendix A). A jet, therefore, is the Gabor-filtered image from the perspective of the jet's position<sup>3</sup>. Figure 2.2 shows an original image and the reconstruction by [Buhmann, Lange, and von der Malsburg, 1989] of a jet taken from the center of an eye. Note the high resolution close to the center of the region and the lower resolution farther out. This is due, in part, to the self-similar nature of the kernels: as the spatial frequency,  $\mathbf{k}$ , is reduced,  $\Sigma$  (and thus the Gaussian window) is increased thereby preserving self-similarity. The result is that the lower spatial frequency kernels have greater spatial extents.

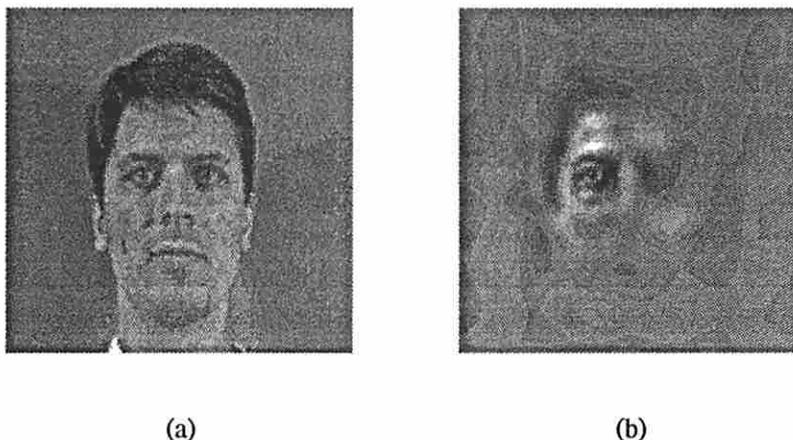


Figure 2.2 - Image representation using a jet. The jet selected from the left eye of (a) is inverse transformed and shown in (b). Taken from [Buhmann, Lange, and von der Malsburg, 1989].

If a jet is formed for each pixel in the image, the resulting data structure is termed a *Morlet block*<sup>4</sup>. See Figure 2.3b. A Morlet block, computed with  $S=4$ ,  $O=6$ , and an image size of  $128 \times 128$ , would require almost 800,000 elements. A more compact data structure involving Gabor functions is the *Morlet pyramid*. See Figure 2.3a. As the spatial frequency of the Morlet

<sup>3</sup> An outgrowth of the Gabor and Morlet definitions is that Gabor components and Gabor jets are similar to Morlet wavelet components and Morlet jets (respectively) with the only difference being that of self-similarity. This terminology is a departure from our previous writings in which Gabor components and Gabor jets implied the self-similarity now attributed only to the Morlet definition.

<sup>4</sup> Previous writings have used the terms Gabor blocks and Gabor pyramids for what we now call Morlet blocks and Morlet pyramids. The reason for this change is in the evolving nature of the terminology and is consistent with the terms detailed in the previous footnote.

wavelet is decreased, its resolution is also decreased thereby allowing a reduced sampling rate and the storage of a reduced number of elements at that resolution. That is the essence of the Morlet pyramid: the highest resolution elements are sampled at the highest rate and stored at the base while lower resolution elements are sampled at progressively lower rates and stored progressively nearer the apex.

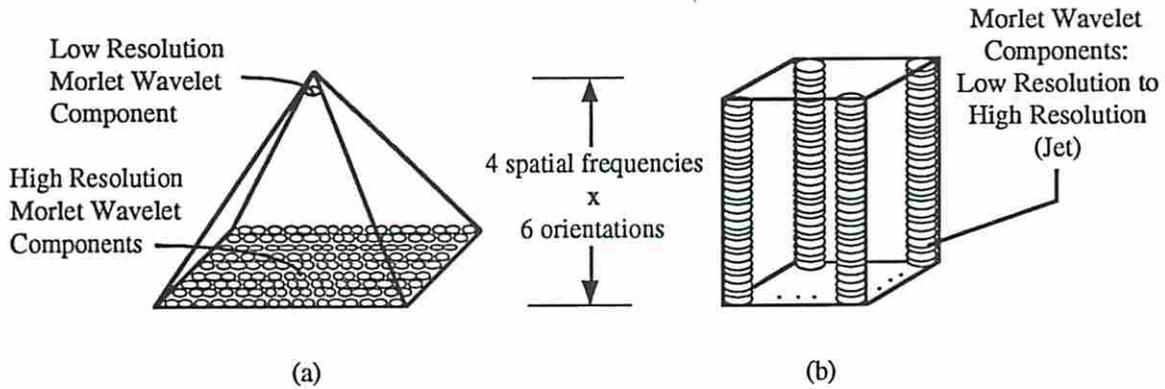


Figure 2.3 - Alternate Gabor-filtered image representations (a) the Morlet pyramid contains a large number of high resolution Morlet wavelet components at the base with progressively fewer and progressively lower resolution components toward the apex, and (b) the Morlet block contains  $N^2$  jets.

There are, however, several disadvantages to the pyramid structure. The most serious deficiency is that the components are not explicitly stored as jets. Since jets are required for recognition, they must be constructed from the available information from within the pyramid. A jet for a given point in the image can be formed by:

- 1) Using the components whose center is closest to the desired image point.
- 2) Performing a linear interpolation between the closest components.
- 3) Performing a higher order interpolation (such as a cubic spline) of the components.

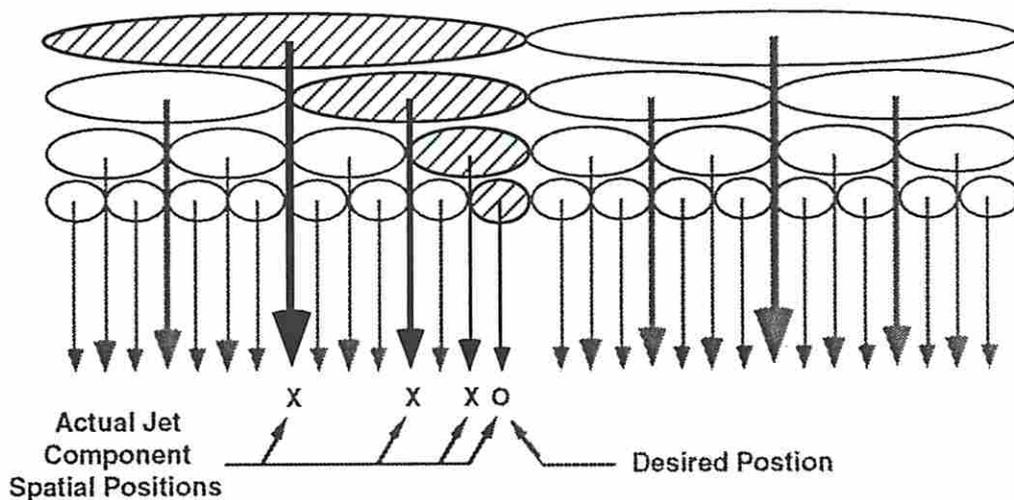


Figure 2.4 - Misalignment of Morlet wavelet components in a Morlet pyramid. Due to the variable sampling rates per spatial frequency, jets constructed from Morlet pyramids cannot be formed using the natural components.

The first option leads to an inaccurately reconstructed jet. See Figure 2.4. Performing linear interpolation is time consuming and inadequate in estimating components. Performing higher order interpolations on so many data points at so many levels is impractical from a complexity standpoint. The point is that implicitly storing jets, as is done in pyramids, leads to inaccurate or computationally expensive jet reconstruction. We have therefore rejected the use of Morlet pyramids in favor of Morlet blocks.

In our work, we have chosen  $O=6$  orientations and  $S=4$  spatial frequencies with separations of one octave. The choice of  $S=4$  was motivated by psychophysical evidence for four spatial frequency detection bands in the human visual system [Wilson and Bergen, 1979]. In addition, [Daugman, 1987] demonstrated that jets of such parametric configuration ( $O=6$ ,  $S=4$ ) contain sufficient information to perform good data compression (100:1) and reconstruction.

### 2.1.3 Laplacian Preprocessing

It was shown by [Field, 1987] that in "natural images," the power spectrum is inversely proportion to frequency squared,  $1/|k|^2$ . As a result, the wavelet decomposition (using the basis functions of Equation 2.1) typically exhibits stronger response at lower frequencies than at higher frequencies. This over-emphasis of low spatial frequencies produces undesirable effects, especially with respect to illumination-induced intensity gradients that makes the system sensitive to lighting direction. In addition, the structure of an object is best described by its edges which are primarily expressed by high frequency components. It is therefore desirable to flatten the processed power spectrum to de-emphasize illumination gradients and accentuate edges.

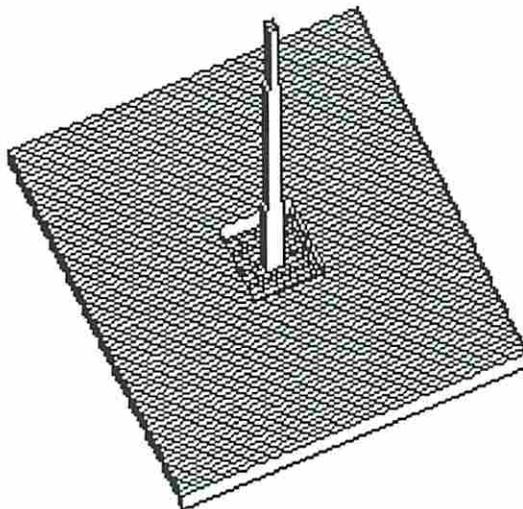


Figure 2.5 - Laplacian mask shape. The peak corresponds to a single pixel with a value of 24; the base is comprised of 24 pixels with a value of -1. This approximation of a Laplacian is used, in our system, for edge enhancement.

One solution is to modify the wavelets such that the energy for each kernel is proportional to  $|k|^2$  [Lades et al., 1991] thereby offsetting the frequency distribution of the image. Our

approach is to use an edge enhancement process on the input image to attenuate lower spatial frequencies and to amplify higher ones in rough approximation to  $|k|^2$ . The edge enhancement process used is based on the 5x5 Laplacian operator shown in Figure 2.5 and is intended to mimic the center-surround processing found in retina and lateral geniculate nucleus. This use of a distinct preprocessing step also aids in system development and understanding in as much as the effects of high frequency amplification and low frequency attenuation can be explicitly observed.

The result of the Laplacian filtering may be described as a frequency dependent gain applied directly to the wavelets. In Fourier space, the Laplacian of Figure 2.5 is a negative sinc function (due to the -1 pit) with a broad band component (contributed by the impulse). Similarly, in Fourier space, the Morlet wavelets (exemplified by Figure 2.2) are Gaussians, the centers of which are shifted from DC by spatial frequency,  $k$ . The spatial frequencies that we use for our wavelets all fall within the mainlobe of the -sinc function. As a result, each wavelet is associated with a gain that is a function of its spatial frequency and derives from the continuous portion of the -sinc. Figure 2.6 shows the frequency dependent gain applied to each wavelet as a result of the Laplacian filtering. The effect is an attenuation curve that roughly approximates  $|k|^2$  but which has more biological coincidence than direct modification of the wavelets.

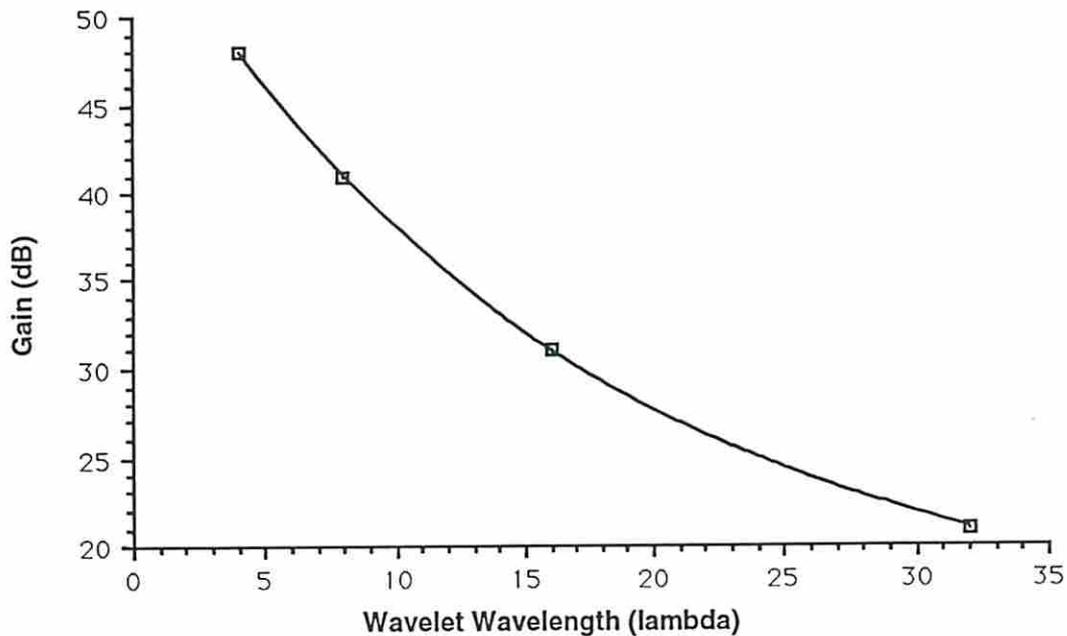


Figure 2.6 - Wavelet gain resulting from Laplacian edge enhancement as a function of spatial wavelength. Wavelet wavelengths are at 4, 8, 16, and 32 pixels. Calibration is such that a gain of 0 dB is applied at DC.

#### 2.1.4 Real vs. Complex Decompositions

While all the theory and results discussed in this dissertation assume the use of the complex Gabor function, much of our earlier work [Flaton and Toborg, 1989] [Flaton, 1991] used real

Morlet wavelets, i.e., only the cosine portion of the Gabor function. As we were attempting to develop higher level features from the primitive Morlet wavelet components and Morlet jets, we discovered that certain types of image features were indistinguishable when filtered with real Morlet wavelets. Figure 2.7 shows five such indistinguishable features under a real wavelet decomposition as well as their jet image representations. Specifically, those features are four  $90^\circ$  corners and a  $90^\circ$  cross. The reason for the feature ambiguity is simple: in the real kernel, the feature extractors are simple line detectors. In each case, the Morlet wavelet components corresponding to  $0^\circ$  and  $90^\circ$  lines are activated and although there may be a magnitude difference in the component values, there is still insufficient information to distinguish the features.

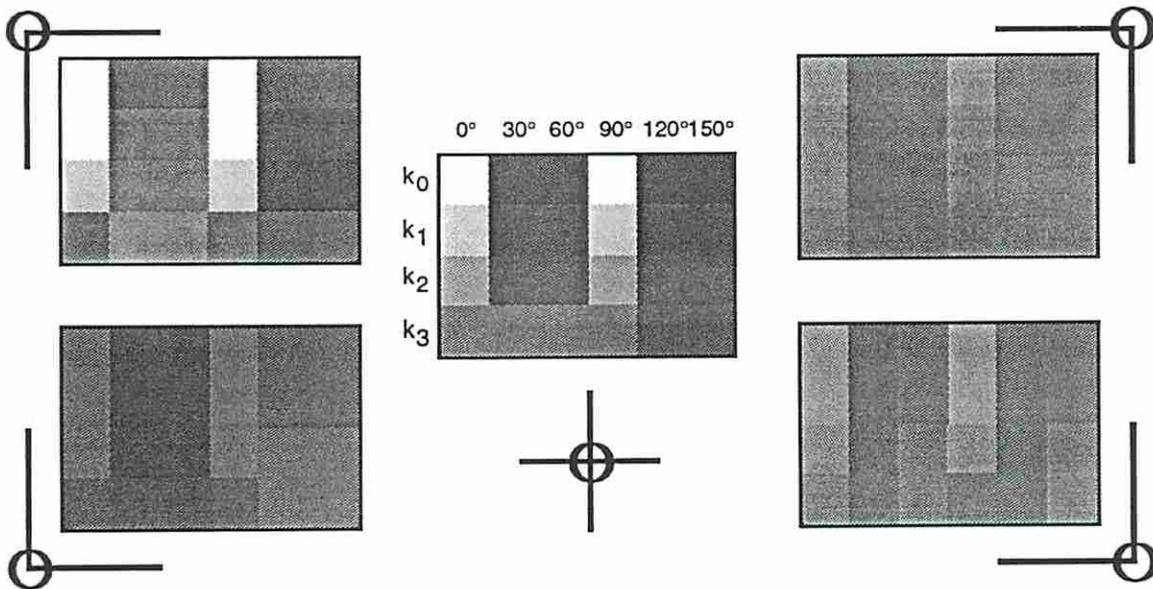


Figure 2.7 - Feature ambiguity problem using real wavelets. Each jet image shows the response of the (real) Gabor filters for each of five features: four corners and a cross. The responses are approximately the same inasmuch as the strongest responses consistently come from the 0 and 90 degree filters. Jets were taken from the center of each vertex (circled area).

In our thesis proposal, we suggested the use of neighborhood information to resolve the feature ambiguity problem noted above. Specifically, we were interested in using something similar to the Neocognitron [Fukushima, 1988] (we called our version a Hierarchical Feature Builder) which essentially template matched neighborhoods in a hierarchical fashion. Very briefly, the idea was that the Hierarchical Feature Builder (HFB) would learn significant and topological jet clusters and, when presented with similar clusters, would respond with the closest stored cluster (represented by a high level feature label). It was expected (and confirmed) that a jet at the vertex of a corner-type image feature would show a crossing type of feature while its neighbors would confirm the existence or absence of connecting collinear lines.

After several months of intensive work with a Neocognitron-like network, we rejected the concept of the Neocognitron/HFB in favor of using complex Morlet wavelets and adaptive vector quantization. The reasons were:

- 1) Learning, using the Neocognitron rules, is glacial [Hecht-Nielson, 1990] and is prone to imprinting [Tacket and Lincoln, 1991].
- 2) The computational resources required are immense. For each sub-feature that the system is designed to be able to detect, there must be one entire plane of detectors each with enough synapses to cover its receptor field. In addition, the Neocognitron/HFB is a multi-stage system requiring a number of such multi-plane groupings. We did not see a realistic way of reducing this problem by implementing the network in dedicated hardware.

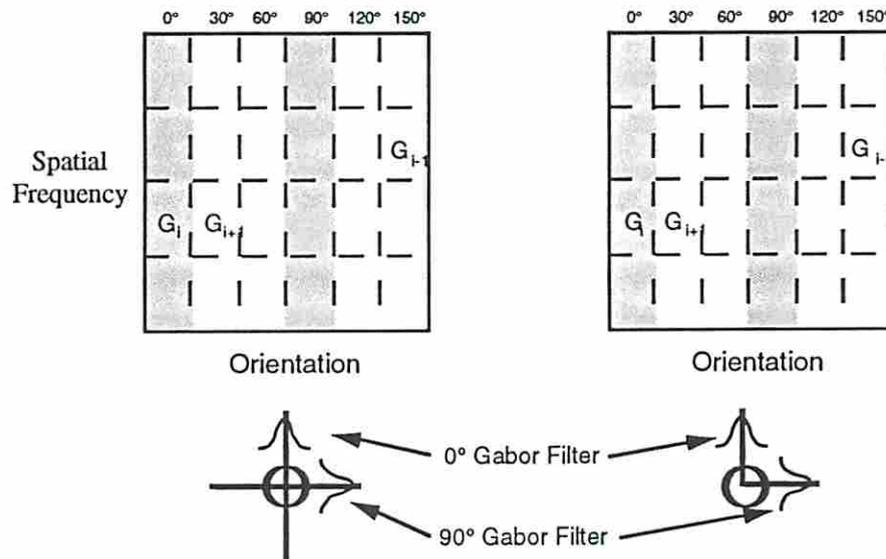


Figure 2.8 - Nature of feature ambiguity problem with real (even-symmetric) Morlet wavelets.

In addition, complex jets detect endstopping which is a sufficient condition for solving the feature ambiguity problem. In other words, a single jet located at the vertex of either the corners or cross described above, contains sufficient information for a vector quantizer to differentiate the features. The reasoning is as follows. See Figure 2.8. The real portion of the kernel performs as before, that is, as an oriented line detector. In our example, it will detect the  $0^\circ$  and  $90^\circ$  lines in each case. The imaginary portion performs the endstopping. See Figure 2.9. Since the imaginary kernel is odd-symmetric, the total contribution to these  $0^\circ$  and  $90^\circ$  filters from the cross is zero due to equal amounts of positive and negative response. For each of the corners, however, the odd-symmetric  $0^\circ$  and  $90^\circ$  filters will, in aggregate, respond uniquely.

The description of the effects of the imaginary Morlet wavelet on Stickville imagery is particularly appropriate to our application. Because the input greyscale image is edge enhanced by a Laplacian operator, the image presented for wavelet decomposition approximates a line drawing. While the lines are neither smooth nor perfectly aligned with the directional Gabor filters, the use of one filter for every  $30^\circ$  provides sufficient angular resolution to characterize the image features.

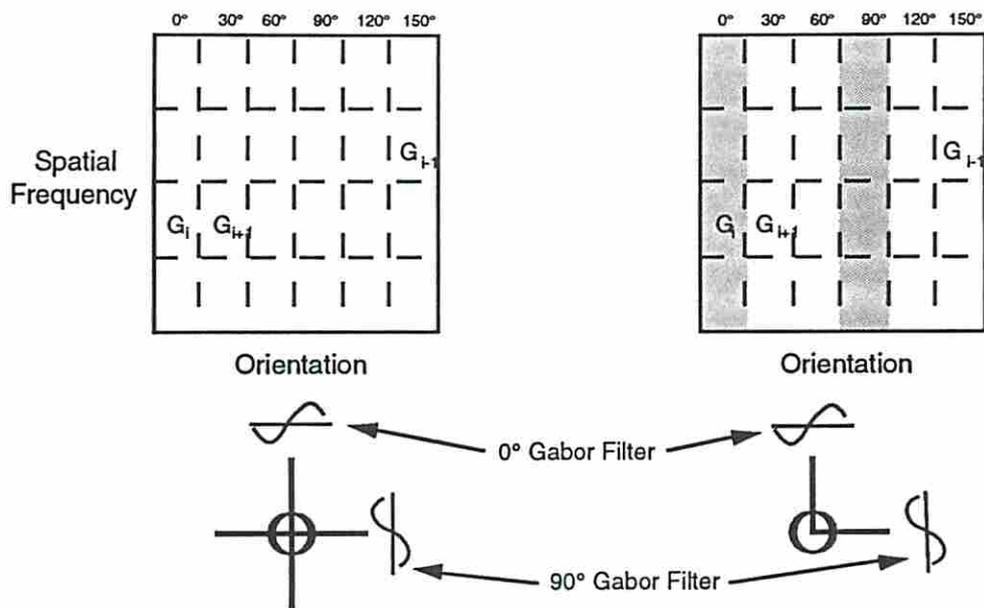


Figure 2.9 - Complex kernel solution to the feature ambiguity problem. Imaginary (odd-symmetric) Morlet wavelets provide endstopping.

Finally, it was our feeling that the extension from real to complex Morlet wavelets was systemically simple and more aesthetically pleasing than the inclusion of an additional processing stage bearing the deficits associated with the Neocognitron.

### 2.1.5 Orientation Column Tuning

The volume of the vector space spanned by the Morlet jets formed in the above described manner is generally quite small. Because ultimately we must be able to reliably classify these jets (using adaptive vector quantization), we desire that the space spanned by the jets be as great as possible. To accomplish this, we employ a method that we call *orientation column tuning* that is motivated by biological mechanisms described by [Hubel and Wiesel, 1974].

Quite simply, the idea is to create competition between neighboring orientation components of the same spatial frequency for the purpose of tuning or peaking the responses of the dominant orientations. In our case, the use of the word *columns* refers to the columnar organization of the orientation components as shown in the jet image representation of Figure A.1b (Appendix A). The range of the competition is extremely local: the mask used is  $\{-1, 2, -1\}$ . Note that this is not a winner-take-all strategy as there is only a single convolution. The intent is only to enhance orientation differences and not to cause one orientation to prevail. Figure 2.10 shows a typical response for a jet and the result of orientation column tuning.

(A simple extension of this approach, and one that we have not tried, is to create cooperation along the orientation columns, across spatial frequencies. We would expect the results to improve response to noise in the spatial frequency domain; this has not been a problem with our data sets and so is still a matter of conjecture for us at this point.)

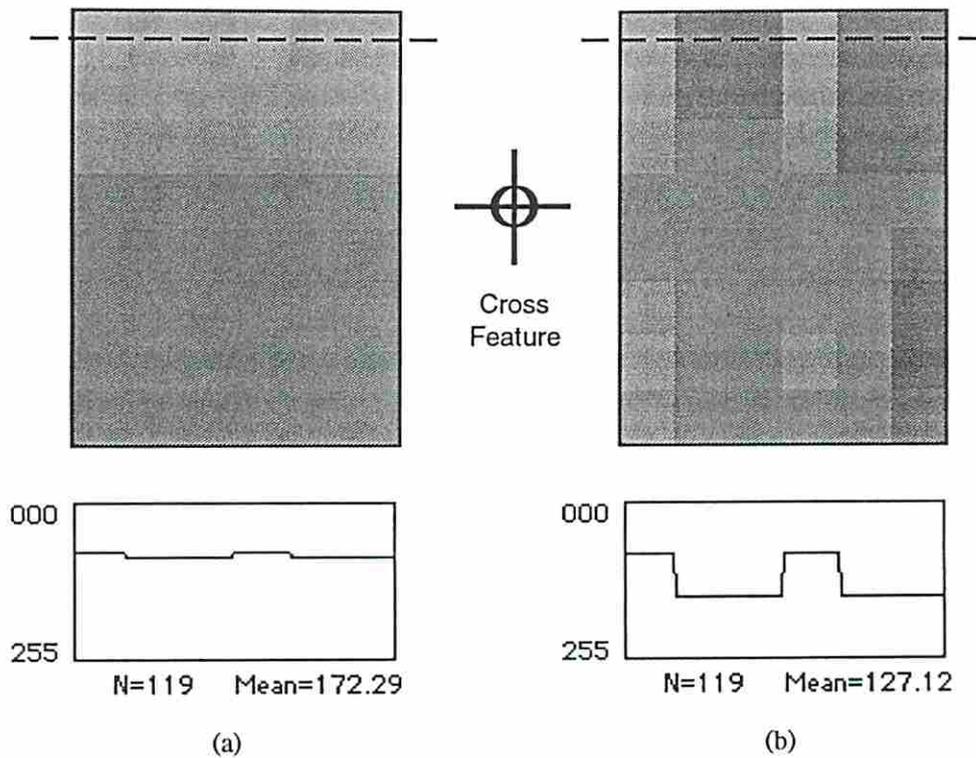


Figure 2.10 - Orientation column tuning by convolution with  $\{-1,2,-1\}$ . (a) jet image of cross-feature jet before tuning and, (b) after tuning. Note the increase in dynamic range resulting from competition of neighboring orientation components. Cross-section plots are taken at the lowest spatial frequency over all orientations (dashed line).

### 2.1.6 Saliency and Jet Magnitudes

Both information theory and psychophysics indicate that image discontinuities are important, i.e., lines and edges are salient image features. A visual processing system may make use of image salience in a variety of ways including: saccades, shifts of internal focus of attention, increased processing efficiency, and reduction in storage requirements.

The use of the MWD on edge enhanced images allows us to define a saliency measure that demonstrates large values at points corresponding to discontinuities or features in the image. The measure that we employ is simply the jet magnitude. Recall that a jet contains the responses to a bank of edge and line filters of varying orientation and spatial frequency. One would expect that such a vector would have large component values, and thus a large vector magnitude, if some number of filters were responding to stimuli. The more filters responding, the larger the vector magnitude. This is, indeed, a phenomenon that is observed in the data as shown in Chapter 5 where vertices have larger jet magnitudes than lines which, in turn, have larger magnitudes than background.

The jet magnitude saliency measure is used in a number of capacities and is discussed in the relevant sections. The use of saliency in our system includes: training jet selection, AVQ update gain modification, and DyLink spotlight control (shifting of internal focus of attention).

Jet magnitudes that form the saliency measure are not necessarily the magnitudes of the jets that form the image representation. Briefly, it is a problem of resolution: we desire high resolution for saliency and lower resolution for image representation. The jet magnitudes used for saliency are typically magnitudes of jets containing responses one octave higher than the jets used for image representation. This is discussed more fully in Chapter 3.

### 2.1.7 Algorithm Implementation

The program that we have written to perform MWD (which also performs vector quantization and a number of other related functions) is known as *GTVQ* for Gabor Transform - Vector Quantization. The general program flow for *GTVQ*, the MWD, and related processing is shown in Appendix C. Macintosh dialog boxes (and therefore processing options and parameters) for *GTVQ* are given in Appendix G.

The information provided in the aforementioned appendices is generally self-explanatory with a single exception: the method by which the Morlet wavelet transforms are computed. Convolution is normally an  $O(n^4)$  operation. However, under certain conditions, the convolution can be performed using 2D FFTs in  $O(n^2 \log n)$  time. *GTVQ* performs both real and complex MWDs using FFT convolution. (By way of anecdote, a brute force convolution running on the current platform, a Mac IIfx, required more than two hours to perform twenty-four 128x128 Morlet wavelet transforms. The FFT version runs in 6 minutes.) One caveat that must be considered when using FFTs, however, is that the data sampling rate must be high enough to prevent aliasing. *GTVQ* provides the option of zero-padding the arrays to prevent this problem.

## 2.2 Adaptive Vector Quantization

A Morlet block contains  $N^2 * S * O$  data elements and is thus quite memory expensive. It is desirable to reduce this raw data to some set of elemental features such as oriented lines and vertices or perhaps even to complex features such eyes and noses. One must consider also that the final stage of processing (i.e., the graph matcher) prefers scalar labeled graphs. The conversion of jets to scalars may be accomplished by the vector quantization of jets to known model vectors.

By viewing the jet not as the vector of Figure A.1a (Appendix A), but as a two-dimensional jet image as in Figures A.1b (Appendix A) and 2.10 it becomes obvious that certain types of features can be extracted from those feature matrices. For example, Figure 2.11a shows the real jet image representation of a 60° line; Figure 2.11b shows the real jet image representation of 60° and 120° lines, i.e., an intersection.

Figure 2.12 shows some elemental features that can be and were extracted from jets using vector quantization during early stages of our research [Flaton, 1991]. At that time, we used handcrafted real feature vectors, similar to those depicted in Figure 2.11, to represent the 21 oriented lines and intersections of Figure 2.12. The scheme worked quite well in Stickville

and even showed relatively good results with simulated infrared imagery. What was obvious, even at that point, was that the system needed to be able to learn its feature set thereby easing the limitations inherent in any human autocratic design. The theme then became adaptive vector quantization or AVQ.

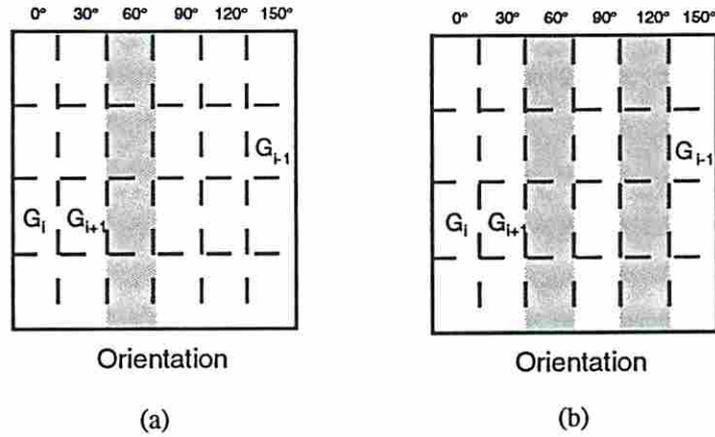


Figure 2.11 - Feature content of a jet. (a) matrix representation of a 60° line, (b) matrix representation of two intersecting lines.

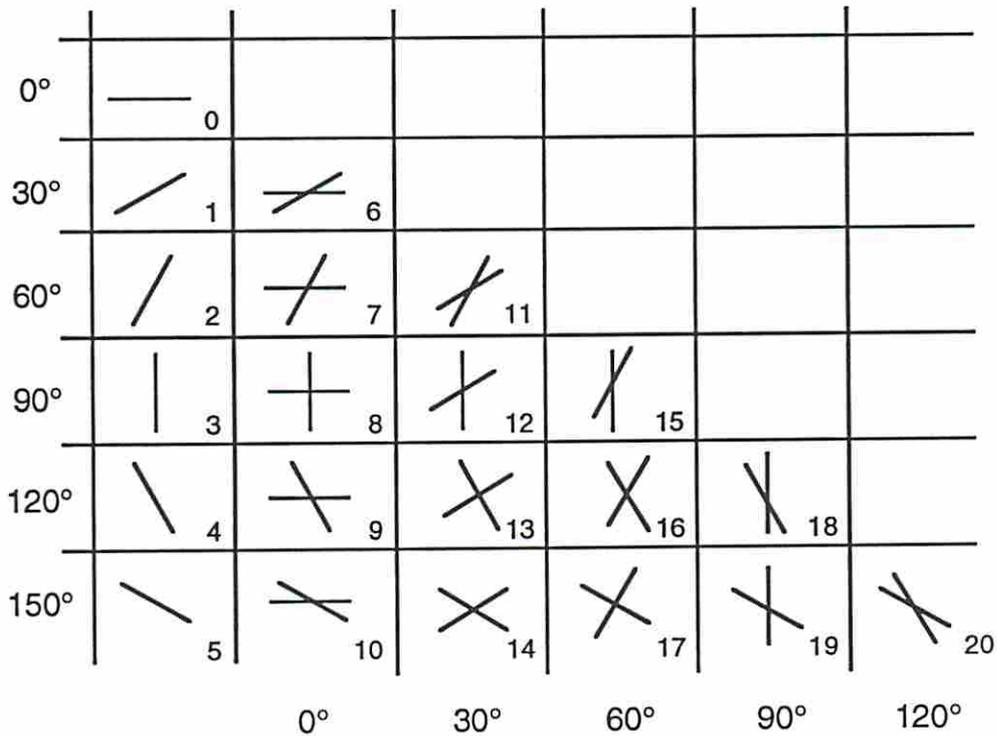


Figure 2.12 - Elemental features extractable by vector quantization of Morlet jets.

AVQ consists of two parts: the learning or adaptive stage in which the system learns to differentiate between different vector classes (features) and the quantization stage in which the system determines to which class an arbitrary feature vector belongs. The methods that we use

to perform AVQ are well discussed in the literature and we have made little modification to those algorithms except where it was appropriate.

## 2.2.1 Learning

Two types of vector learning were explored: self-organizing vector maps [Willshaw and von der Malsburg, 1976] [Kohonen, 1988] and complexity controlled vector quantization [Buhmann and Kühnel, preprint]. The latter, while showing a high degree of promise, was determined to be algorithmically immature and so was rejected in favor of a Malsburg/Kohonen-type of self-organizing vector map.

The idea behind using AVQ is that instead of hand-generating a model vector set as in Figure 2.12, the system learns the different elemental feature types. Input jets from a Morlet block can then be quantized with respect to the learned feature vectors (alternately called model, representative, and codebook vectors) with the resulting feature image being passed to a graph builder and ultimately to a graph matcher for recognition.

Self-organizing vector maps may be trained in two ways: through supervised or unsupervised learning. In both methods, the first step is to determine which model vector is closest to the input vector. In unsupervised learning, that model vector (i.e., the winning model vector) and its neighborhood are updated in the direction of the input vector. In supervised learning, the system is told to which model vector the input vector should be assigned (i.e., the true model vector). The true model vector and its neighborhood are updated in the direction of the input vector. If the winning model vector is different from the true model vector then the winning model vector (to which the input vector had been incorrectly assigned) and its neighborhood are updated in the *opposite* direction of the input vector.

Because we have no prior knowledge as to what the distribution of the input vectors of the model vector map should be (this is, in fact, what we are trying to determine), we have chosen the unsupervised learning method. We begin the training by initializing a 2D vector map to randomly oriented vectors. (Dimensionality of the map is not necessarily limited to two; higher dimensions provide greater resolution and more degrees of freedom in discriminating vectors. The cost is that of space and time.) Each element of the vector map is a  $d$ -dimensional model vector; in our case  $d = S*O = 24$ . A vector (jet)  $\mathbf{J}$  is input and compared to each model vector  $\mathbf{m}_i$  using the metric  $d(\mathbf{x}, \mathbf{m}_i)$  given by:

$$d(\mathbf{J}(t), \mathbf{m}_i(t)) = \|\mathbf{m}_i(t) - \mathbf{J}(t)\|. \quad (2.3)$$

The model vector  $\mathbf{m}_k$ , that is closest in a Euclidean sense to the input vector, and a defined neighborhood of  $\mathbf{m}_k$ ,  $\mathbf{N}_k$ , are updated. All other model vectors are left unchanged. That is,

$$k = \underset{i}{\operatorname{argmin}} \{d(\mathbf{J}(t), \mathbf{m}_i(t))\}, \quad (2.4)$$

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)[\mathbf{J}(t) - \mathbf{m}_i(t)] \quad \forall i \in \mathbf{N}_k, \quad (2.5)$$

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) \quad \forall i \notin \mathbf{N}_k. \quad (2.6)$$

Typically  $\alpha(t)$  is a Gaussian shaped neighborhood function whose peak magnitude decreases linearly over iteration as suggested by [Kohonen, 1988]. That recommendation assumes that all input vectors are of equal importance, which is clearly not the case in our system. We have thus modified  $\alpha$  such that it includes a gain term that is a function of the input jet's saliency. The more salient the training jet, the greater its effect on the update:

$$\alpha(t) = \alpha_k * \|J\| * \left(1 - \frac{t}{t_{\max}}\right) * \exp\left(\frac{-\delta^2}{2\sigma^2}\right) \quad (2.7)$$

where  $\alpha_k$  is the nominal gain,  $\|J\| = [0,1]$  is the scaled input jet magnitude,  $t_{\max}$  is the maximum number of iterations,  $\delta$  is a displacement vector in the map space between  $\mathbf{m}_k$  and  $\mathbf{m}_i$ , and  $\sigma$  is the Gaussian roll-off parameter.

### 2.2.2 Quantization

After learning has been completed, i.e., after a number of sample vectors have been applied to the system and the model vectors have stabilized, the learned model vectors may be used for quantization. Quantization consists of a brute force application of Equation 2.4 for a given input  $J$ . When the minimum  $d(J, \mathbf{m}_k)$  is found,  $J$  is replaced with the value of  $\mathbf{m}_k$ , i.e., it is quantized:

$$V: \mathbb{J} \rightarrow \mathbb{M} \quad (2.8)$$

where  $V$  is the vector quantization function defined by Equation 2.4,  $\mathbb{J}$  is the set of jets, and  $\mathbb{M}$  is the set of model vectors. Each quantized jet is then assigned the label of its quantizing vector (e.g., the value of the index  $k$ ):

$$L: \mathbb{M} \rightarrow \mathbb{L} \quad (2.9)$$

where  $L$  is the labeling function and  $\mathbb{L}$  is the set of labels corresponding to  $\mathbb{M}$ . A compound quantization and labeling function is then defined as:

$$Q: \mathbb{J} \rightarrow \mathbb{M} \rightarrow \mathbb{L}. \quad (2.10)$$

The 2D array of labels resulting from the application of  $Q$  are passed as a *feature image* to the next stage of processing -- graph generation.

### 2.2.3 Algorithm Implementation

Being the two stage process that it is (learning and quantization), the AVQ function is implemented as two separate programs. One program is responsible for the self-organization of the vector map (learning) and the second program is responsible for quantization and the generation of feature images. Appendix B shows the general algorithm flow and how the AVQ implementations fit in.

### 2.2.3.1 Learning

The program that was written to perform the training of the model vector map is called *AVQ* for Adaptive Vector Quantization. The general program flow for AVQ learning is shown in Appendix D. Macintosh dialog boxes (and thus processing options and parameters) for AVQ learning are given in Appendix H.

### 2.2.3.2 Quantization

The code that was written to perform the jet quantization is incorporated in program *GTVQ*. The general program flow for *GTVQ* and related processing is shown in Appendix C. Because the vector quantization algorithm is so trivial and is adequately described above, there are no detailed flowcharts explicitly for that function. Macintosh dialog boxes (and thus processing options and parameters) for *GTVQ*, quantization related processing, and other *GTVQ* related processing are given in Appendix G.

## Chapter 3 - Graph Formation

Graphs are extremely powerful representational structures, especially when used in object recognition. Feature sensitivity notwithstanding, they are inherently invariant to translation and rotation and robust with respect to partial occlusion. By eliminating distance metrics from the edges, we facilitate robustness to scale and other distortional variations. The challenge in using graphs in visual recognition is in forming the graphs in a reliable and stable manner, both in node and edge senses. In this chapter, we discuss the methods by which we are able to form robust graphs that are stable over several forms of distortion.

We will use the term object-graph to describe a graph that, by itself, represents an object. An input graph is an object-graph that serves as an input to the graph matcher. The term model-subgraph is also used to describe a graph that represents a single object but is a subgraph of a larger, multi-object model graph. The multi-object model graph forms a database against which input graphs are compared by the graph matcher<sup>5</sup>. The manner in which input and model graphs are generated is the same with the obviously necessary difference being the quantity of represented objects contained by the graphs<sup>6</sup>.

This chapter discusses the graphs and their formation in terms of nodes and edges. It should be noted that, relative to our system, graphs are only convenient representational structures for what will later be interpreted as neural network topologies with the nodes becoming active neurons and the edges becoming constraints on dynamically varying synaptic links. The above caveat is intended to increase the reader's insight into the nature of our system and not to imply a reduced scope of generality. Indeed, the use of graph theoretic description is also intended to make our work more available to those who find only specific portions of the system useful, either as a novel form of feature-extraction/object-representation or as an interesting way of doing graph-matching/object-recognition.

### 3.1 Node Generation

The first step in the graph formation process is the selection of appropriate features representing local multi-resolution edge data. The features are generated in the two preceding processing modules and are the indices of model vectors to which Morlet jets have been

---

<sup>5</sup> Use of the word *model* indicates that the graph is a stored representation.

<sup>6</sup> As generated by this module, model graphs are disjunct combinations of object graphs. In the subsequent graph matching module, the model graph may be reconfigured such that nodes and edges are shared by more than one pattern. This concept concerning mixing ratio will be discussed in Chapter 4.

quantized. Once chosen, the selected features become the labels of nodes that comprise an object-graph.

Selecting the appropriate features is crucial to the recognition task. The features chosen must be salient, reliable, stable, and accurate. By selecting features with such attributes, we are able to reduce the burden imposed on the recognition process.

A salient feature is one that possesses a sufficiently high degree of perceptual information and may be used to minimize the number of nodes required to both adequately describe and differentiate an object. For that purpose, we use the saliency measure discussed in Chapter 2, i.e., jet magnitudes. Recall that large jet magnitudes indicate the existence of discontinuities in the input image and that discontinuities, from both information theoretic and perceptual standpoints, have high information content. Saliency is the primary selection criterion for features.

Reliable features are those that are likely to be repeatedly selected despite subtle changes in the input image. We have noted experimentally that large feature patches of homogeneous type are fairly stable over input distortion. Features in the center of a patch tend to remain stable with perturbed input when compared with features at the edges of the patch. One way, then, of determining the reliability of a feature is to note its relationship to neighboring features. In our scheme, if a feature is similar to its neighborhood then it is considered to be reliable. Specifically, feature reliability may be judged by feature continuity which we define as

$$C_{ij} = \frac{1}{|\mathcal{N}_{ij}|} \sum_{\mathbf{x} \in \mathcal{N}_{ij}} \mathbf{J}_{ij} \cdot \mathbf{J}_{\mathbf{x}} \quad (3.1)$$

where  $C_{ij}$  is the feature continuity for the feature at location  $(i,j)$ ,  $\mathbf{J}_{ij}$  is the jet that leads to the feature at  $(i,j)$ , and  $\mathcal{N}_{ij}$  is the set of neighborhood indices about  $(i,j)$ . As can be seen from Equation 3.1, continuity is the average dot product of the reference jet with respect to its neighborhood. Qualitatively, it is the measure of how similar the reference jet is to its surround.

Feature accuracy refers to the degree to which the feature label characterizes the underlying image information. Recall from Chapter 2 that a Morlet wavelet decomposition forms jets that are subsequently quantized to learned model vectors. The quantization process produces a value called quantization error that is a measure of how well a particular jet fits the model vector to which it has been quantized. Specifically,

$$E_{ij} = |\mathbf{m}_k - \mathbf{J}_{ij}| \quad (3.2)$$

where  $E_{ij}$  is the quantization error of the feature at  $(i,j)$ ,  $\mathbf{J}_{ij}$  is the jet that gave rise to the feature at  $(i,j)$ , and  $\mathbf{m}_k$  is the model vector to which  $\mathbf{J}_{ij}$  is quantized (Equation 2.8) and whose label is the feature at  $(i,j)$  (Equation 2.9).

Ultimately we want the selection process to be stable, that is, we want to choose features that do not vary greatly in type over distortion. The first effects of distortion occur in the fine detail of an image and are therefore most detectable in the high spatial frequency components of a jet. In other words, high spatial frequencies are more susceptible to image distortion than

low spatial frequencies. At the same time, high spatial frequency filter responses provide finer spatial accuracy on the position of image features and saliency than do low spatial frequency filter responses. It is for this reason that we generate features using lower bands of spatial frequencies than those we use for the saliency measure. The result is that we have both spatially accurate saliency information and stable features.

In addition to the desired node attributes discussed above, we impose one other constraint on feature selection: minimum spacing. Because the features have the property of extended locality (i.e., the features contain information about the image that is relatively distant from the feature location), neighboring features often carry approximately the same feature information. Requiring a minimum spacing between features helps reduce the redundant coding of features.

The first selection criterion for a node label is that it exists near a point of locally maximum salience. Because the saliency array is formed using the highest possible spatial frequency ( $\lambda=4$  pixels) and because of the discrete sampling function implicit in the digital processing, there is a tendency for the jet magnitudes to have spurious local maxima. Our solution to this has been to median filter the saliency array such that the spurious local maxima (which look like uncorrelated noise) are removed while the correct spatial locations of the true local maxima are retained (i.e., no signal smearing).

We define here, for convenience of explanation, a data structure called a *sampling array*. At this point in the processing, the sampling array consists of the local maxima of the saliency array. In the next processing step, the location of the sample points are adjusted using a combination of feature reliability (continuity) and feature accuracy (quantization error). Specifically, a continuity/accuracy array (or a  $\Gamma$ -array) is computed using

$$\Gamma_{ij} = \rho C_{ij} - (1-\rho)E_{ij} \quad (3.3)$$

where  $E_{ij}$  are defined by Equations 3.1 and 3.2, respectively. Next, the sampling array points are spatially adjusted by relocating them to the largest  $\Gamma$ -value within a specified radius. The desired (and obtained) effect is to select a sampling point in a highly salient region that is also maximally reliable and accurate.

Finally, the features whose indices coincide with active sampling points become labels for the newly created nodes.

## 3.2 Edge Generation

In our graphs, edges are directional, unlabeled, and represent a degree of locality between nodes.<sup>7</sup> Edge generation begins with the arbitrary selection of a destination node. Other nodes are then considered for connection to the destination node beginning with the spatially closest

---

<sup>7</sup> This description of an edge is valid only for this processing module. In the initialization of the subsequent graph matching process, edges take on a refined meaning and valuation that relates to the similarity of feature types.

node<sup>8</sup> and ending when either all nodes have been exhausted or a constraint requires the termination of the process for the current destination node. In addition to the minimum distance constraint imposed by the node selection process, the edge constraints have the following priority:

- 0) Minimum Distance Between Nodes (node selection constraint)
- 1) Minimum Number of Inbound Connections
- 2) Maximum Distance Between Nodes
- 3) Maximum Number of Inbound Connections

Because we wish to be robust with respect to scale, the distance constraints must be made flexible so that different sized views of the same object generate similar graphs. That flexibility is achieved by allowing the distance constraints to vary with the size of the object which is measured by a bounding rectangle. The distance constraint values correspond to a defined "standard size" object. Objects that differ in size from the standard have their distance constraints scaled accordingly. Thus, a larger object will have larger minimum and maximum distances and the nodes will be correspondingly farther apart.

### 3.3 Implementation

The program that we have written to form input and model graphs is called SGMaker for Sparse Graph Maker. The general program flow for SGMaker is shown in Appendix E. Macintosh dialog boxes (and thus processing options and parameters) for SGMaker are given in Appendix I.

---

<sup>8</sup> Although there is no distance metric explicit on the edges of our graphs, graph formation may be considered a transition process that, in part, takes a data set with a distance topology and converts to one with only a connection topology. It is during this transition that attributes from both topologies may be used and it is in this way that we may discuss spatial proximity of features/nodes with respect to a non-distance topological system.

## Chapter 4 - Dynamical Link Graph Matcher

The previous processing stages of our system translate input greyscale imagery into labeled, directed graphs of two types: single-object input graphs and multi-object model graphs. The model graph comprises the database of stored graphs against which the input graph is to be matched. In what follows, the terms *node* and *neuron* will be considered interchangeable. In addition, all references to neurons should be taken in the figurative sense and refer to the artificial variety.

The mechanism used in the graph matching process has its foundation in the Dynamical Link Architecture (DLA) [von der Malsburg, 1981] and is patterned closely after the graph matcher introduced by [von der Malsburg and Bienenstock, 1987] [Bienenstock and von der Malsburg, 1987]. These latter two papers, taken together, will henceforth be referred to as the *Europhysics papers*. Neural networks based upon DLA are significantly different from classical neural networks. In classical neural nets, neurons are connected by links or synapses that remain fixed in value during classification and vary only during learning. In DLA systems, the links vary in strength during recall (or classification) as a function of the temporal correlation of the connected neurons.

Our work with the DyLink<sup>9</sup> graph matcher is concerned primarily with the recognition or matching process. Unlike many neural paradigms, our neural graph matcher does not learn its representations in a Hebbian sense. Its database of stored models is formed by the preceding graph formation module and is effectively imprinted into the associative memory of the graph matcher.

To provide the reader with a general understanding of the system architecture and dynamics, the first section of this chapter presents only a brief overview of the graph matcher with greater detail to be found in the two following sections. The section ends with an explanation of mixing factors and their effects, and the differences between our implementation and that of the originating authors.

### 4.1 Overview

Figure 4.1 provides a schematic representation of the DyLink graph matcher. Note that in what follows, graphs are implemented as neural networks with nodes as neurons and edges as synapses.

---

<sup>9</sup> The terms dynamical link and DyLink are used interchangeably throughout.

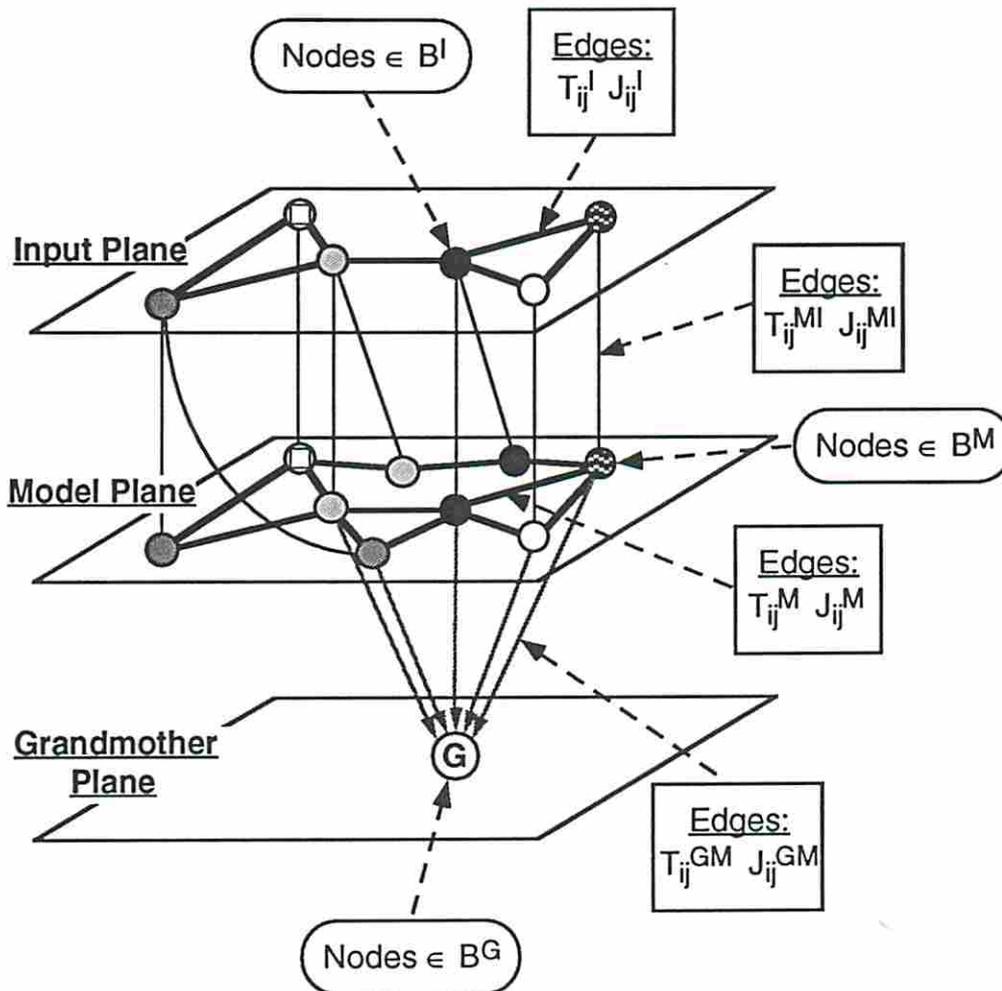


Figure 4.1 - Schematic representation of the DyLink graph matcher.

Architecturally, the graph matcher consists of three neural planes: an *input plane* that holds the object-graph to be matched, a *model plane* that holds the model graph (or database of stored objects), and a *grandmother plane* containing *g-cells* (or grandmother cells) that measure the degree of match achieved by each stored pattern.

Connections within the input plane are fixed by the object that it represents. Connections within the model plane are made in a similar fashion except that the links are dynamic, that is, the links vary in strength during recall as a function of the temporal correlation of the nodes that they connect: the greater the correlation, the stronger the link becomes. Connections from the model plane to the grandmother plane are fixed by the architecture: each *g-cell* is assigned one model-subgraph and receives connections from each of nodes of that subgraph.

Dynamically, the system operates on an iterative-time basis. Each system iteration consists of two phases: an activation and binding phase and a readout phase. See Figure 4.2.

Graph matching begins by loading the input plane with an input graph. See Figure 4.3. Connections are formed between input plane neurons and model plane neurons such that only neurons with similar type are connected. During recognition, a connected ensemble of nodes

in the input plane is activated which, by way of connections from the input plane to the model plane, activates nodes of similar feature types in the model plane. After a short time, activation of the input plane nodes is changed to a neighboring and overlapping ensemble (i.e. attention is shifted) and the activation dynamics are repeated: the input-to-model-plane connections transmit this new set of activations to the model plane and a new set of model nodes is stimulated.

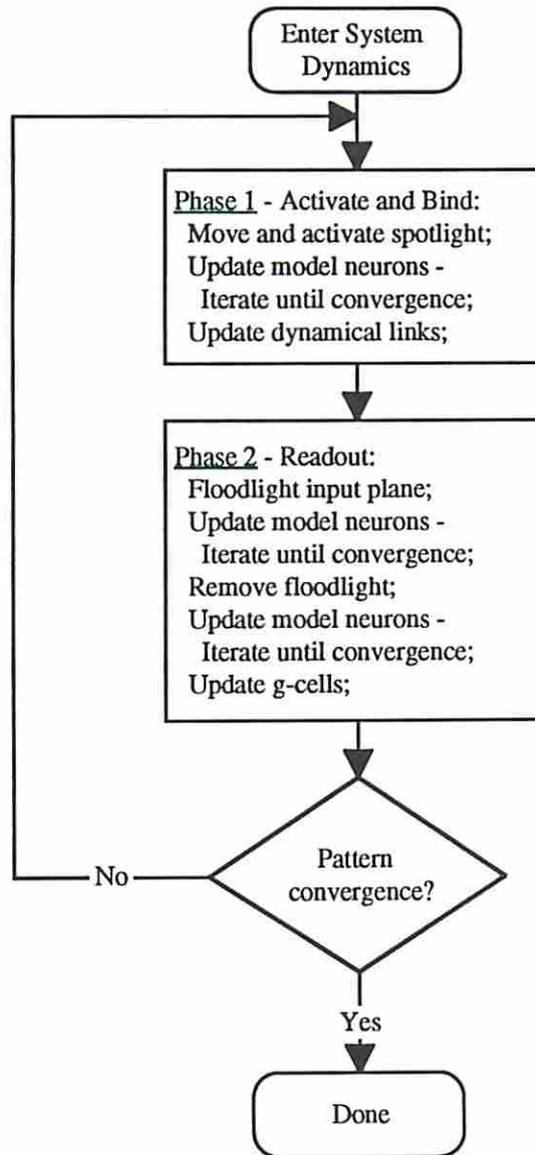


Figure 4.2 - Main system dynamics. One execution of the loop constitutes a single system cycle or iteration.

The shifting of excitation of nodes in the input plane may be thought of as a scanning spotlight [Triesman and Gelade, 1980] [Crick, 1984] passing over that plane: the spotlight activates the ensemble which it lights. The neuron at the "center" of the spotlight is termed the *hot neuron* while the other neurons in the spotlight are called *warm neurons*.

The fundamental idea is that simultaneity of activation in the model plane corresponds to locality of activation in the input plane. In other words, if two neurons are simultaneously active in the model plane, then there must be, correspondingly, two neighboring neurons in the input plane with similar feature types. That is, in fact, the essence of how graph disambiguation in the model plane occurs.

The above described dynamics specify how a model subgraph may be activated. What is needed is a way of stabilizing those activations; this is the function of the dynamical links. As ensembles of model plane nodes become active, their correlated firings cause their connecting dynamical links to become stronger thereby binding those ensembles together more tightly. Nodes that are not a part of these ensembles become disconnected by reduced dynamical link strengths and disassociate themselves with the congealing model-subgraphs.

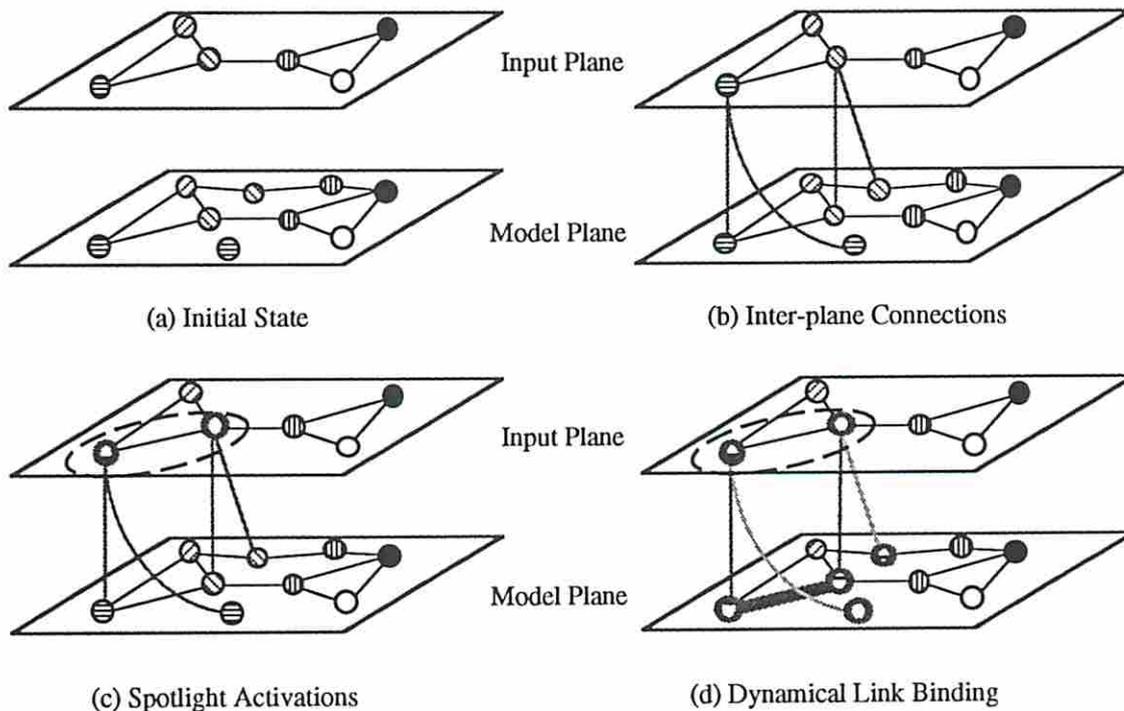


Figure 4.3 - Overview of the matching process. (a) The initial state for recall is one in which the connections within the model plane form an associative memory of stored subgraphs and in which the input plane is a locally-connected graph. (b) Inter-plane connections are made between similarly labeled nodes. (c) Recall begins with the activation of an ensemble of input plane nodes (bold circles). (d) Active input nodes activate similarly labeled model nodes. Network dynamics modify dynamical connections, or J-links (bold line), to bind temporally correlated model nodes. (For ease of depiction, some inter-plane connections and the entire g-cell plane are not shown.)

At the end of each cycle of the system recall dynamics, the dynamical link strengths of the model plane are "read out" to the respective g-cells. The g-cells respond with activity levels that are commensurate with the magnitude of the dynamical links of their specific model-subgraphs, that is, they measure the degree of match between their assigned patterns and the input graph. Pattern convergence is said to have occurred when the output of the strongest g-cell is stable for some specified number of iterations. Chapter 6 discusses the effects of gallery size and graph ambiguity on the convergence rate.

## 4.2 Architecture

Figure 4.1 presents a schematic description of the dynamical link graph matcher.

### 4.2.1 Neural Planes

#### 4.2.1.1 Input and Model Plane Neurons

Input and model plane nodes are labeled by features as determined by preceding processing stages. While the graph matcher itself is not restricted to any domain of features, the feature set that we use are derived from our adaptive feature extractor (discussed in Chapter 2). Those features represent line and edge data of varying orientation and spatial frequency. Each neuron, therefore, while identical in function, is abstractly assigned a label whose value directly influences only the connection topology.

More formally,  $B^I$  and  $B^M$  are the set of nodes (or neurons) in the input and model planes, respectively. Additionally,

$$B^x = \{ \mathcal{l} = (i, \alpha) \mid i = 1..N^x_\alpha, \alpha = 1..F \} \quad x \in \{I, M\} \quad (4.1)$$

where  $\mathcal{l}$  is a neuron,  $\alpha$  is an index on the label or feature type,  $i$  is an index on the neurons of feature index  $\alpha$ ,  $F$  is the number of feature types, and  $N^x_\alpha$  is the number of neurons of feature index  $\alpha$  in plane  $x$ .

Recalling that the model graph contains multiple object-subgraphs, we define the set of nodes of a model-subgraph to be

$$\mathbb{L}_\pi = \{ \mathcal{l} \mid \mathcal{l} \rightarrow \mathbb{P}_\pi, \mathcal{l} \in B^M \} \quad (4.2)$$

in which  $\pi$  is the index on the set of patterns  $\mathbb{P}$ . In other words,  $\mathbb{L}_\pi \subset B^M$  is the set of nodes belonging to pattern  $\pi$ .

We define  $\mathbb{F}$  as the set of features:

$$\mathbb{F} = \{ f(\alpha) \mid \alpha = 1..F \} \quad (4.3)$$

where  $f(\alpha)$  may be a specific color, vertex type, line direction, etc. For future notational convenience, we also define a function  $\mathfrak{F}$  that operates on a node and returns the label of that node:

$$\mathfrak{F}(\mathcal{l}) = f(\alpha) \in \mathbb{F}. \quad (4.4)$$

The nodes act as binary neurons that fire or are silent according to the strength of their inputs:

$$\sigma_i = \zeta_x \left( \sum_j J_{ij} \sigma_j - \Theta_i \right) \quad x \in \{I, M\} \quad (4.5)$$

where  $\sigma_i$  is the output of neuron  $i$ ,  $J_{ij}$  are fast modulated synaptic weights (as discussed below),  $\Theta_i$  is the threshold value for neuron  $i$ , and

$$\zeta_x(\varphi) = \begin{cases} 1 & \text{if } \varphi \geq 0, \\ 0 & \text{otherwise;} \end{cases} \quad x \in \{I, M\}. \quad (4.6)$$

Although our experiments employed primarily binary neurons, the use of graded neurons should be just as effective.

#### 4.2.1.2 Grandmother Plane Neurons

$B^G$  is the set of nodes (or neurons) in the  $g$ -cell plane. Each  $g$ -cell is connected to a model-subgraph and corresponds to that model.  $G$ -cells, then, are essentially readout neurons whose activity level corresponds to the amount of match incurred by the respective pattern. Specifically,

$$B^G = \{ \mathcal{G} = \pi \mid \pi = 1.. \Pi \} \quad (4.7)$$

where  $\mathcal{G}$  is a neuron indexed by pattern number  $\pi$  and where  $\Pi$  is the number of patterns contained by the model graph. Unlike the neurons of  $B^I$  and  $B^M$ ,  $B^G$  neurons are graded. That is, Equation 4.6 for  $g$ -cells is

$$\zeta_x(\varphi) = \frac{1}{1 + e^{-\gamma\varphi}} \quad x=G \quad (4.8)$$

where  $\gamma$  is the sigmoidal gain.

#### 4.2.2 Links

In artificial neural networks, neurons are connected to each other by links or weighted synapses. It is the nature of the links in a DLA system that separates it taxonomically from other artificial neural paradigms. In this section, we describe the links in an architectural sense; a later section will deal with the dynamics. We begin by describing the basic links much as they are given in the Europhysics papers. The second subsection deals with modifications that we have made to the basic link constructs and is the description of the actual links in our system.

### 4.2.2.1 Basic Link Constructs

Two types of "connection" matrices are used. The first type is contained in the J matrix, the elements of which are called *J-links*. J-links are signal carrying connections of variable synaptic weight and are the so-called *Dynamical Links*. More specifically, they are fast modulated connections that vary with the temporal signal correlations and are the secondary dynamic variables during recognition (with neural activations being the primary dynamic variables). Their purpose is two-fold: 1) to carry information between the connected neurons and 2) to bind or isolate the connected neurons depending upon the correlation of the neural activity.

The second variety of "connection" matrix is contained in the T matrix, the elements of which are called *T-links*. J matrices and T matrices are coherent: for every J-link there is a T-link and vice versa. T-links are, in fact, not true connections at all but provide an upper bound on the variation of the associated J-link, i.e.,

$$0 \leq J_{ij} \leq T_{ij} . \quad (4.9)$$

The  $T_{ij}$  connections vary with learning (over a long time scale) and remain fixed during recognition<sup>10</sup>. (As noted earlier, our system does not learn its graphs in the usual sense; T matrices are created by the graph formation module of Chapter 3 and are imprinted onto the model plane.) The purpose of the T matrices is to define the network topology.

For ease of description, we refer to the combined J and T matrices as C matrices. There are separate C matrices for each network topology. Specifically, the  $C^I$ ,  $C^M$ , and  $C^G$  matrices contain links for connections only within the input, model, and g-cell planes, respectively. In addition, C matrices define topology between planes. The  $C^{MI}$  matrix contains elements that represent connections that join  $B^I$  with  $B^M$ . Similarly, the  $C^{GM}$  matrix contains elements that represent connections that join  $B^M$  with  $B^G$ . In our formulation, there are no connections from g-cells to model neurons or between the g-cells, themselves. However, advantages may be had if certain of these connections are actually established. These possibilities are discussed in Chapter 6.

The  $T^I$  and  $T^M$  matrices define the topology of the input and model graphs, respectively. Their non-zero elements represent a spatial proximity<sup>11</sup> of the nodes that they connect. Formally,

$$T_{ij}^x = \begin{cases} 1 & \text{if } \mathcal{L}_i \text{ and } \mathcal{L}_j \text{ are spatially proximate,} \\ 0 & \text{otherwise;} \end{cases} \quad x \in \{I, M\}. \quad (4.10)$$

In other words, the existence of a  $T^I$  or  $T^M$  link between two nodes indicates that the features that it connects were spatially close in the original image domain. For the moment, we assume

---

<sup>10</sup> In the Europhysics version of the graph matcher,  $T_{ij} \in \{0,1\}$ . In our modified architecture,  $T_{ij} = [0,1]$ .

<sup>11</sup> See Chapter 3.2, footnote 4.

that there is no pattern overlap in the model graph, i.e., all model-subgraphs are disjunct. Note that Equation 4.10 leads to symmetric  $T^I$  and  $T^M$  matrices.

Connections between  $B^I$  and  $B^M$  are established only between identically labeled nodes. That is,

$$T_{ij}^{MI} = \begin{cases} 1 & \text{if } \mathfrak{F}(i) = \mathfrak{F}(j), \\ 0 & \text{otherwise.} \end{cases} \quad (4.11)$$

The nodes of each model-subgraph are connected to a single g-cell and are normalized to unity:

$$T_{\pi j}^{GM} = \begin{cases} \frac{1}{N_\pi} & \text{if } \mathcal{L}_j \in \mathbb{L}_\pi, \\ 0 & \text{otherwise,} \end{cases} \quad (4.12)$$

where  $N_\pi$  is the number of nodes in  $\mathbb{L}_\pi$ . Nominally, this is a many-to-one mapping: many model nodes to one g-cell. In the event that some model neurons are used by more than one pattern, the mapping becomes many-to-many.

#### 4.2.2.2 Feature-Correlation Based Links

The architecture described above assumes that the features used to label the nodes can be extracted with high reliability and are never confused. Our system, however, makes use of features that are closely and naturally related, i.e. they are highly feature-correlated. The system must therefore allow for confusion between features, such confusion being caused by small variations in the input image. In addition, our architecture uses grandmother cells to readout the level of pattern match. This scheme can be adversely affected by multiple, neighboring instantiations of the same feature type in a given model-subgraph. We term the first problem the *Feature Confusion Problem* and the second the *Repetitive Labeling Problem*. In this subsection, we discuss these problems and our solutions to them.

##### 4.2.2.2.1 Soft Feature Matching

If the feature set was distinct, such that it was unlikely for one feature to be mistaken for another (e.g., a feature set consisting of only black and white pixels), then the carrying out Equation 4.11 for  $T^{MI}$  would be easy and would make sense: neurons with black feature labels in the input plane would be connected to all neurons with black feature labels in the model plane and vice versa for white pixels. However, when the features are of high similarity, as in our case, it is no longer reasonable to allow connections only between identically labeled nodes. For instance, in Figure 4.4, the circled feature in the input object may be very close in feature space to the circled features in the model objects. The architecture, as described by Equation 4.11, however, prohibits these connections.

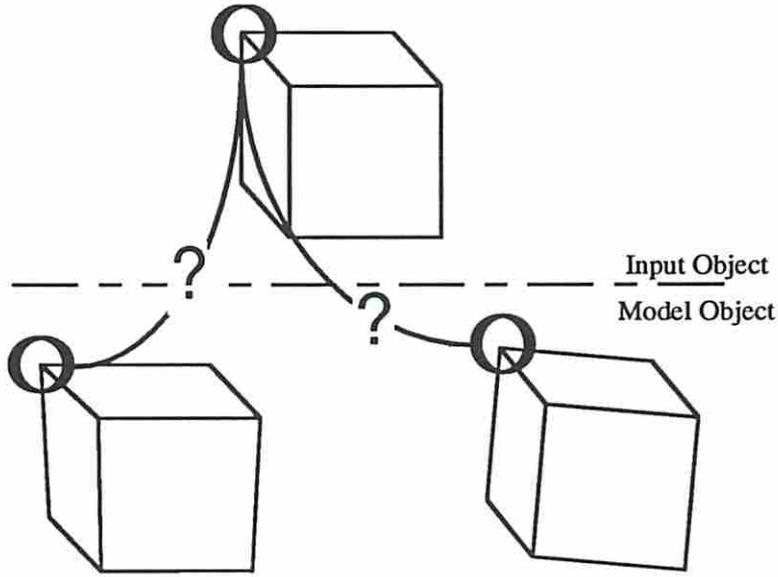


Figure 4.4 - Feature label confusion. The circled features (vertices) are similar but not identical. The original architecture prohibits any connections like those shown labeled by "?". Flexibility in feature matching is improved by providing TMI connections between similar features.

Flexibility is introduced by modifying Equation 4.11 such that nodes of nonidentical, yet similar, feature type may also be connected, albeit with lower strengths indicating the lack of confidence in the correctness of the feature match. Specifically, the feature-correlation based  $T^{MI}$  connection rule is:

$$T_{ij}^{MI} = [C_f \langle \mathcal{F}(i), \mathcal{F}(j) \rangle]^{\sigma_{MI}} \quad (4.13)$$

where  $\sigma_{MI}$  is a scaling constant and  $C_f$  is a feature-correlation function that computes the normalized dot product of the two model vectors represented by the feature labels  $\mathcal{F}(i)$  and  $\mathcal{F}(j)$ . That is,

$$C_f \langle \mathcal{F}(i), \mathcal{F}(j) \rangle = \frac{L^{-1}(\mathcal{F}(i)) \cdot L^{-1}(\mathcal{F}(j))}{|L^{-1}(\mathcal{F}(i))| \cdot |L^{-1}(\mathcal{F}(j))|} \quad (4.14)$$

where  $L^{-1}$  is the inverse of the labeling function defined by Equation 2.9 and returns a model vector when given the corresponding feature label. Because  $C_f \langle \mathcal{F}(i), \mathcal{F}(j) \rangle$  lies between zero and one and typically  $\sigma_{MI} \gg 10$ ,  $T^{MI}$  elements connecting similarly labeled nodes have substantial values whereas  $T^{MI}$  elements connecting dissimilar nodes have zero or inconsequential values. The result is that the image extraction system need not be perfect and that, if necessary, the graph matching architecture can accommodate small discrepancies in feature definition.

#### 4.2.2.2 Repetitive Labeling

Upon further consideration of our architecture it may be observed that there can be many instances in which nodes of identical (or highly similar) feature types are neighbors within a single-object graph. The scenario for this Repetitive Labeling Problem is graphically illustrated in Figure 4.5. When this situation occurs, it is possible (or even likely) that a single activation of an input graph node will lead to simultaneous activations of the neighboring connected model nodes through  $J^{MI}$  connections. (For more detail on the dynamics, see Section 4.3.) Such activation causes the  $J^M$  link connecting the two neighboring model nodes to be increased in value. The erroneously strengthened  $J^M$  links incorrectly stabilize model-subgraphs and extend their activity to dissimilar neighboring nodes. See Figure 4.6. In other words, small subgraphs of identically labeled nodes can become activation self-sustaining and their spatially extended presence can cause other nodes to become erroneously active through  $J^M$  links.

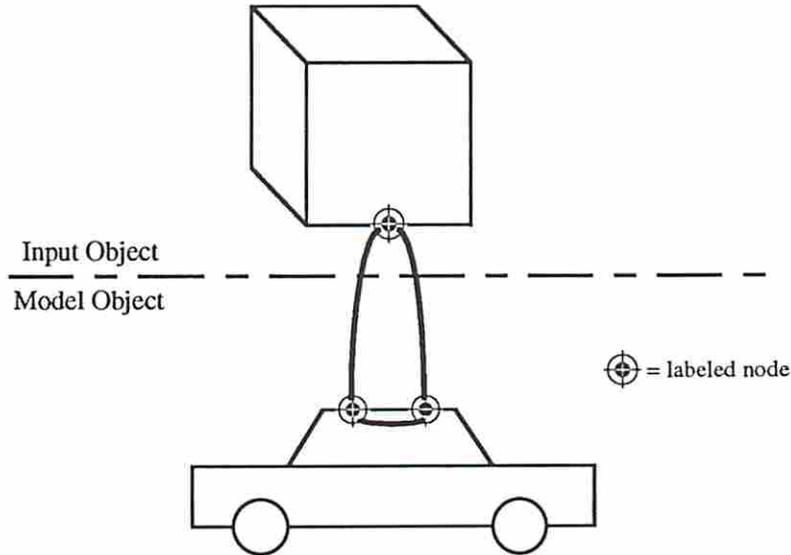


Figure 4.5 - Repetitive Labeling Problem scenario. Neighboring  $B^M$  neurons may be of identical or highly similar feature types. In this figure, the feature type is a horizontal line. Activation of a single horizontal-line  $B^I$  neuron will lead to activation of two neighboring horizontal-line  $B^M$  neurons. The result is an inappropriate strengthening of the  $J^M$ -link.

In addition, during subsequent link readouts, the simultaneous activations of a large number of nodes in a highly feature-correlated cluster contribute to an inordinately high level of stimulus to the pattern's g-cell as may be inferred from Figure 4.6. The result is that the match value for the offending model-subgraph is inordinately high thereby leading to false indications regarding match results. To summarize, the Repetitive Labeling Problem manifests itself in three ways in our architecture: incorrect stabilization of model-subgraphs, inappropriate extension of activation to other neurons, and an inordinately high contribution to g-cells.

Though the Repetitive Labeling Problem existed in the work described in the Europhysics papers, it was not explicitly identified by the authors, perhaps partially because the symptom in which the problem manifested itself was caused by the confluence of several architectural

assumptions. The aforementioned symptom was an extremely slow convergence rate of the original DyLink graph matcher. Because our graph matcher uses a different architecture and incorporates different dynamics, we cannot count on a brute force application of time to solve our Repetitive Labeling Problem.

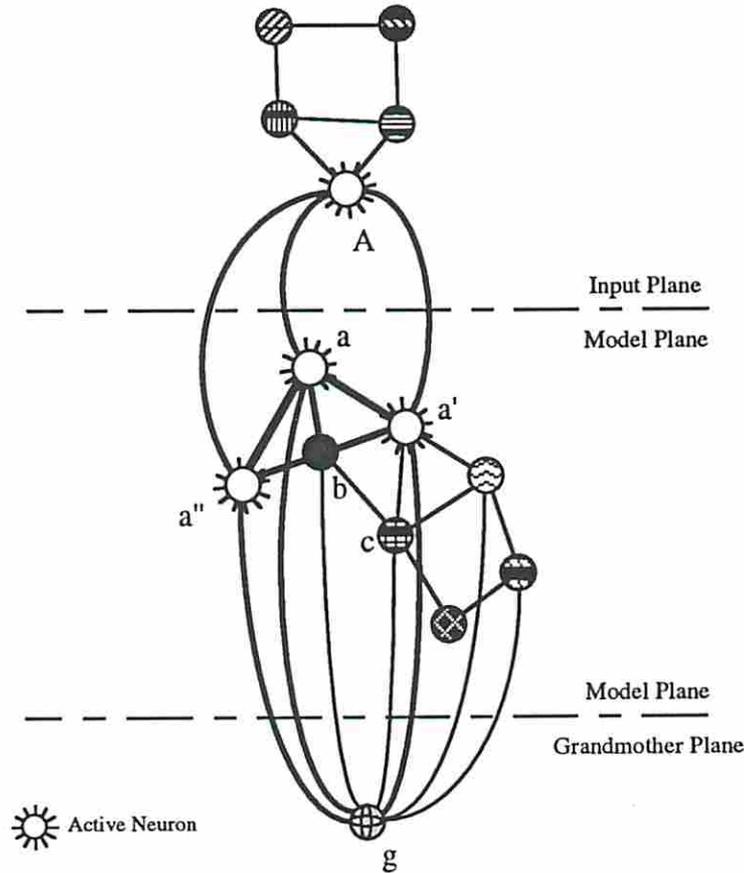


Figure 4.6 - Repetitive Labeling Problem. The activation of  $B^I$  neuron A leads to simultaneous activations of  $B^M$  neurons a, a', and a". These  $B^M$  activations, in turn, lead to the incorrect activation of  $B^M$  neuron b and perhaps later, neuron c.  $J^M$  links a-a', a-a" become strengthened. In addition,  $J^M$  links a-b, a'-b, and a"-b, become erroneously strengthened. Finally, activation of a, a', and a" lead to inordinately strong activation of g-cell g.

We have, instead, used an approach similar to that employed as a solution to the Feature Confusion Problem: feature-correlation based links. The symptoms of the Repetitive Labeling Problem are approached as two separate sub-problems: 1) incorrect stabilization of model-subgraphs and extended activation to inappropriate neurons, and 2) inordinately high contribution to g-cells.

#### 4.2.2.2.1 Solution to Incorrect Model Sub-Graph Stabilization and Activity Extension

The first two symptoms of the Repetitive Labeling Problem (incorrect stabilization of model-subgraphs and extended activation to inappropriate neurons) may be solved together by fixing  $T^M$  connections such that they are a function of the average feature-correlation of a local

subgraph. The idea is to reduce the effect that a large number of nodes in a highly feature-correlated subgraph has on other nodes connected to that subgraph. In general, the more highly feature-correlated a neighboring subgraph is, the lower the values of the  $T^M$  links from that subgraph should be. To be more accurate,

$$T_{ij}^M = 1 - [\bar{C}_f \langle \mathcal{F}(i), \mathcal{F}(j) \rangle]^{\sigma_M} \quad i, j \in B^M \quad (4.15)$$

where  $\sigma_M$  is a scaling constant and  $\bar{C}_f$  is a function that computes the average feature-correlation of a neighboring subgraph using Equation 4.14. Specifically,

$$\bar{C}_f \langle \mathcal{F}(i), \mathcal{F}(j) \rangle = \frac{1}{N_k} \sum_{k \in B^M} C_f \langle \mathcal{F}(j), \mathcal{F}(k) \rangle \quad (4.16)$$

where  $j$  is any node connected to node  $i$ ,  $k$  is a node connected to both nodes  $i$  and  $j$ , and  $N_k$  is the number of nodes in the neighboring subgraph<sup>12</sup>. Because the value of  $C_f \langle \mathcal{F}(i), \mathcal{F}(j) \rangle$  lies between zero and one and typically  $\sigma_M > 1$ ,  $T^M$  links connecting heterogeneous clusters of nodes to nodes exterior of the cluster have near unity values. Conversely,  $T^M$  elements connecting homogeneous clusters to nodes external to those clusters have low values. The result is that repetitively labeled node clusters do not unduly affect neighboring nodes through a single  $B^I$  node activation because their connections to those neighboring nodes are attenuated.

#### 4.2.2.2.2.2 Solution to Inordinately High Contribution to G-Cell Activity

The last symptom of the Repetitive Labeling Problem, inordinately high contribution to g-cells, is solved by fixing  $T^{GM}$  connections such that the contribution of a  $B^M$  node to a specific g-cell is a function of its and its neighbors' feature types. The more similar the feature of a  $B^M$  node is to those of its neighbors, the weaker its  $T^{GM}$  link and vice versa. Specifically,

$$T_{\pi j}^{GM} = 1 - [\bar{C}_f \langle \mathcal{F}(\pi), \mathcal{F}(j) \rangle]^{\sigma_{GM}} \quad \pi \in B^G, j \in B^M \quad (4.17)$$

where  $\sigma_{GM}$  is a scaling constant and  $\bar{C}_f$  is the same average feature-correlation function defined by Equation 4.16. Because  $C_f \langle \mathcal{F}(\pi), \mathcal{F}(j) \rangle$  lies between zero and one and typically  $\sigma_{GM} > 1$ ,  $T^{GM}$  links to g-cells from highly feature-correlated  $B^M$  clusters have near zero values whereas  $T^{GM}$  links connecting poorly feature-correlated  $B^M$  clusters to g-cells have near unity values. The effect, then, is to make stimuli coming from individual nodes of high feature-correlation subgraphs less important to the degree-of-match measurement and to make information coming from nodes with unique feature types (with respect to their neighborhood) more important to that measurement.

---

<sup>12</sup> The neighboring subgraph is considered to consist of neuron  $j$  and of all other neurons connected to neurons  $i$  and  $j$ .

## 4.3 Dynamics

The execution of the graph matcher, or its dynamics, may be described at three levels: neural, link, and system. Neural dynamics deal with activity at the lowest and most active level and give rise to dynamical link activity. Link dynamics occur at an intermediate level and are responsible for binding neurons together and stabilizing sub-networks (i.e., model-subgraphs). System dynamics entail the repeated application of neural and link dynamics in defined phases of activity and the shifting of the internal focus of attention.

### 4.3.1 Neural Dynamics

In Section 4.2 we briefly described our neurons in terms of synaptic connections and transfer functions. In this section, we expand upon those definitions by exploring the temporal dynamics of those neurons. Because our system uses common artificial neurons and because those neurons are described throughout the literature, our discussion of the neural dynamics is brief.

The graph matcher uses two types of neurons: binary neurons in the input and model planes and graded neurons for the g-cells. The function of both types of neurons is approximately the same: the signal on each dendritic synapse is multiplied by the strength of the synapse, the resulting products are summed together, the sum is reduced by a threshold value, and the result passed through a nonlinear transfer function. See Equation 4.5. In the case of the binary neuron, the transfer function is a unit step (Equation 4.6); for the graded neuron it is a sigmoid (Equation 4.8).

The available temporal dynamics are discrete- and continuous-time. We have used discrete-time dynamics throughout, which is an approximation to the continuous update equation:

$$\dot{\varphi}_i(t) = -\alpha \varphi_i(t) + \sum_j J_{ij} \sigma_j(t) \quad (4.18)$$

where  $\varphi_i(t)$  is the neural potential,

$$\varphi_i(t) = \sum_j J_{ij} \sigma_j(t), \quad (4.19)$$

$\alpha$  is a decay constant, and  $\sigma_j$  is the output of neuron  $j$  as defined by Equation 4.5. (The dynamical links,  $J_{ij}$ , are expressed as being independent of time in the preceding equations. This is not exactly true, as we will discuss in later sections, but the dynamical links vary on such a slow time scale with respect to the neural activity that for our purposes here, they are considered time-independent.)

By the Euler method, the time-iterative solution for Equation 4.18 is:

$$\varphi_i(t+1) = \left( -\alpha \varphi_i(t) + \sum_j J_{ij} \sigma_j(t) \right) \Delta t + \varphi_i(t) \quad (4.20)$$

with  $\Delta t$  as the discrete-time step size.

The special case of Equation 4.20 that we have used and which also gives rise to discrete time dynamics is to simply set  $\Delta t = 1$  and  $\alpha = 1$  so that

$$\varphi_i(t+1) = \sum_j J_{ij} \sigma_j(t). \quad (4.21)$$

We note that the neural dynamics of our system is not limited to the use of the discrete-time formulation, only that we have chosen to use it because of its ease of implementation.

### 4.3.2 Link Dynamics

The primary purpose of links or synapses in any neural system is to carry information between neurons. In systems based on Dynamical Link Architecture, the links serve an additional purpose which is to bind (or disconnect) neural ensembles. That function is accomplished through link dynamics: a stronger link (i.e., greater synaptic weight) binds neurons together whereas a weaker link separates them.

We use the term "binding" in an activation sense. If one neuron is easily excited by a second neuron, we say that the first neuron is tightly bound to the second. If the activation of any neuron in a cluster activates all the neurons in the cluster, we say that the cluster is tightly bound. We also limit the specificity of the term "binding" to the steady-state type of neural dynamics described above and do not include, in this discussion, the possibility of phase binding of oscillatory neural systems.

Recall that our architecture uses the input and model planes of the Europhysics papers and that the model plane consists of a large network (or graph) containing numerous neuron clusters (or subgraphs) where each neural cluster is a topological representation of a stored object. Recall also that the input plane contains a single neural cluster representing an object that the system must try to recognize. The recognition, in general terms, is accomplished by finding the best match between the input neural cluster and the model neural clusters.

The increase of dynamical link strengths within the model plane permits the binding together of neural ensembles that correspond to good matches with input ensembles. Conversely, decreases in dynamical link strengths within the model plane force the decoupling of neural ensembles that correspond to poor matches with input ensembles. In the end, the model-subgraph that has been most tightly and completely bound is considered to be the closest match.

The underlying assumption in DLA is that correlation of activity is significant. If two things are happening simultaneously, then it is probably not accidental and those events are somehow related. The events may, for example, represent different aspects of the same

thought or different features of the same object. In either case, a DLA system attempts to bind those different aspects together by increasing the link strength that connects them. The effect is to stabilize the temporal correlation: future activation of either component results in an increased likelihood of activation of the other bound components. Conversely, events that are temporally uncorrelated are assumed to have no relative relationship and so are disconnected from each other. Dynamical Link Architecture, therefore, is a very powerful mechanism that is a direct solution to the binding problem with applications wherever binding is an issue.

The variable that drives the link dynamics is the temporal correlation of the neural activity. In our scheme, two neurons that are simultaneously active are considered temporally correlated. Conversely, if two neurons are of opposite polarity, they are *anti-correlated*. Intermediately, if two neurons are both inactive, then they are considered *uncorrelated*.

Correlation	Neural Pair Activation	Link Dynamic
Correlated	(On,On)	Increase Strength
Uncorrelated	(Off,Off)	No Change
Anti-correlated	(On,Off), (Off,On)	Decreased Strength

Table 4.1 - General relationship between neural correlation and link dynamics.

The general relationship between temporal correlation and link dynamics for our system is shown in Table 4.1. We define the temporal-correlation function as

$$C_t(i,j) = \begin{cases} 1 & \text{if } i,j \text{ correlated,} \\ 0 & \text{if } i,j \text{ uncorrelated,} \\ -1 & \text{if } i,j \text{ anti-correlated,} \end{cases} \quad (4.22)$$

where  $i$  and  $j$  are neurons.

There are other possible definitions of temporal correlation. The most obvious is the use of longer term measurements which form a more reliable measure of the correlation. Our view, however, is that not only is our method adequate for our application, but it is also faster, requiring only one iteration to determine the degree of correlation. The assumption that allows this to work, of course, is that there is little activation noise.

The remaining task is to define the actual link dynamic. We have experimented with two sets of link dynamics. The link dynamic that is the easiest and perhaps most obvious we call the *Update-Constant Dynamic* and is defined as

$$\dot{J}_{ij} = k C_t(i,j) \quad (4.23)$$

where  $k$  is a constant. In other words, changes in the dynamical link are simply in steps of size  $k$ , the direction of which is dependent upon the temporal-correlation of the neurons.

The beauty of Equation 4.23 is its simplicity and that it performs the basic required function. Unfortunately, those are also the causes of its problems. Usage of the Update-

Constant Dynamic results in system instability partially due to a shifting input plane activation and partially due to the symmetry of Equation 4.23. By symmetry, it is meant that the rate of increase is the same as the rate of decrease which is constant no matter what the value of  $J_{ij}$  or  $T_{ij}$ . The result is that gains made in one direction (binding or disconnection) can be erased as quickly as they are achieved. This is a problem for a system, such as ours, in which input plane activations are externally shifted. We refer to this as the *Spotlight Halo Problem*. The spotlight refers to the active ensemble of input plane neurons; the halo refers to neurons just outside the spotlight. See Figure 4.7. Specifically, only the model plane counterparts of input neurons in the spotlight have their links increased; links with one neuron in the spotlight and one outside it are subject to incorrect but inevitable attenuation. Because nodes are more often peripheral to the spotlight than in it, the dynamical links connecting them are eventually driven to zero.

Another problem associated with the Update-Constant Dynamic is that a J-link with a high T value is at a disadvantage when compared to a J-link with a low T value; more correlations are needed for the former J-link to saturate than the latter. The necessary use of feature-correlation based connections, discussed in Section 4.2, then becomes debilitating. We refer to this as the *Unfair Bias Problem*.

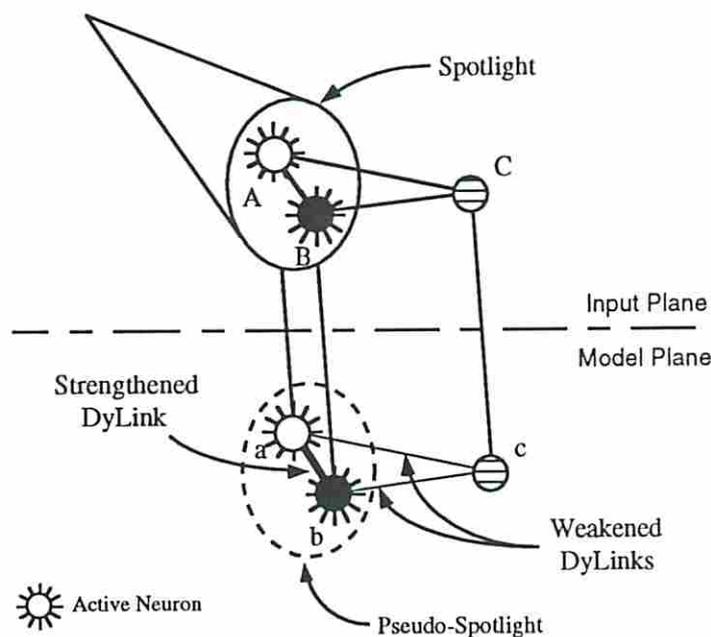


Figure 4.7 - Spotlight Halo Problem. Spotlight activation of input plane nodes inevitably causes some links to be weakened that should not be. Because input plane nodes A and B are active, model plane nodes a and b are also active and the DyLink between them is strengthened. Node c, however, is inactive and the DyLinks between it and nodes a and b are weakened despite the fact that we wish this model-subgraph to be completely bound. If the spotlight were to shift to cover A and C, then DyLinks a-b and c-b are weakened while a-c is increased. If, assuming Equation 4.23, the spotlight is continuously scanned over the input plane, then all the DyLinks of the model graph are eventually reduced to zero.

We have solved both of the problems associated with the Update-Constant Dynamic of Equation 4.23 with what we call the *Rail Capture Dynamic*. Our objective in designing the new dynamic was to produce link stability, to eliminate the Spotlight Halo Problem, and to allow a J-link to grow or recede naturally according to its potential value (thereby solving the Unfair Bias Problem). The update equation for the Rail Capture Dynamic is

$$\dot{J}_{ij} = C_t(i,j) * P(J_{ij}, T_{ij}) * H(C_t(i,j)) \quad (4.24)$$

where  $C_t(i,j)$  is defined by Equation 4.22,  $P(J_{ij}, T_{ij})$  is an inverted parabolic function defined by Equation 4.25 and graphically illustrated in Figure 4.8, and  $H(C_t(i,j))$  is an asymmetric anti-haloing function defined by Equation 4.26 and graphically illustrated in Figure 4.9.

$$P(J_{ij}, T_{ij}) = P \left[ 1 - \left( \frac{2J_{ij}}{T_{ij}} - 1 \right)^2 \right] \quad (4.25)$$

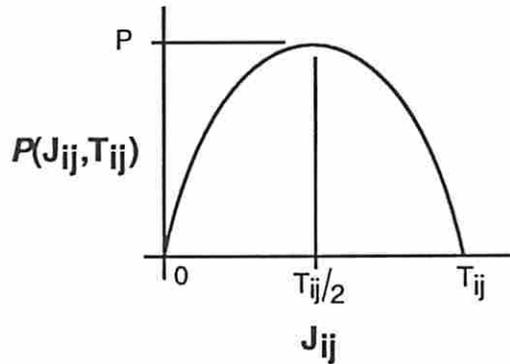


Figure 4.8 - Graphic depiction of Equation 4.25, Inverse Parabolic Function.

$$H(C_t(i,j)) = \begin{cases} d_1 & \text{if } C_t(i,j) > 0, \\ 0 & \text{if } C_t(i,j) = 0, \\ d_2 & \text{if } C_t(i,j) < 0; \end{cases} \quad d_1 \geq d_2. \quad (4.26)$$

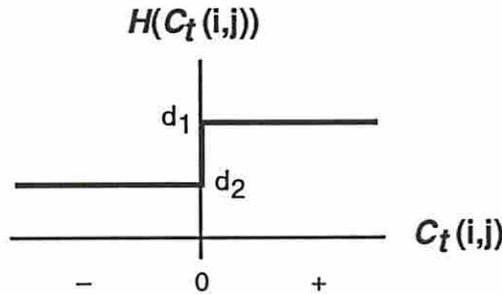


Figure 4.9 - Graphic depiction of Equation 4.26, Asymmetric Anti-Halo Function.

The first term of Equation 4.24 sets the direction of the change in DyLink strength according to the temporal correlation of the nodes that the link connects. The second term provides the rail grabbing characteristic of the equation: a J-link near its mid-range experiences large deviations whereas a J-link near either extreme sees very little change. The effect is that the J-link tends to be driven to a rail and is reluctant to move off it. Empirical study of the

stability of the Rail Capture Dynamic (Chapter 5) shows it to be of high stability and that the rail that captures the link corresponds well to the correlation statistics. In addition, the normalization of  $J_{ij}$  by  $T_{ij}$  allows each DyLink to move in accordance with its potential, thereby solving the Unfair Bias Problem and permitting use of feature-correlation based T-links.

The third term in Equation 4.24 is the Asymmetric Anti-Halo function and is the solution to the Spotlight Halo Problem described above. It works by forcing the increments in J-links to be larger than the decrements. The problem, again, is that a DyLink is more often on the fringe of a pseudo-spotlight (see Figure 4.7) than in it and is therefore subject to more iterations of weakening than to strengthening. The asymmetry of Equation 4.26 makes up for that imbalance. Typically  $d_1/d_2$  is approximately the average number of links per neuron. Experimental results show that with  $d_1/d_2$  approximately one, the Spotlight Haloing Problem is present and most links tend to disintegrate to zero. Elimination of the Spotlight Haloing Problem is observed as  $d_1/d_2$  approaches the average number of links per neuron.

### 4.3.3 System Dynamics

This section discusses dynamics that occur at the system level of the graph matcher. The first section describes how (and when) lower level dynamics are applied to the architecture. The second section explains how the internal focus of attention is shifted.

#### 4.3.3.1 Bind and Latch

During matching, the results of the disambiguation process are evidenced by the strengths of the DyLinks in the model planes: matched subgraphs "congeal" where the links are strong and the nodes are bound together. In some circumstances, this distributed and indirect effect may be considered useful (e.g., in motor control or some other direct action-oriented perception scheme) but in our case we want explicitly to know the results of the graph matching process. There is a need therefore to "read-out" the links and in a way that is not totally artificial.

We do that by allowing the model plane nodes to drive each other (or latch up), which they will do only if they receive and supply enough stimulus to and from their neighboring nodes. Since the amount of stimulus transferred between nodes is a direct function of the link strength, measuring the sum activation of nodes in a model-subgraph is the most direct method of reading out the link strengths<sup>13</sup>. This is, in fact, the current purpose of the g-cells: to accumulate latched model-subgraph activation in a single neuron.

Latching, however, interferes with the disambiguation process in the sense that latched pairs of neurons are always temporally correlated and so their DyLinks are always kept strong

---

<sup>13</sup> The assumption made here (that "the amount of stimulus transferred between nodes is a direct function of the link strength") is derived from the facts that 1) the neurons are binary, not graded, so there are no shades of activation that need to be distinguished, and 2) the Rail Capture Dynamics tend to force the DyLinks to their extremes, allowing us to ignore their shades of value.

regardless of any new contradicting input. Additionally, the activation of any  $B^M$  nodes peripheral to latched nodes are immediately taken to be correlated to the latched nodes and their connecting links are increased in strength; this occurs regardless of whether both node types are active in the input plane or not. Conversely, inactive  $B^M$  nodes adjoining latched nodes, are take to be anti-correlated and are incorrectly disconnected. One solution is a complex and explicit system of inhibitory synapses and auxiliary neurons that latch specifically for g-cell readout. We have taken that approach but modified it in the direction of simplicity: we do allow model plane latching but only during link readout and do not use explicit inhibitory synapses or auxiliary neurons. The tradeoff, here, was one of simplicity versus biological plausibility and we went with simplicity.

Specifically, we use a bi-phase Bind and Latch program. See Appendix F. In the first phase the spotlight is moved and activated.  $B^I$  neurons transmit their activations to  $B^M$  neurons through the  $J^{MI}$  matrix.  $J^M$  transmissions are blocked;  $B^M$  neurons become active based only on inputs from  $B^I$  neurons. Activity correlations in the model plane are then measured and the dynamical links are updated. Note that because the  $B^M$  neurons are not allowed to interfere with each other (due to the blocked  $J^M$ -links), model plane activity is an accurate reflection of what is occurring in the input plane.

In the second phase, the  $J^M$ -links are unblocked. A *floodlight* replaces the spotlight in the input plane, that is, all neurons in the input plane are activated.  $B^M$  neurons now become active based on both input and model plane activations. The floodlight in the input plane is turned off and model plane activity decays to only mutually-driven  $B^M$  neurons; the model plane is thus latched.  $B^M$  activations are passed to grandmother cells ( $B^G$ ) which quantify pattern activation.

The bi-phase process is repeated until pattern convergence is attained. Pattern convergence is declared when the output from the strongest g-cell remains stable for some number of iterations. Typically, the stability activation window is approximately zero (that is, the g-cell must maintain exactly its value through the entire stability time window) and the stability time window is 20 iterations (where the number of iterations required for convergence is typically less than 100).

#### 4.3.3.2 Attention Spotlight

Recall that the fundamental idea of the DyLink graph matcher is that simultaneity of activation in the model plane corresponds to locality of activation in the input plane. From this precept, it becomes obvious that simultaneous activation of all neurons in the input plane presents a highly ambiguous situation to the model plane. Rather, a so-called "spotlight of activity" is swept across the input plane and neurons that "fall within the spotlight" are activated while those "outside" the spotlight remain inactive<sup>14</sup>. See Figure 4.7. The purpose of the spotlight is to activate a localized ensemble of input plane neurons whose activations are

---

<sup>14</sup> The spotlight terminology is nothing more than a metaphor for an externally activated cluster of input plane neurons and provides an appropriate analogy for the mechanisms involved.

passed, by way of the  $J^{MI}$  connection matrix, to the model plane where graph disambiguation can occur.

The size of the spotlight is given by the number of input plane neurons that it activates. For our purposes, the size of the spotlight is fixed throughout the graph matching process though this need not be the case. We have used a spotlight size of 2 or 3 neurons for most of our experiments.

A key element to spotlight control is to determine how it will move. It is desirable, especially in a system with a neural memory ( $\alpha < 1$ ,  $\Delta t < 1$  in Equation 4.20), for the spotlight to move from one ensemble of neurons to a neighboring and overlapping ensemble. The reason for this requirement comes again from the maxim that simultaneity of activation in the model plane corresponds to locality of activation in the input plane. Were the spotlight to jump from one area of the input graph to a disjunct, non-neighboring area, two unrelated input plane ensembles would be active: the newly lit ensemble and the previously lit, still decaying ensemble. Such a situation would erroneously indicate to the model plane that the two ensembles are neighbors when clearly that is not the case; graph disambiguation would become problematic.

At the same time, it is also desirable that the most salient neurons be given priority in their activation, both in terms of order and dwell. Saliency, here, refers to the relative contribution a node might make to graph disambiguation. For example, if the input graph contained a feature that no stored model subgraph possessed, save the matching model subgraph, then the subgraphs could be disambiguated solely on the basis of that feature and that feature would therefore be considered highly salient. It would be reasonable, from a system standpoint, to make the activation of that input neuron a priority, and perhaps to dwell on it for an extended period.

While we have not attempted, in this work, to develop a saliency measure in the terms just described, we have developed a mechanism that allows the spotlight to scan smoothly from highly salient nodes to less salient nodes. The saliency measure that we use here is the same as is defined in Section 2.1: high frequency jet magnitudes. We emphasize, however, that our scanning mechanism is independent of the actual saliency definition; it only uses saliency values as a guide.

The spotlight dynamics begin with a dwell on the most salient node in the input graph which is designated the *hot neuron* and forms the "center of the spotlight". Included in the spotlight are some number of neurons (i.e., the spotlight size minus one) connected to the hot neuron which are called *warm neurons*. In addition to the requirement that they be connected to the hot neuron, warm neurons are chosen as a result of having the highest available saliencies. The spotlight remains in place for the duration of the dwell, which in our case, has been a single system iteration (during which time activation dynamics, link dynamics, and system dynamics are performed). At the beginning of the next system iteration, the saliency for all active input plane neurons is reduced and the saliency for all inactive input plane neurons is increased. The reduction in saliency of active input plane neurons is justified with the claim that, having been stared at for some period, those neurons are no longer as interesting as they were. Likewise, the inactive neurons are increased in salient value because, in relative terms, they are now more interesting. The decrease and increase in saliency of active and

inactive neurons enforces a round-robin type of schedule in which no neuron starves for attention.

The hot neuron remains hot until all of its neighbors have been activated (i.e., have been made warm neurons). When each of the neighbors of the hot neuron has been activated, a *destination neuron* is selected which corresponds to the new, most salient neuron in the input graph. However, instead of immediately making the destination neuron the new hot neuron and jumping directly to it, the shortest and most salient path between the current hot neuron and the destination neuron is found (using a pruned tree search). The first neuron along this path is designated as the new hot neuron and the spotlight covers it and its selected warm neuron neighbors for one dwell period (i.e., a system iteration). Since the current hot neuron is not the highest-saliency destination neuron, the dwell is limited to only one iteration after which the shortest, most salient path to the destination neuron is recomputed and the process resumes. In this manner, the spotlight scans along salient paths from one highest-saliency node to the next, stopping for long dwells at each destination. This behavior was intended to roughly mimic anecdotal information regarding the action of eye saccades.

Finally, we admit that in its implementation, this is not a neural paradigm. However, we gratefully acknowledge Biology for providing the approach and the example for us to follow.

## 4.4 Other Considerations

### 4.4.1 Mixing Factors

Much of the description in this chapter assumed that each model or pattern stored in the model plane was a disjunct subgraph. This does, of course, lead to optimal matching results due to the minimal amount of ambiguity. In a real system, however, we would be concerned with how much information can be stored in a given set of resources (neurons) and how the denser packing might affect the matching results. In this section we define some terms and concepts that are useful in evaluating that tradeoff.

We define a pattern mixing factor as

$$M_{\pi} = \frac{\sum_{i=1}^{N_{\pi}} (S(i) - 1)}{N_{\pi} \cdot (\Pi - 1)} \quad i \in \mathbb{L}_{\pi} \quad (4.27)$$

where  $N_{\pi}$  is the number of nodes in pattern  $\pi$ ,  $\Pi$  is the number of patterns stored in the model plane, and  $S(i)$  is an inverse assignment function that returns the number of patterns assigned to node  $i$ :

$$S(i) = \sum_{\pi=1}^{\Pi} s_{\pi}(i) \quad i \in B^M \quad (4.28)$$

in which  $s_{\pi}(i)$  is a membership function that returns 1 if node  $i$  is an element of pattern  $\pi$  and 0 otherwise. Formally,

$$s_{\pi}(i) = \begin{cases} 1 & i \in \mathbb{L}_{\pi}, \\ 0 & \text{otherwise;} \end{cases} \quad i \in B^M \quad (4.29)$$

where  $\mathbb{L}_{\pi}$  is defined by Equation 4.2 and is the set of model plane nodes for pattern  $\pi$ .

Values of  $M_{\pi}$  range from 0, for no ambiguity, to 1.0 for total ambiguity, i.e., all nodes shared by all patterns. An  $M_{\pi}$  of zero is achievable only with zero overlap. Note that an  $M_{\pi}$  of 1.0 is not likely to be achieved given that some patterns are likely to have nodes with labels that do not exist in other patterns. That is, it is likely that each pattern will have uniquely labeled nodes so that a completely overlapping assignment is not possible. It then becomes reasonable to refer to absolute and relative mixing ratios where the former is defined by Equation 4.27 above and is computed on a pattern by pattern basis. We informally define a relative mixing factor as the number of shared nodes in ratio to the number of shareable nodes.

#### 4.4.2 Uniqueness

A sufficient degree of similarity exists between the graph matcher described in the Europhysics papers and ours such that we feel the need to itemize the significant differences between the two. We begin by noting that C. von der Malsburg and E. Bienenstock provided the framework around which we have built: the pair for their work on DyLink graph matchers, and von der Malsburg for his development of Dynamical Link Architecture.

The first observable difference between the graph matchers is in the implementation: the Europhysics papers used a Monte Carlo minimization of a compound Hamiltonian whereas we have implemented the algorithm directly as a neural network requiring substantially higher levels of detail. The earlier work used densely labeled graphs with a binary feature set whereas we have used sparsely labeled graphs with a rich feature set.

Architectural differences may be found in our use of feature-correlation based links, a grandmother cell plane, and the use of DyLinks only within the model plane.

Differences in dynamics include our use of Rail Capture Dynamics, a smooth scanning saliency driven spotlight, and link readout with bi-phase iterations.

Finally, Chapter 5 will show that we have demonstrated the viability of our system using greyscale imagery under numerous distortions and have explored the impact of node sharing.

## 4.5 Implementation

The program that we have written to perform the graph matching is called DyLink GM for Dynamical Link Graph Matcher. The general program flow for DyLink GM is shown in Appendix F. Macintosh dialog boxes (and therefore processing options and parameters) for DyLink GM are given in Appendix J.

## Chapter 5 - Results

This chapter presents the data with which we validate Chapters 2 through 4. We begin by presenting results for each of the processing modules. The second section provides results of end-to-end recognition tasks involving a 9-object greyscale gallery. Inputs to the recognition tasks are distorted by scale, perspective, in-plane rotation, lighting direction, and occlusion. Outputs from each module are provided as are overall recognition results. We end the chapter with a brief summary of the results contained herein.

### 5.1 Processing Module Results

This section provides results obtained from the individual processing modules. The data sets shown here are the same as those employed during the early phases of development. They are presented because the use of simple data sets often result in a better understanding of complex processing methods.

#### 5.1.1 Morlet Wavelet Decomposition

The theory for the Morlet wavelet decomposition (MWD) is given in Chapter 2. Flowcharts for the actual code can be found in Appendix A and the Macintosh dialog boxes, with their various parametric options, are in Appendix E.

In this section we use three types of images: Stickville, infrared, and greyscale. Stickville images are used to show the basic effects of the processing; infrared and greyscale examples are provided to show the effects of processing on more complex imagery. In all of the processed images (i.e., the images that are not input images), black is high and white is low. All images are 128 pixels x 128 pixels x 8 bits.

Figure 5.1 shows the result of a complex MWD on a Stickville polygon and is the "standard" MWD that we have employed throughout our work. That standard MWD consists of 30 filters of six orientations (one every 30°) and five spatial frequencies (each separated from the other by one octave). The highest spatial frequency has a wavelength of 4 pixels.  $\sigma$  is set to  $\pi/2$ . Recall from Chapter 2 that the four highest spatial frequencies are used in the computation of the jet magnitude saliency function while the four lower spatial frequencies are used in the construction of the jets.

Figures 5.2 and 5.3 show the results of frequency selection in the generation of jet magnitudes. Recall that jet magnitudes serve as our saliency function and should theoretically peak where complex features occur in the image. This may be observed, to some degree in the jet magnitude images of Figures 5.2 and 5.3 but is more obvious in the 3D waterfall plots. The polygon waterfall plot shows high jet magnitude along straight lines and peaks in magnitude at the line intersections. The face waterfall plot shows the highest jet magnitudes at the contours of the face and at the eyes, nose, and mouth, all of which are considered salient in face recognition. Jet magnitudes, composed of lower spatial frequency jets, show the effects of the lower pass filtering as blurring.

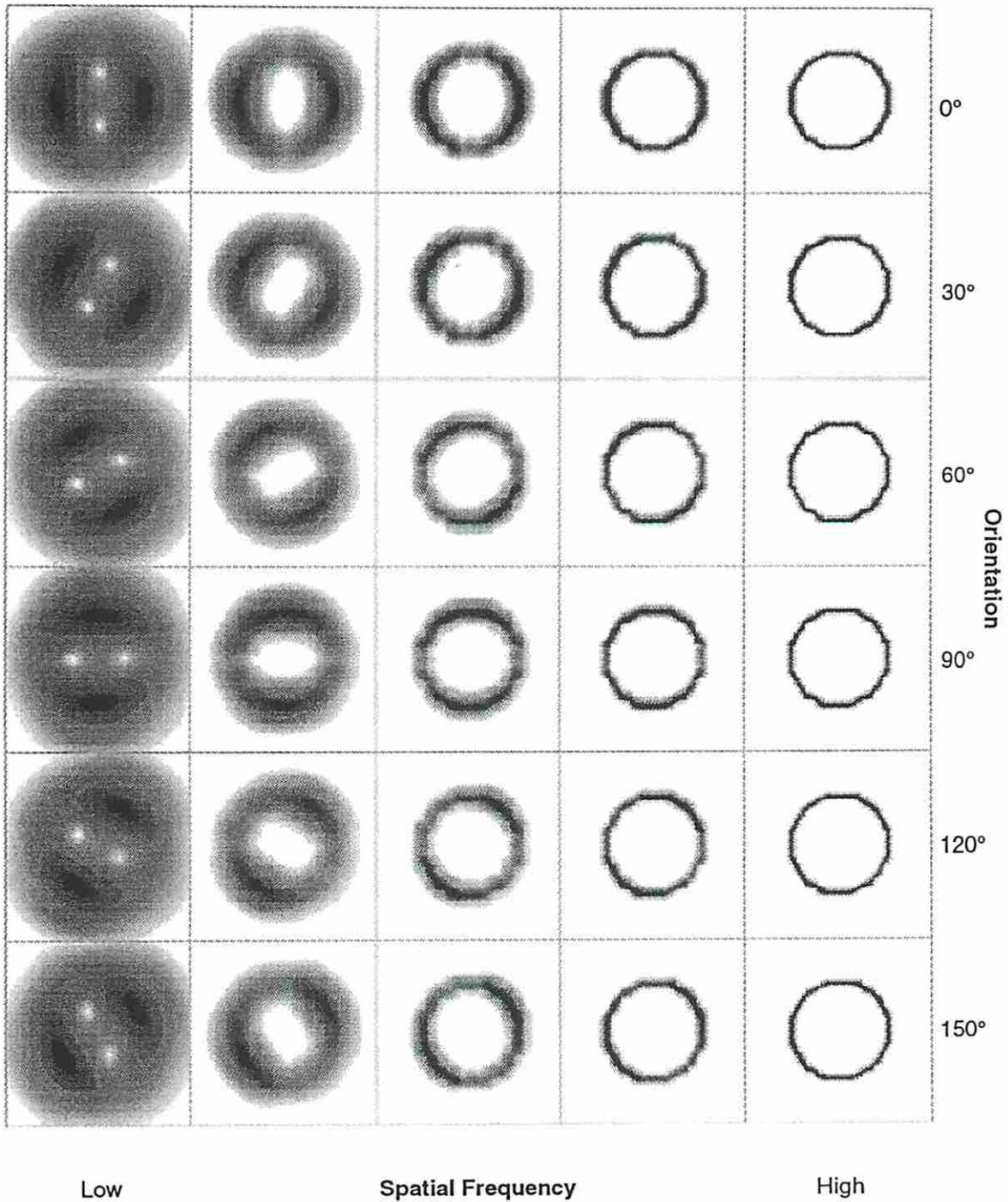


Figure 5.1 - Standard MWD of Stickville polygon: 6 orientations, 5 spatial frequencies,  $\lambda_{\min} = 4$  pixels,  $\sigma = \pi/2$ . Note the effects of increasing spatial frequency and response to filter orientation.

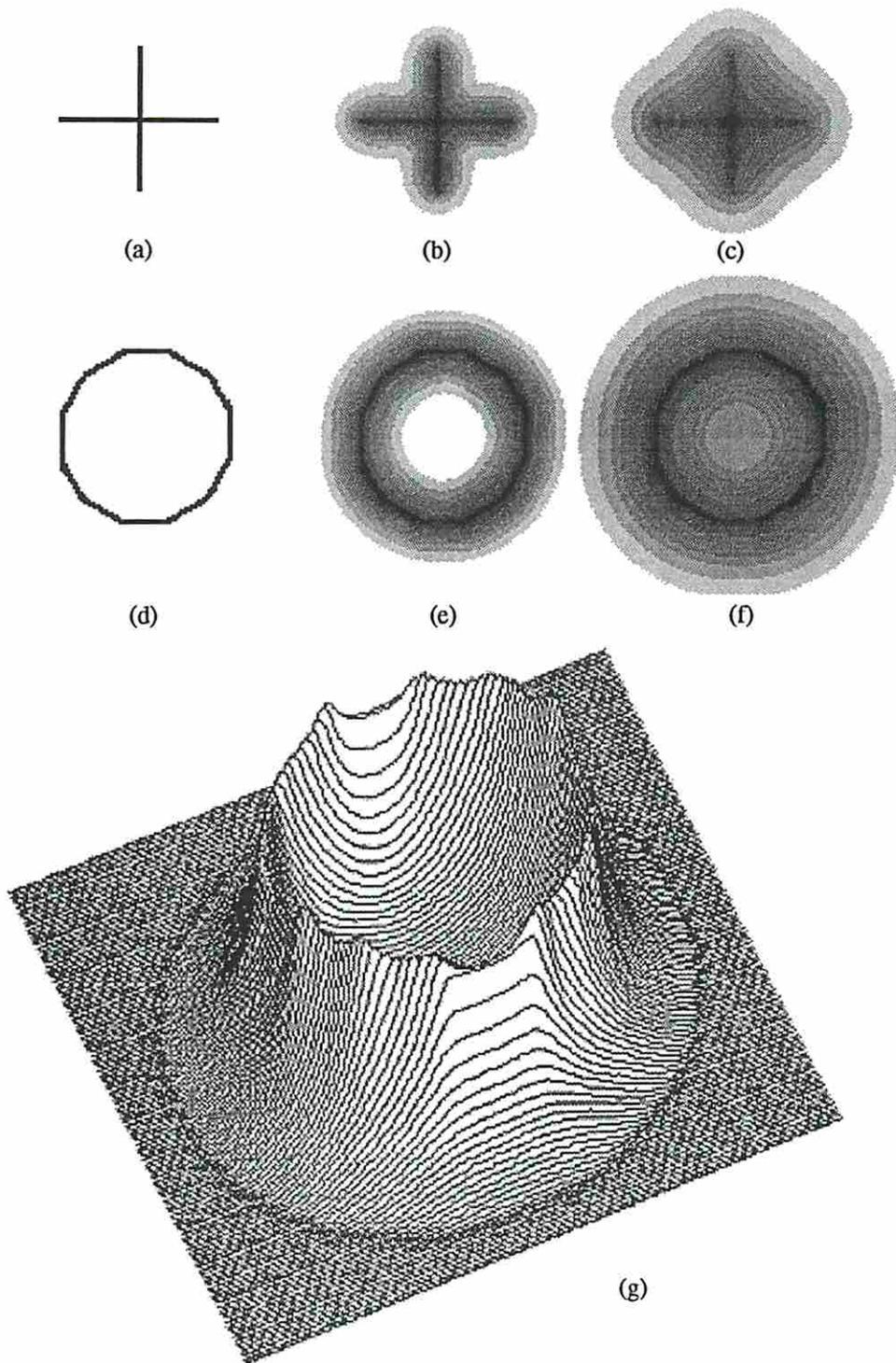


Figure 5.2 - Jet magnitudes (saliency) for Stickville cross and polygon. (a) and (d) are inputs, (b) and (e) are high frequency jet magnitudes, and (c) and (f) are low frequency jet magnitudes. (Black is high, white is low.) Note the loss of resolution in the low frequency magnitudes. (g) shows a 3D plot of (e). Note the strength of the jet magnitudes along the straight lines and the peaks at the line intersections.

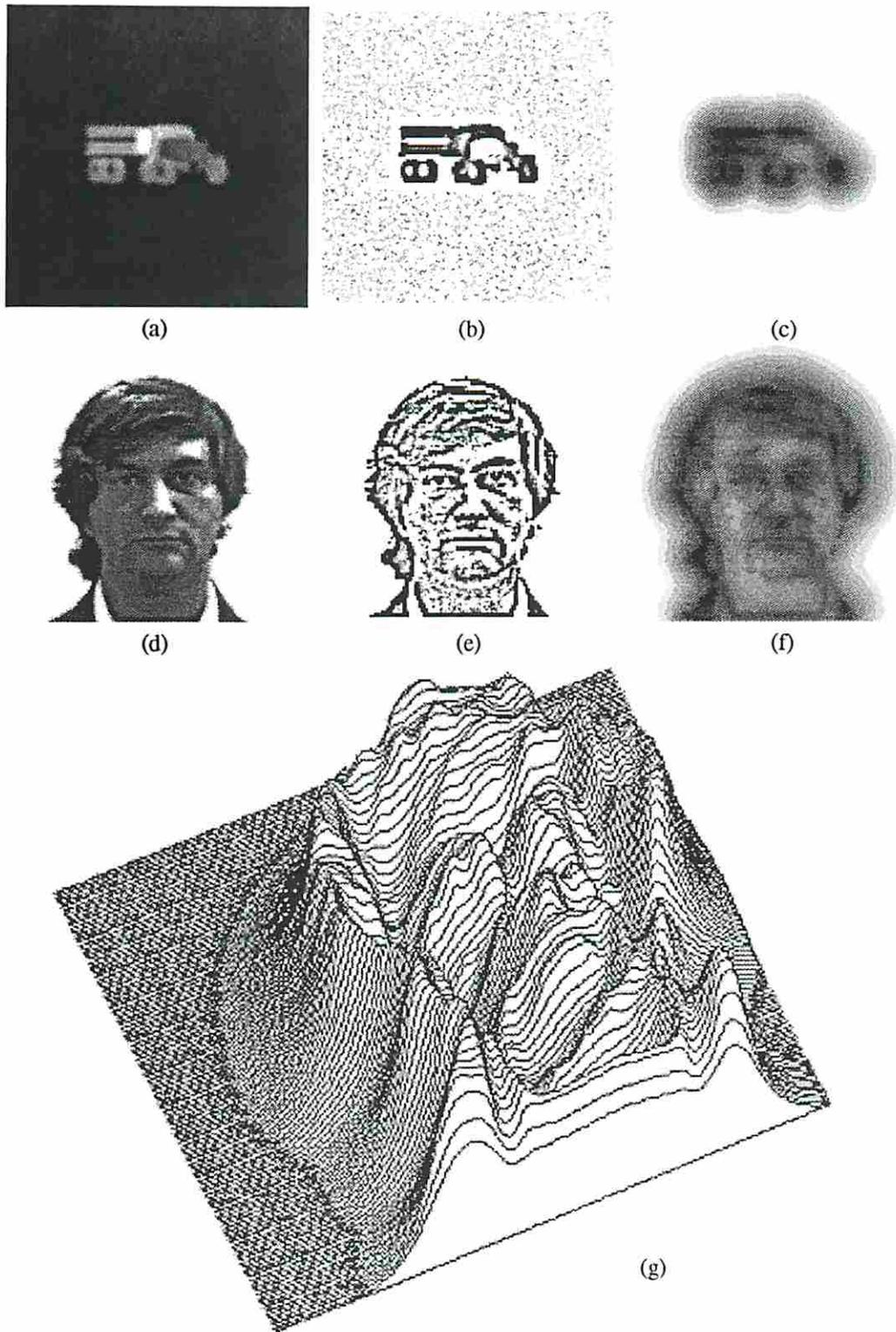


Figure 5.3 - Jet magnitudes (saliency) for infrared truck and greyscale face. (a) and (d) are input images, (b) and (e) are the result of 5x5 Laplacian filtering, and (c) and (f) are the high frequency jet magnitudes. Note that the saliency function corresponds well with areas of high feature importance. (g) shows a 3D plot of (f). The strength of the jet magnitudes are especially high around the contours of the face and at the eyes, nose, and mouth.

Figure 5.4 shows the response of varying  $\sigma$  on the jet magnitudes. Recall that  $\Sigma$  in Equation 2.1 controls the width of the Gaussian window. In order to maintain self-similarity of the wavelets,  $\Sigma$  is made a function of spatial frequency:  $\Sigma = \sigma / |k|$ . For a given spatial frequency, then, increasing  $\sigma$  will increase the Gaussian window and allow more ringing of the sinusoid. Conversely, decreasing  $\sigma$  causes the outer oscillations to be reduced and dampens the ringing. We have found that  $\sigma = \pi / 2$  provides a good trade-off between ringing and maintaining a good Gabor shape.

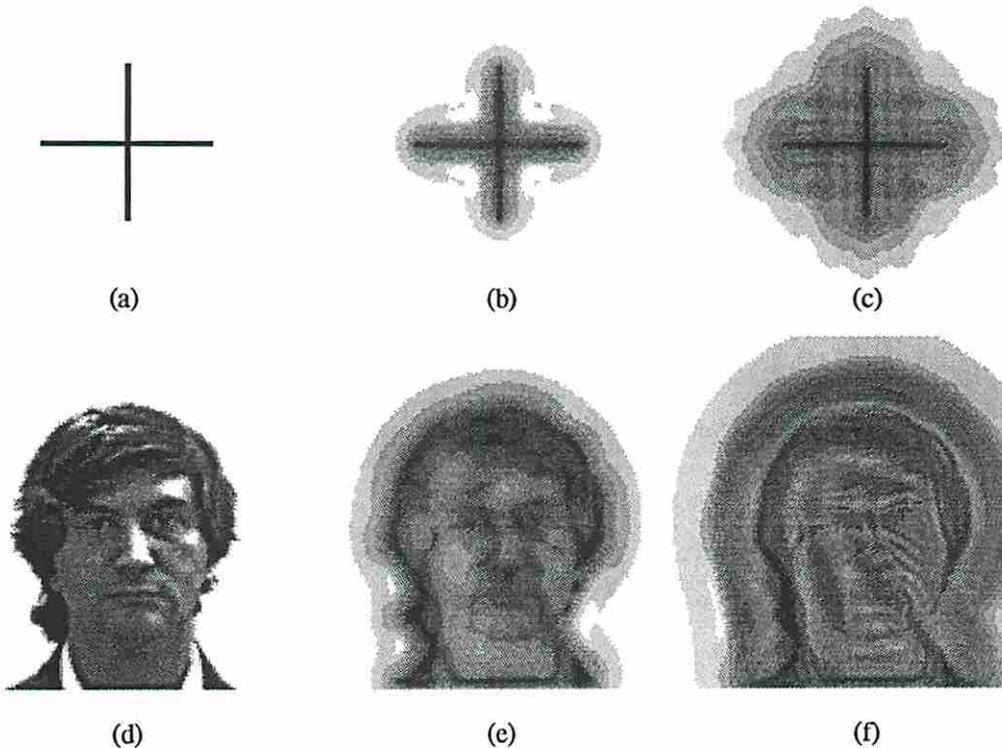


Figure 5.4 - Response of varying  $\sigma$  on jet magnitudes. (a) and (d) are input images, (b) and (e) are jet magnitudes resulting from  $\sigma = \pi / 2$ , and (c) and (f) are jet magnitudes resulting from  $\sigma = \pi$ . Note the ringing in (c) and (f): close examination shows ringing at each spatial frequency of the MWD.

Figure 5.5 shows the results of FFT convolution with and without zero padding. Although zero padding takes approximately four times longer and requires four times as much space, it eliminates the aliasing problems associated with insufficient sampling rates. (Unless otherwise noted, all results given have been obtained with zero padding.)

Figure 5.6 shows jet images for five different jets taken from five different features: four right angle corners and one right angle cross. Each jet was taken from a point at a vertex. Each block in the jet images is a Morlet wavelet component; the representation is identical to that of Figure A.1b (Appendix A) with filter orientation along the horizontal axis and spatial frequency along the vertical axis. The upper four rows correspond to real components while the lower four rows correspond to imaginary components. From examining the jet images, it becomes evident that the discrimination of the feature based solely on the real components is unlikely to be reliable or even possible. By introducing the complex components, the task of discriminating these features is eased.

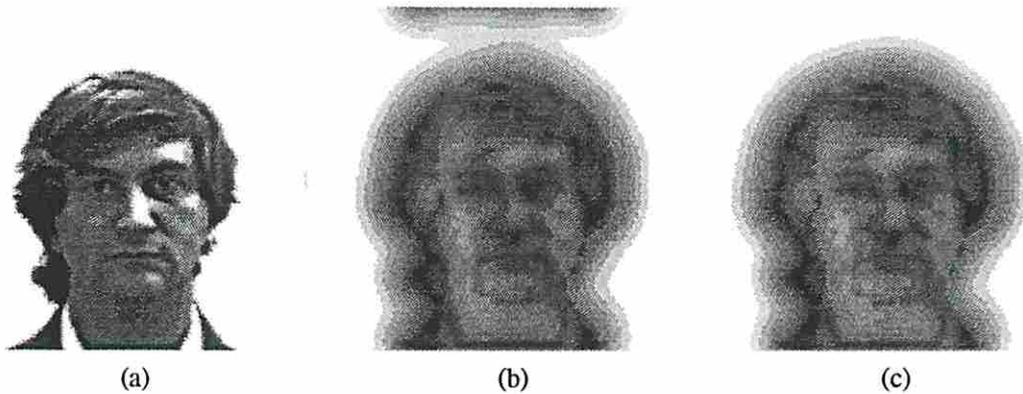


Figure 5.5 - Results of FFT convolution with and without zero padding. (a) is the input image, (b) is the jet magnitude image from a MWD using FFT convolution in which there was no zero padding, and (c) is the jet magnitude image from a MWD using FFT convolution in which there was zero padding. Note that the aliasing in (b) manifests itself as vertical wrap around. Horizontal wrap around also occurs, however due to zero-valued vertical edges, it is not apparent.

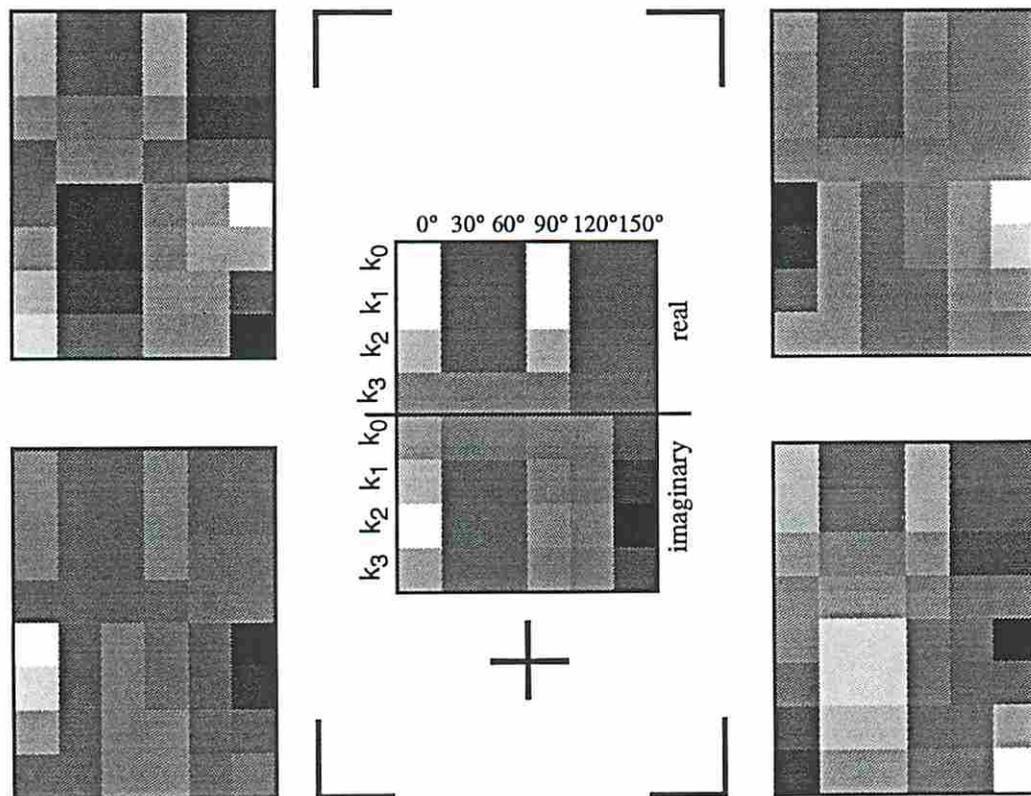


Figure 5.6 - Jet images for five features: four corners and a cross. The use of real components is insufficient for the discrimination of these features. Complementing the real components with imaginary components (thus forming a complex jet) makes the features discriminable.

Figure 5.7 shows the results of continuity processing. Recall that continuity is a measure of homogeneity of a jet with respect to its surround and is used in selecting reliable features during graph formation. Specifically, the continuity value of a jet is the average dot product of the jet with respect to its neighborhood. See Equation 3.1.

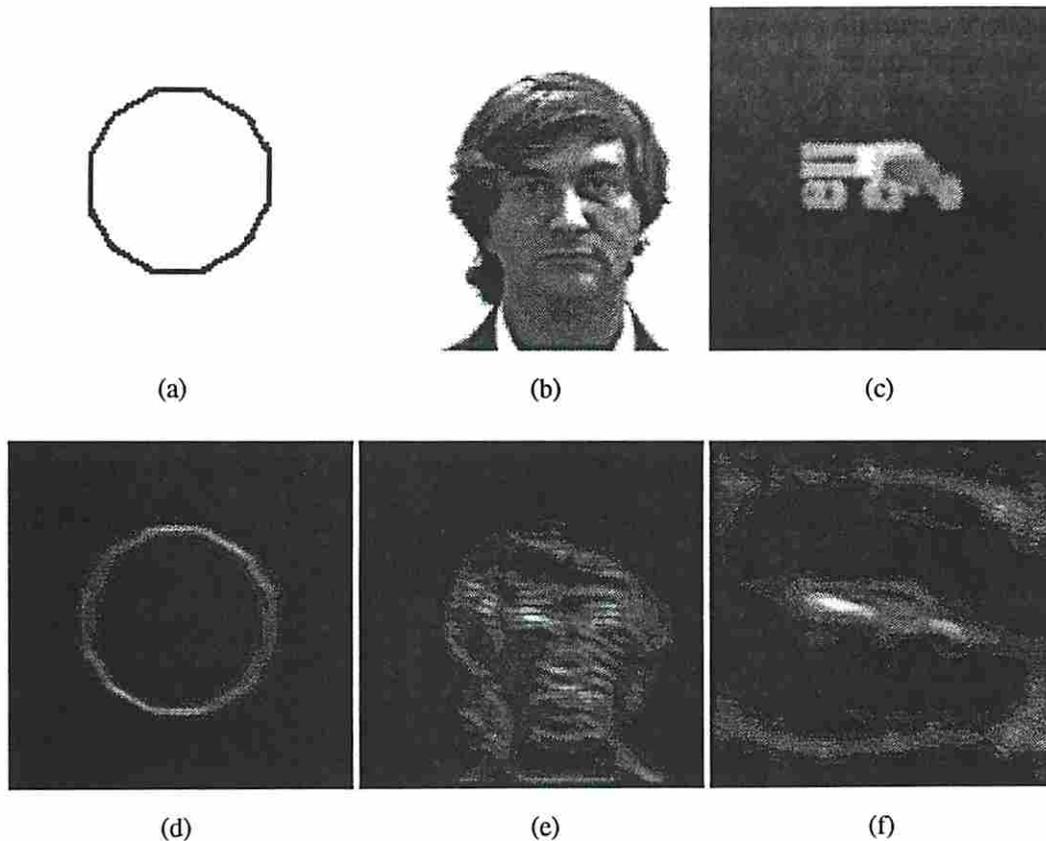


Figure 5.7 - Jet continuity images. (a) through (c) are input images, (d) through (f) are continuity images. (Black is high, white is low.) The continuity of a jet is defined as the average inner product between the jet and its surround. (d) and (e) use 1.42 pixel radius for the surround; (f) uses a 3 pixel radius.

### 5.1.2 Adaptive Vector Quantization

This section presents some of the results obtained during the development of the AVQ module. Because AVQ proceeds in two phases, learning and quantization, we provide results separately for the corresponding phases.

The theory for the AVQ portion of our system is given in Chapter 2. Flowcharts for the actual code can be found in Appendix B and the Macintosh dialog boxes, with their various parametric options, are in Appendix F.

We begin by showing the results of learning on a self-organizing vector map given a simple (and now familiar) training set: the four corners and a cross. Figure 5.8 uses the jet array representation of Figure A.1 (Appendix A) to show the original random vectors before training, the training jets, and the resulting learned vector map. Five of the nine training jets had the largest magnitudes of their image and correspond to points at the center of each vertex. The other four training vectors also had high jet magnitudes and were selected somewhat arbitrarily. The map was assumed to have a wraparound topology, i.e., the left side connects to the right and the top connects to the bottom. The update gain was set at 0.2 and was independent of iteration or jet magnitude. The neighborhood update radius was 3 pixels.



Figure 5.8 - Self-organization of a vector map. A  $10 \times 10$  random vector map self-organizes, using the AVQ algorithm described in Chapter 2, to produce the vector map shown on the right. Five of the nine training jets represent vertices: four corners and a cross. The remaining four training jets were selected somewhat randomly from the four-corners-and-a-cross data set.

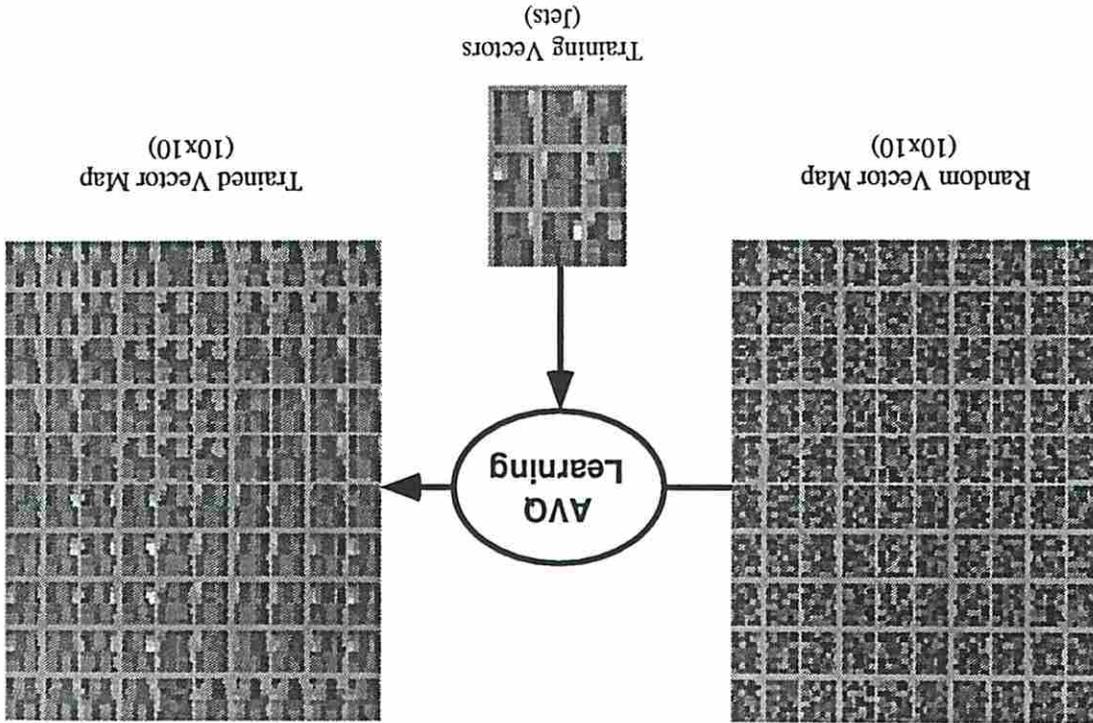


Figure 5.9 shows the progression of the quantization error during training and demonstrates a logarithmically asymptotic convergence. Learning was terminated after 2000 iterations.

Figure 5.10 shows the results of quantizing the original training vectors after learning. Note the regular placement of the training vectors in the vector map with the exception of the two neighboring mappings on the right side of the map. Upon further inspection, however, one will note that two of the training vectors are very similar in appearance; it is therefore reasonable to expect that their mappings in the topological feature space would be proximate. Other points of interest include a relatively untouched vector in the top row of the trained vector map and what looks like phase transitions in the map (i.e., sections of the map in which there are large differences between vectors). These phase transitions, or *map cracks*, are caused by the mapping of high dimensional features into a lower dimensional (2D) space. By high dimensional features we refer not to the dimensionality of the vector but to the dimensionality inherent to the feature. Map cracking raises the lower bound on minimum quantization error and can be eliminated by determining the dimensionality of the features and using vector maps of that (or greater) dimension. We have not been concerned with map cracking in our work because our quantization error after training has been acceptably low.

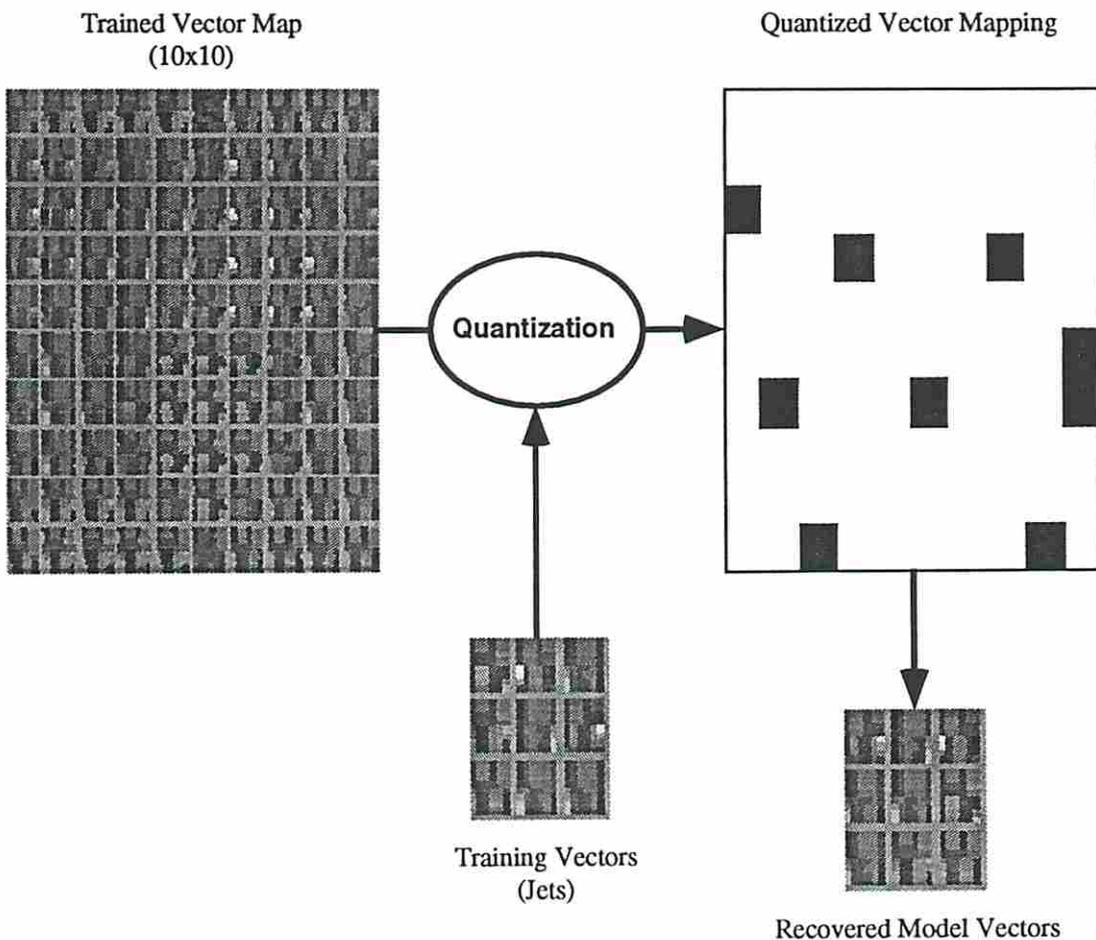


Figure 5.10 - Quantization results using trained vector map (from Figure 5.8) and the original training vectors. Shown at right is a mapping of the training vectors mapped to the trained vector map. Below is the set of model vectors (i.e., vectors taken from the vector map) that result from the quantization of the training vectors. Visual inspection indicates (and numerical analysis confirms) that the model vectors are almost identical to the training vectors.

Figure 5.11 shows two feature images: the results of a vector quantization operation on the jets of a Stickville cross and an infrared truck. Each color in the feature image represents a different feature. (For presentation purposes, the feature images are 2x larger than the input images.) The model vectors, against which this data was quantized, were not learned, but were hand generated from the features shown in Figure 2.11, i.e., six oriented lines and pairwise intersections of those lines. In addition, both the model vectors and jets in this data set are real. The vertical red patches in the cross feature image correspond to patches of vertical line features (label 3 in Figure 2.11); the horizontal blue patches correspond to patches of horizontal line features (label 0 in Figure 2.11). In the center of the cross feature image is a lighter blue square that represents a right angle cross feature (label 8 in Figure 2.11). The second feature image, that of an infrared truck, is quantized using the same model vectors designed for Stickville and shows a patchwork of features. Nonetheless, it may be observed that two familiar features (horizontal and vertical lines) are present in the truck feature image and in appropriate locations. The fragmentation of the truck feature image leads to a lack of reliability in the selection of features for nodes in the ensuing graph formation process and is substantially reduced by learning the proper model vectors, reducing the jet spatial frequency, and by using complex jets.

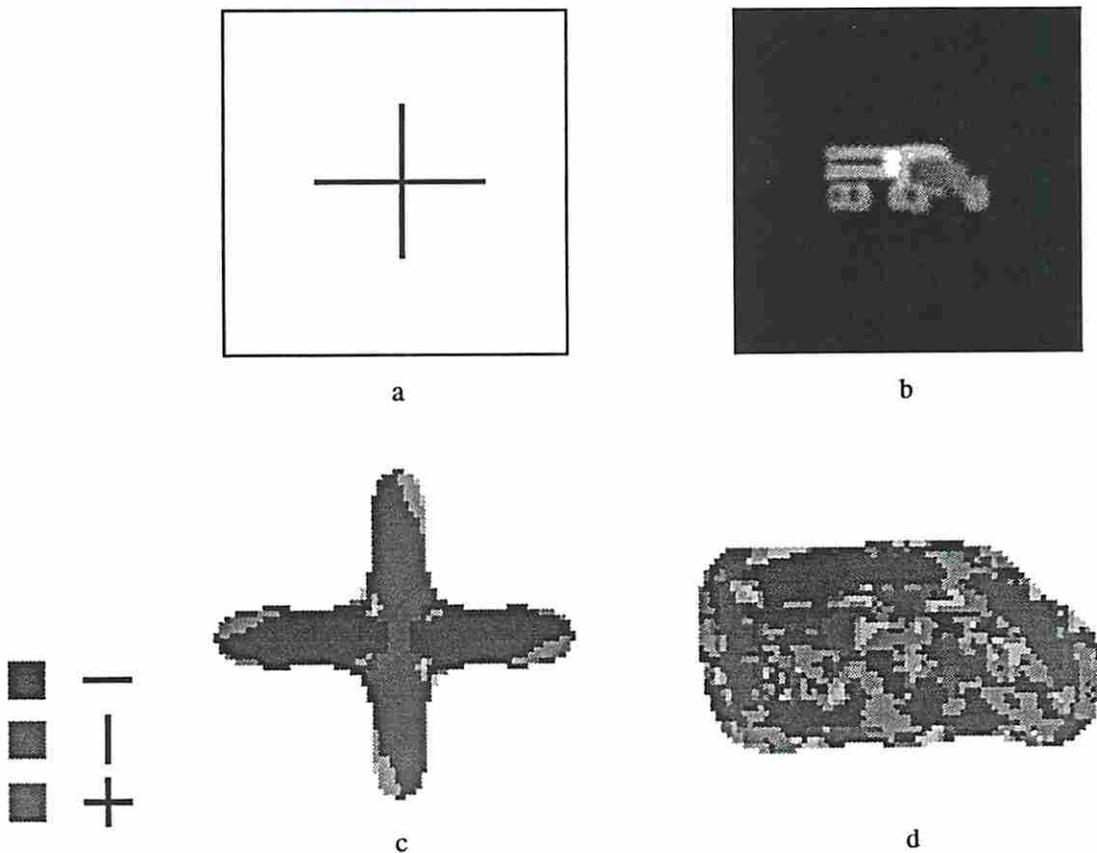


Figure 5.11 - Feature images using model vectors from Stickville. (a) is an input Stickville cross, (b) is an input infrared truck, (c) is the feature image for the cross, and (d) is the feature image for truck. The model vectors against which the jets for (a) and (b) were quantized were handcrafted and represent the features in Figure 2.11.

### 5.1.3 Graph Formation

This section presents examples of processing used in the generation of stable object graphs. The theory for the graph formation portion of our system is given in Chapter 3. Flowcharts for the actual code can be found in Appendix C and the Macintosh dialog boxes, with their various parametric options, are in Appendix G.

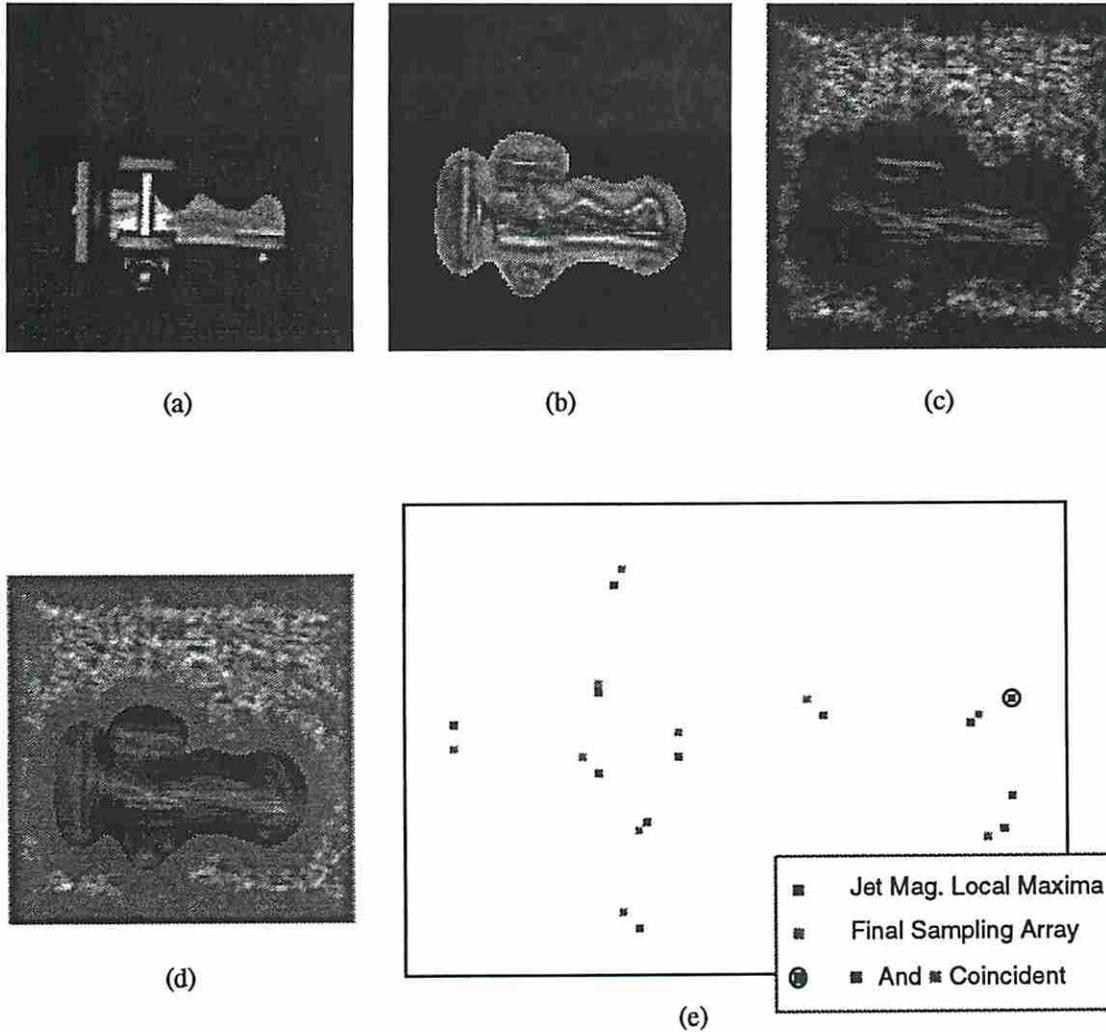
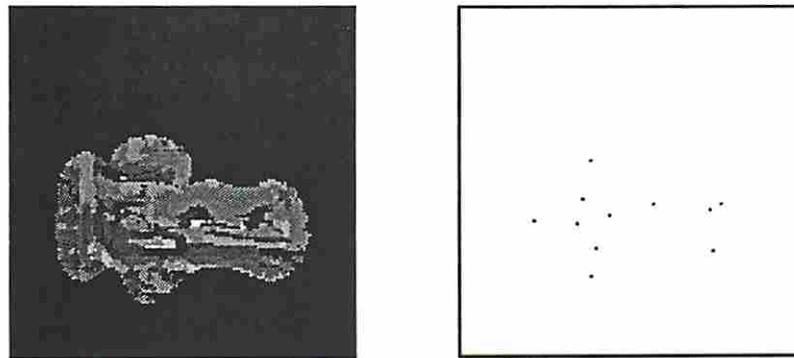


Figure 5.12 - The node selection process combines quantization error with jet continuity (Equation 3.3) to modify an initial sampling array based on saliency. The computed sampling array selects feature labels from the feature image and assigns them to nodes in the graph being formed. (a) shows the input object, (b) the quantization error image, (c) the continuity image, and (d) the computed  $\Gamma$ -array. Black is high; white is low. (e) shows the initial sampling array in black and the final sampling array in grey.

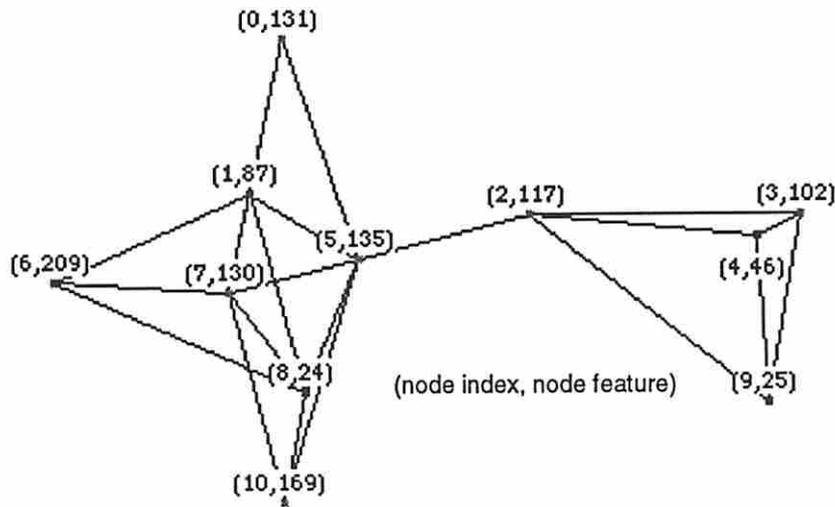
Figure 5.12 graphically illustrates the process by which nodes are selected for inclusion in object graphs. Recall from Chapter 3 that a feature image forms the substrate for all node labels and that the saliency measure is used to initialize the sampling array (i.e., saliency is the primary selection criterion). Recall also that a combined function of label accuracy and

reliability is used to adjust the sampling array to obtain salient and stable features. Figure 5.12 shows the error associated with the vector quantization for the toy biplane and is a measure of label accuracy, i.e., how close each jet is to the model vector to which it was quantized. Also shown in Figure 5.12 is the continuity image which gives the degree similarity each jet has with its neighborhood. In this data set, the neighborhood is defined with a radius of 1.42 pixels. The two arrays, combined using Equation 3.3, form the  $\Gamma$ -array, also shown in Figure 5.12. In this particular computation,  $\rho = 0.7$ . The final frame of Figure 5.12 shows the initial sampling array (i.e., the local maxima of the saliency function) in black and the final sampling array in grey. In one case, the circled black pixel, the initial sampling point remained unmodified by the  $\Gamma$ -array.



(a)

(b)



(c)

Figure 5.13 - The nodes of a labeled, locally connected graph are formed by sampling a feature image with a sampling array, assigning the selected feature labels to nodes, and connecting the nodes. (a) shows the feature image for the object in Figure 5.12a, (b) shows the sampling array, and (c) shows the resulting graph.

The final results of the graph formation process are illustrated in Figure 5.13. The feature image is sampled by the sampling array with the selected features being assigned to nodes in the graph. The edge constraints, imposed parametrically and by the relative size of the object, are used to generate connections between the nodes. The parenthesized text near each node in the graph of Figure 5.13 provides the index number and the feature label of the node. For example, (2,117) indicates node number 2 with feature label 117. The feature set used in this example comes from a 15 x 15 vector map and was generated using jets from a large number of similar objects. Topologically proximate features are therefore labeled consecutively or consecutively on modulo 15.

#### 5.1.4 Dynamical Link Graph Matching

This section presents an initial demonstration of DyLink graph matching in which seven handcrafted graphs of Stickville objects are used. The theory for the graph matching portion of our system is given in Chapter 4. Flowcharts for the actual code can be found in Appendix D and the Macintosh dialog boxes, with their various parametric options, are in Appendix H.

We begin this section by considering the behavior and stability of the Rail Capture dynamics. We define a *Rail Capture function* that, when given a *correlation-input* (i.e., correlated, uncorrelated, or anti-correlated), responds with the Rail Capture dynamic of Equation 4.24. We define a *trial* to be some number of iterations where a different correlation-input is applied to the Rail Capture function at each iteration. Let the correlation-inputs to the Rail Capture function be random and specified by probability (e.g. 20% correlated firings, 20% anti-correlated firings, 60% uncorrelated firings).

Recall that a characteristic of the dynamic is that it drives the link value to a rail and, once there, tends to stabilize it. If we take a large number of trials for a given correlation distribution and measure (at the end of each trial) the number of times the link is driven to which rail, we get an idea as to its behavior. What we find is that the rail to which the link is driven is closely associated with the input distribution, i.e., a high ratio of correlated/anti-correlated firings leads to a predominance of high-rail latching and vice versa. Ratios near one lead to a lack of rail latching entirely.

The Rail Capture dynamic is insensitive to P (the peak value of the inverted parabola) but looks best in the neighborhood of 0.15 to 0.25. Figure 5.14 gives an example of the output from one set of trials for P=0.10 and should be interpreted in the following manner. The axes of the figures represent the amount of correlated or anti-correlated input that is applied to the function. Uncorrelated input is implied by

$$\% \text{ uncorrelated} = 100 - (\% \text{ correlation} + \% \text{ anti-correlation}). \quad (5.1)$$

A point in the upper-left triangle of any figure gives the relative amount of rail latching that occur given the pixel's location with respect to the axes (i.e., the correlation-input to the function). For example, a 50% correlated input combined with a 10% anti-correlated input (implying a 40% uncorrelated input) shows that the function will almost always latch to the high rail (i.e., the relevant pixel in the High Rail Latching figure is high while the corresponding pixel in the other figures is low). Conversely, for a 0% correlated input

combined with a 0% anti-correlated input (implying a 100% uncorrelated input), the first two figures show few, if any, rail latching while the third shows a strong tendency to no latching. This is, of course, the desired behavior. Note that the lower-right triangle is, and should be, zero. That area represents points at which

$$\% \text{ correlated} + \% \text{ anti-correlated} \geq 100. \quad (5.2)$$

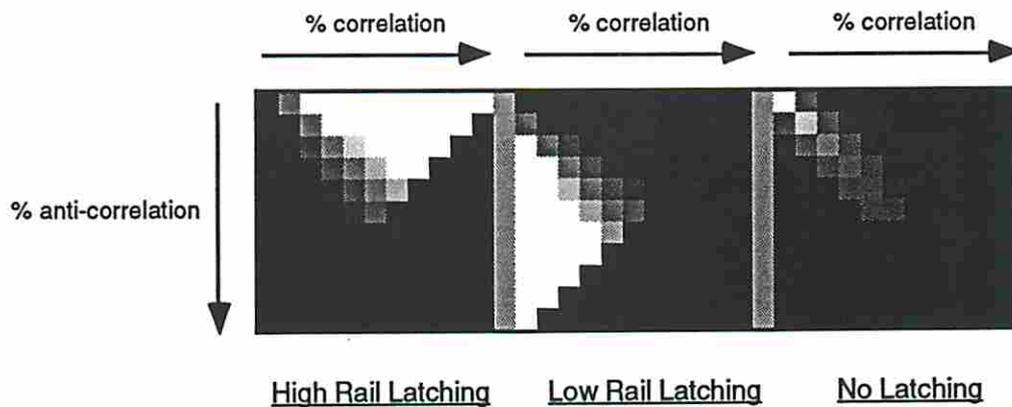


Figure 5.14 - Behavior of the Rail Capture dynamic with respect to different correlation-inputs. White is high and black is low. The horizontal axes represent the relative amount of correlated input provided to the function whereas vertical axes represent the relative amount of anti-correlated input. Uncorrelated input is implied by Equation 5.1.

Trials	1000
Iterations/Trial	100
$J_{(t=0)}$	0.500000
P	0.100000
d1	1.000000
d2	1.000000

Table 5.1 - Parameters used to generate Figure 5.14.

Figure 5.15 shows seven objects segmented into their features. Graphs are formed by generating edges between nearest neighbor, collinear features. Saliency values are arbitrarily assigned in a connected sequential manner.

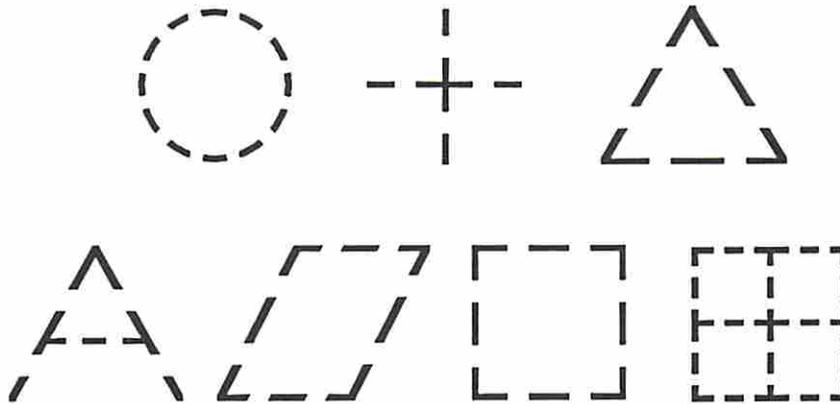


Figure 5.15 - Stickville objects manually segmented into their features. The objects are referred to as Polygon, Cross, Triangle, Letter A, Rhombus, Square, and Window.

The first thing to notice about the data set is that the similarity between objects varies from unrelated to very similar. The Polygon, for example, is completely unlike any other object. As a result, the graph matcher easily recalls the Polygon which has no competition from the other objects. See Figure 5.16. (In Figures 5.16 through 5.21, no feature-correlation based links are used. Due to the sigmoidal gain, g-cell activity is at a minimum at 0.438 and at a maximum at 0.562. The convergence time window is 20 iterations.) The Triangle and the Letter A objects are similar, consequently the graph matcher recalls both when either is input but recalls the correct graph more strongly. See Figures 5.17 and 5.18. While the Square and the Cross have little in common, both are used to form the Window. Figures 5.19 and 5.20 show the responses of the graph matcher to the Square and the Cross, respectively. Note that both activate the Window, which has some features of both.

Figure 5.21 shows the response to the Window. The system incorrectly determines that the input is the Cross. The reason for this is the way in which the system measures the degree of match. The Cross graph contains 5 nodes whereas the Window graph contains 21. Because the Cross graph is a subgraph of the Window graph and because it consists of only three features, it becomes fully active long before the entire Window graph. A fully active Cross graph will be fully bound and will maximally activate its g-cell. The Window graph, containing more nodes and more features takes longer to fully bind and therefore activates its g-cell to some value less than the maximum. We note here that this problem is highly pathological and is only easily developed by using the simple Stickville features employed in this example.

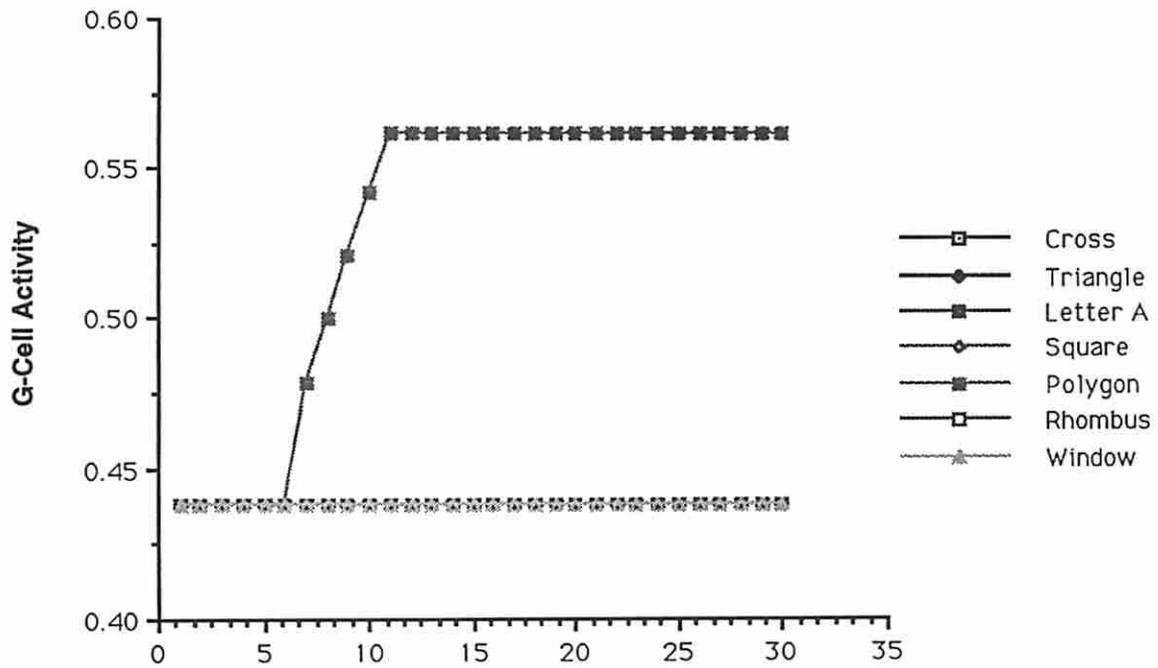


Figure 5.16 - Graph matching results when the Polygon is the input. The only active g-cell corresponds to the Polygon; all others are at their base values.

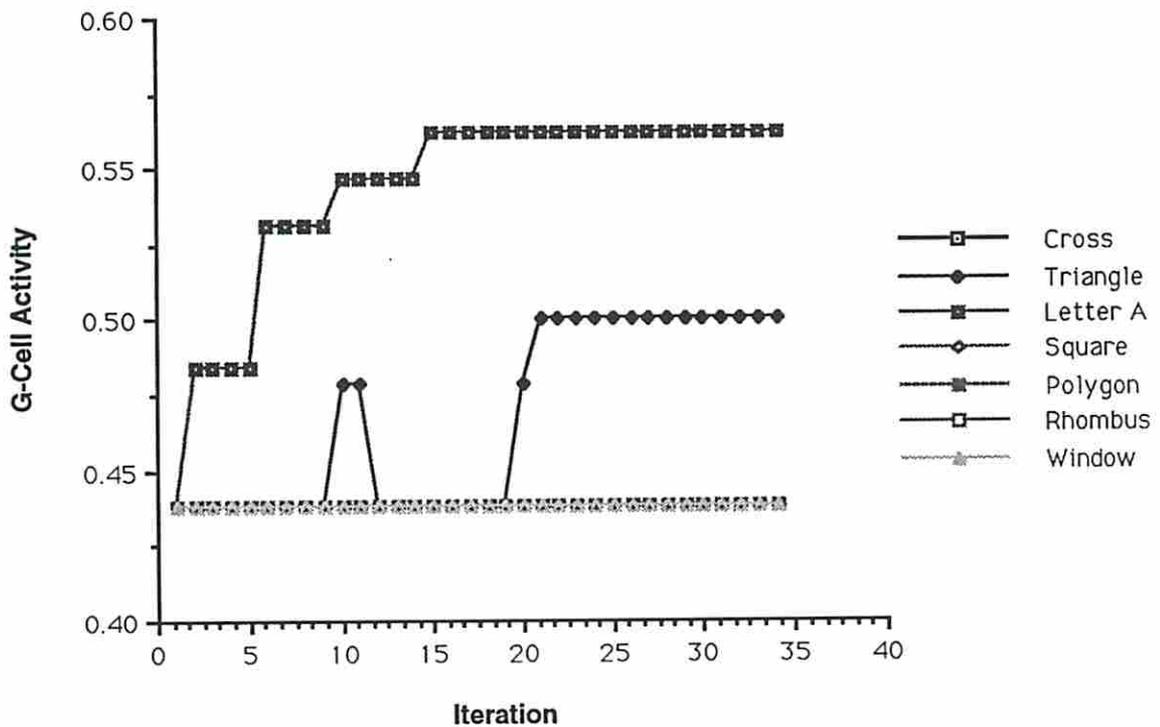


Figure 5.17 - Graph matching results when the Letter A is the input. The only active g-cells correspond to the Letter A and the Triangle; all others are at their base values.

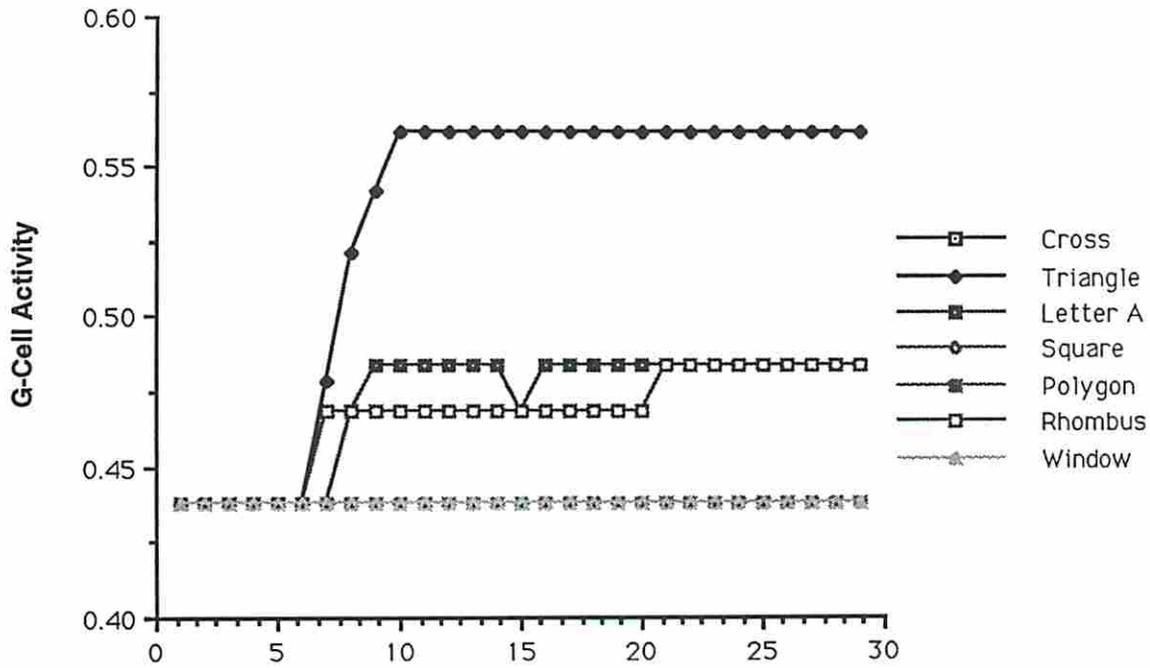


Figure 5.18 - Graph matching results when the Triangle is the input. The only active g-cells correspond to the Triangle, Letter A, and Rhombus; all others are at their base values.

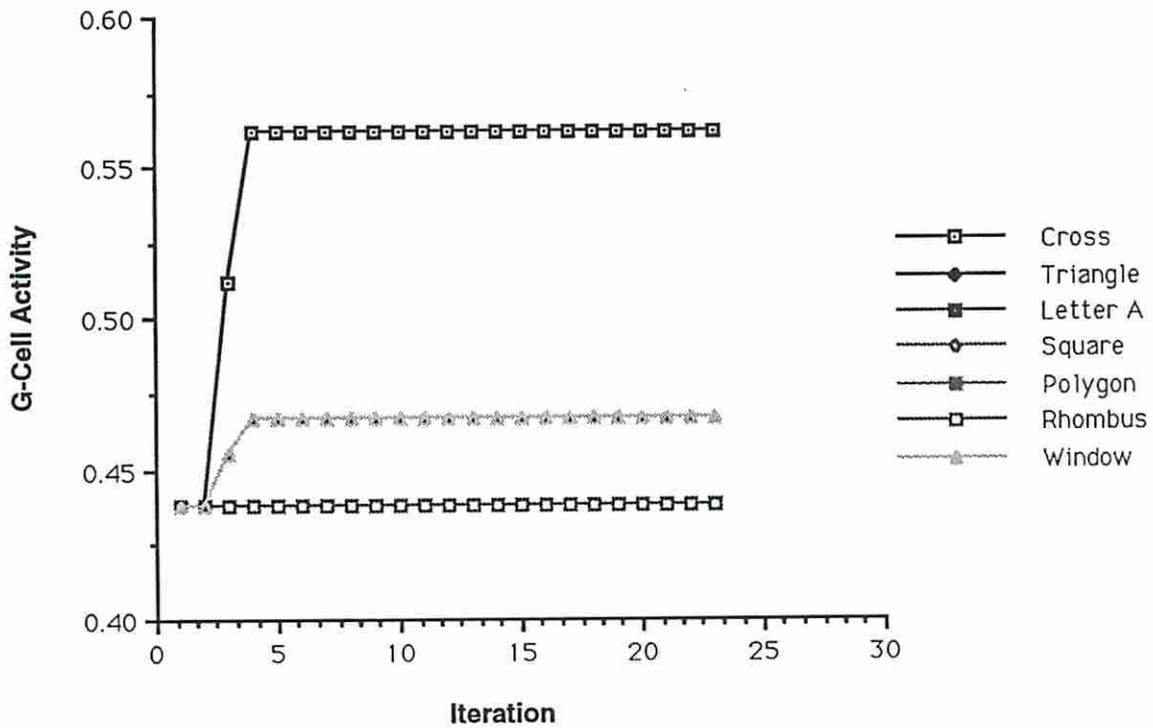


Figure 5.19 - Graph matching results when the Cross is the input. The only active g-cells correspond to the Cross and the Window; all others are at their base values.

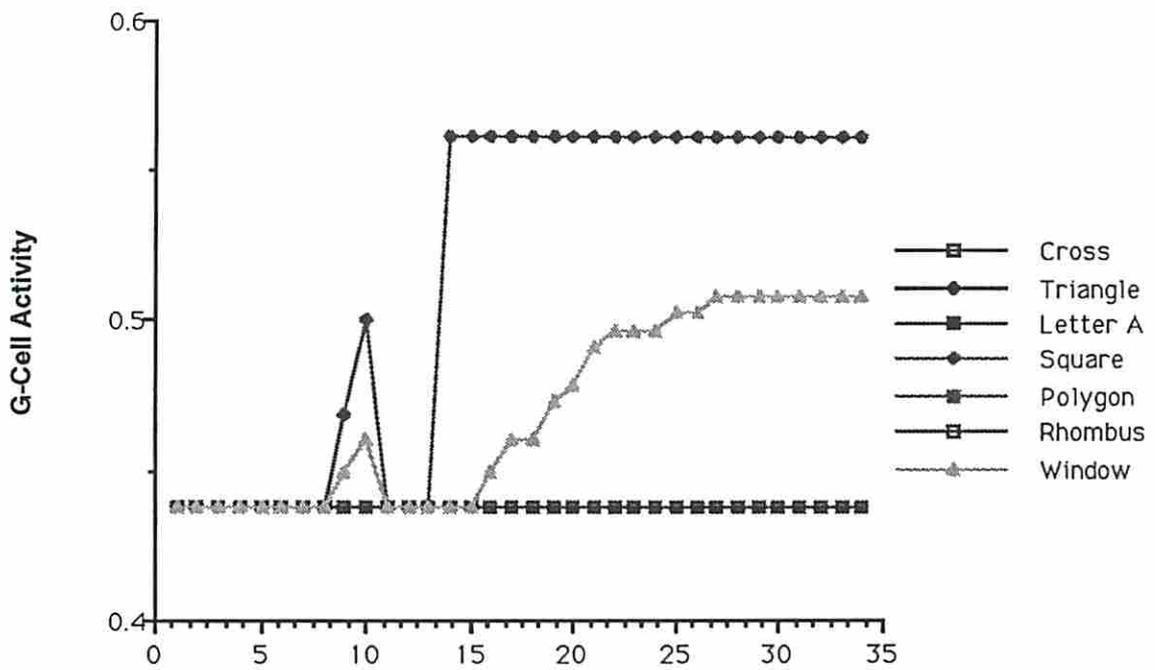


Figure 5.20 - Graph matching results when the Square is the input. The only active g-cells correspond to the Square and the Window; all others are at their base values.

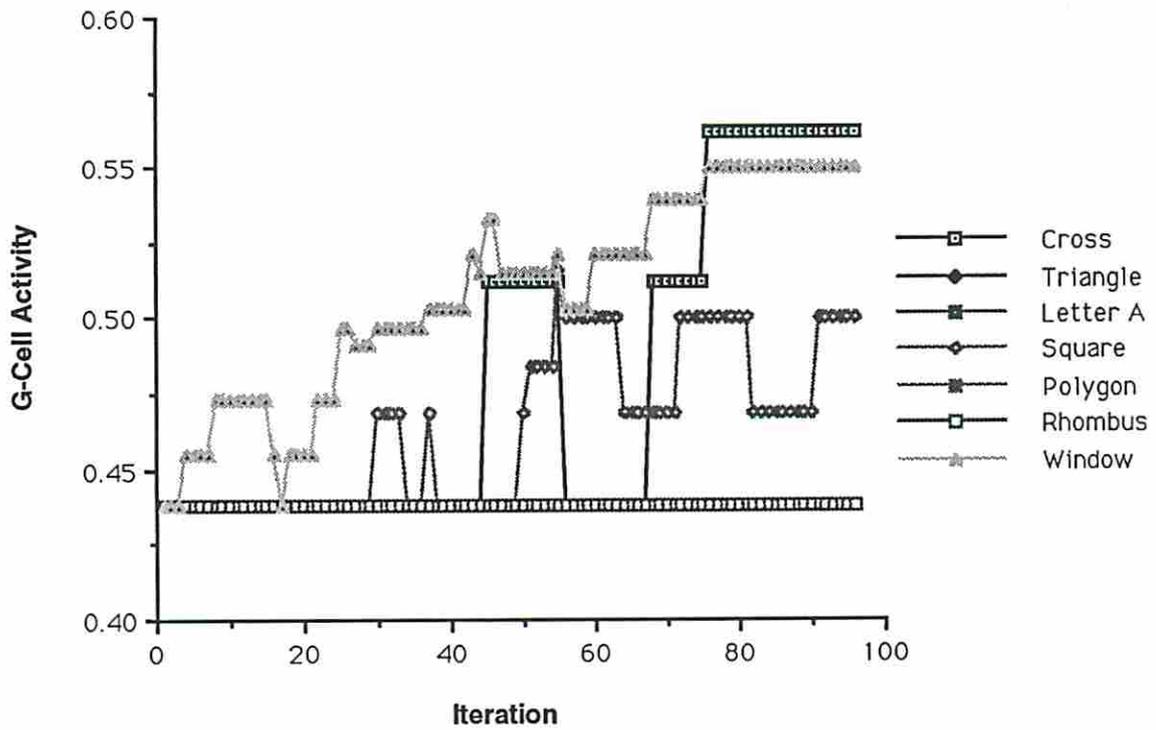


Figure 5.21 - Graph matching results when the Window is the input. The only active g-cell corresponds to the Window, the Cross, and the Square; all others are at their base values. See the text for an explanation of the incorrect association.

## 5.2 Recognition in Toyland

This section shows processing results on a module by module basis for an end-to-end recognition task. Figure 5.22 shows the objects that make up the database: a toy biplane viewed head-on ( $0^\circ$ ), a toy biplane viewed from the side ( $90^\circ$ ), a toy boat, a toy car, a toy helicopter, a toy elephant, a toy football (USC), a real phone, and a real shoe. Although the toy biplanes at  $0^\circ$  and  $90^\circ$  are the same physical object, the orthogonal views are dissimilar enough to store as separate data objects and are so treated. For completeness, Figure 5.23 shows the Laplacian of the objects of Figure 5.22.

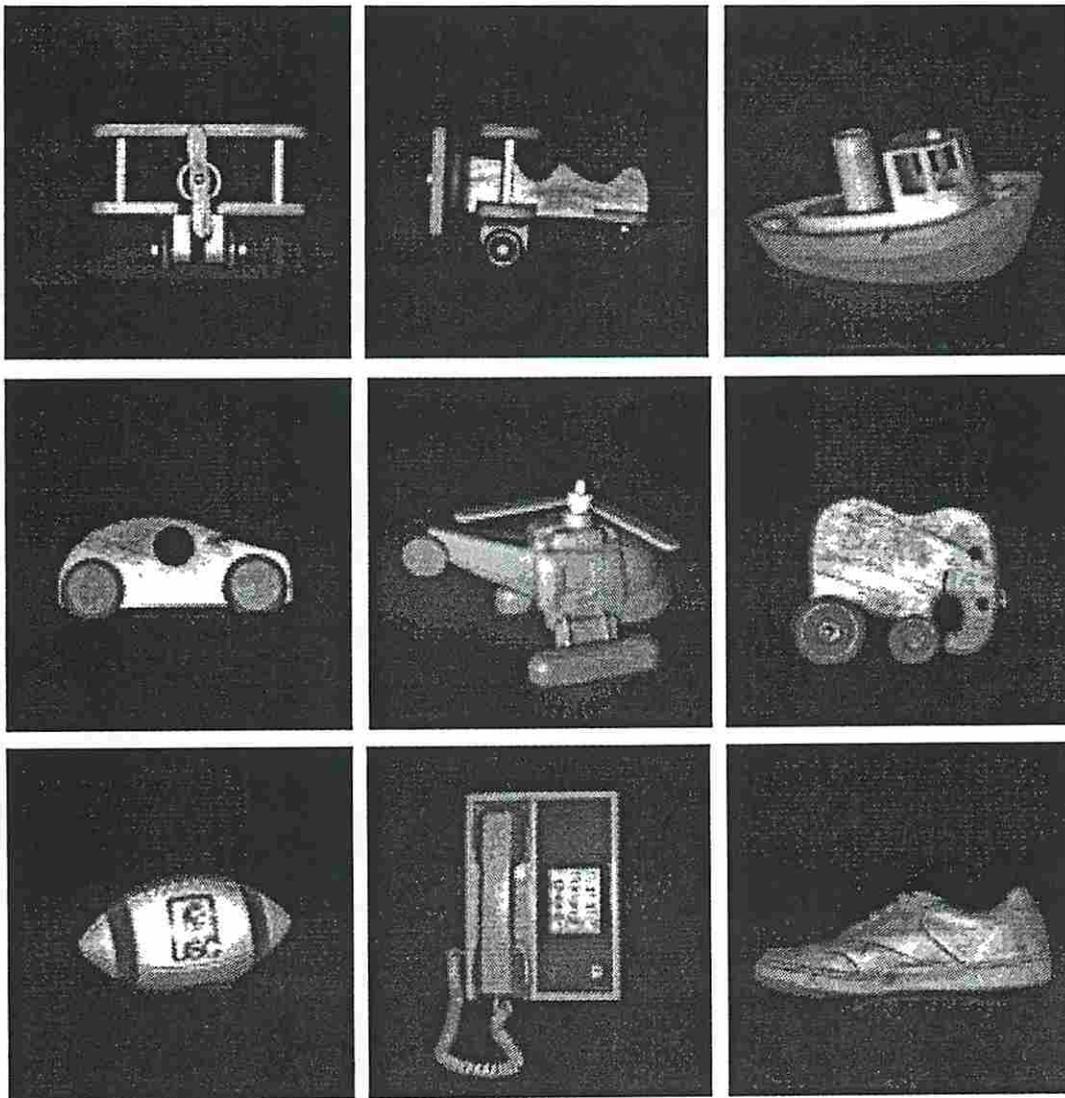


Figure 5.22 - Model objects used to form graph matcher database. In sequence, the objects are a toy biplane viewed head-on ( $0^\circ$ ), a toy biplane viewed from the side ( $90^\circ$ ), a toy boat, a toy car, a toy helicopter, a toy elephant, a toy football (USC), a phone, and a shoe. All images are 128 pixels x 128 pixels x 8 bits.

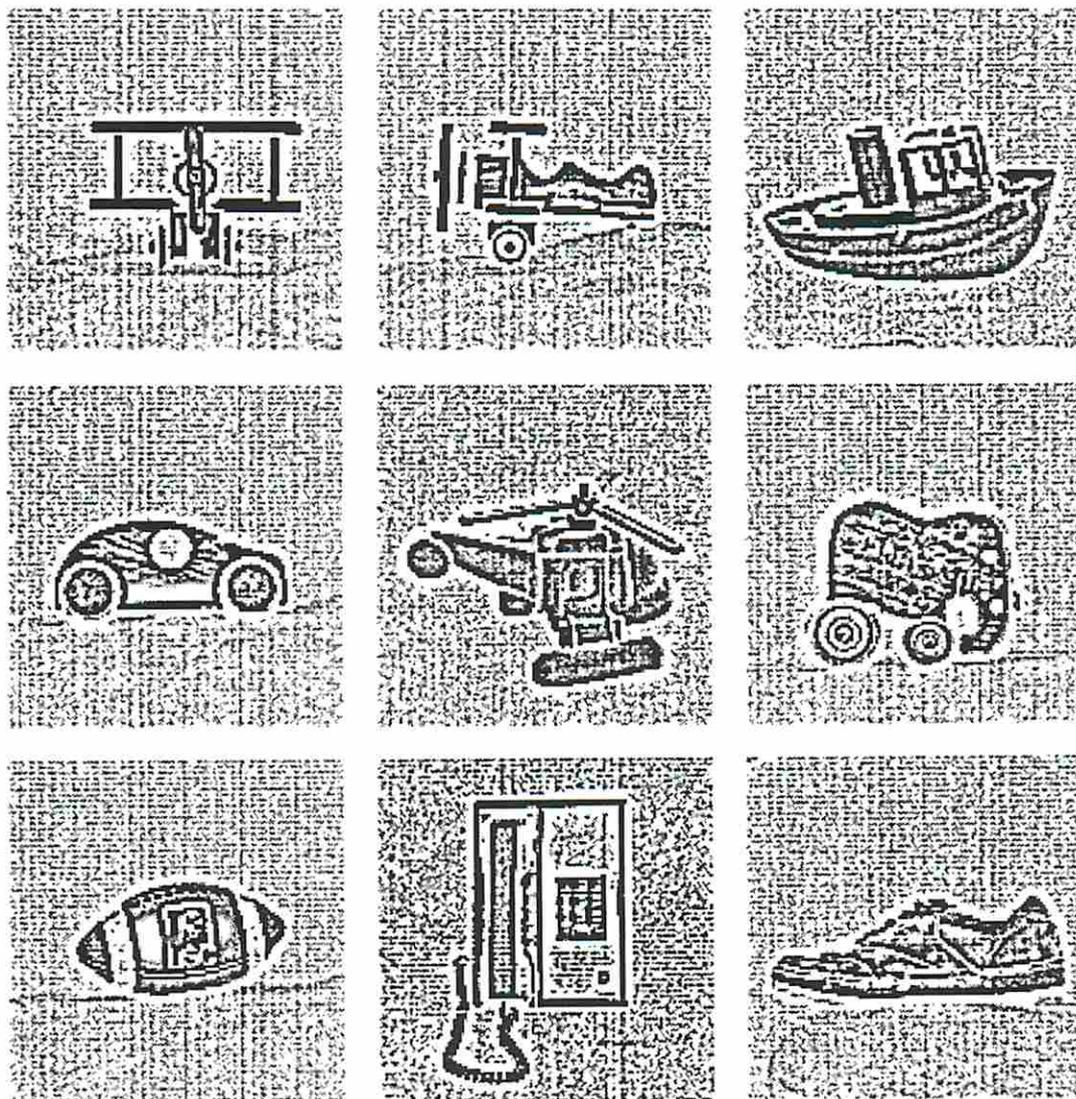


Figure 5.23 - 5x5 Laplacian of the model objects from Figure 5.22.

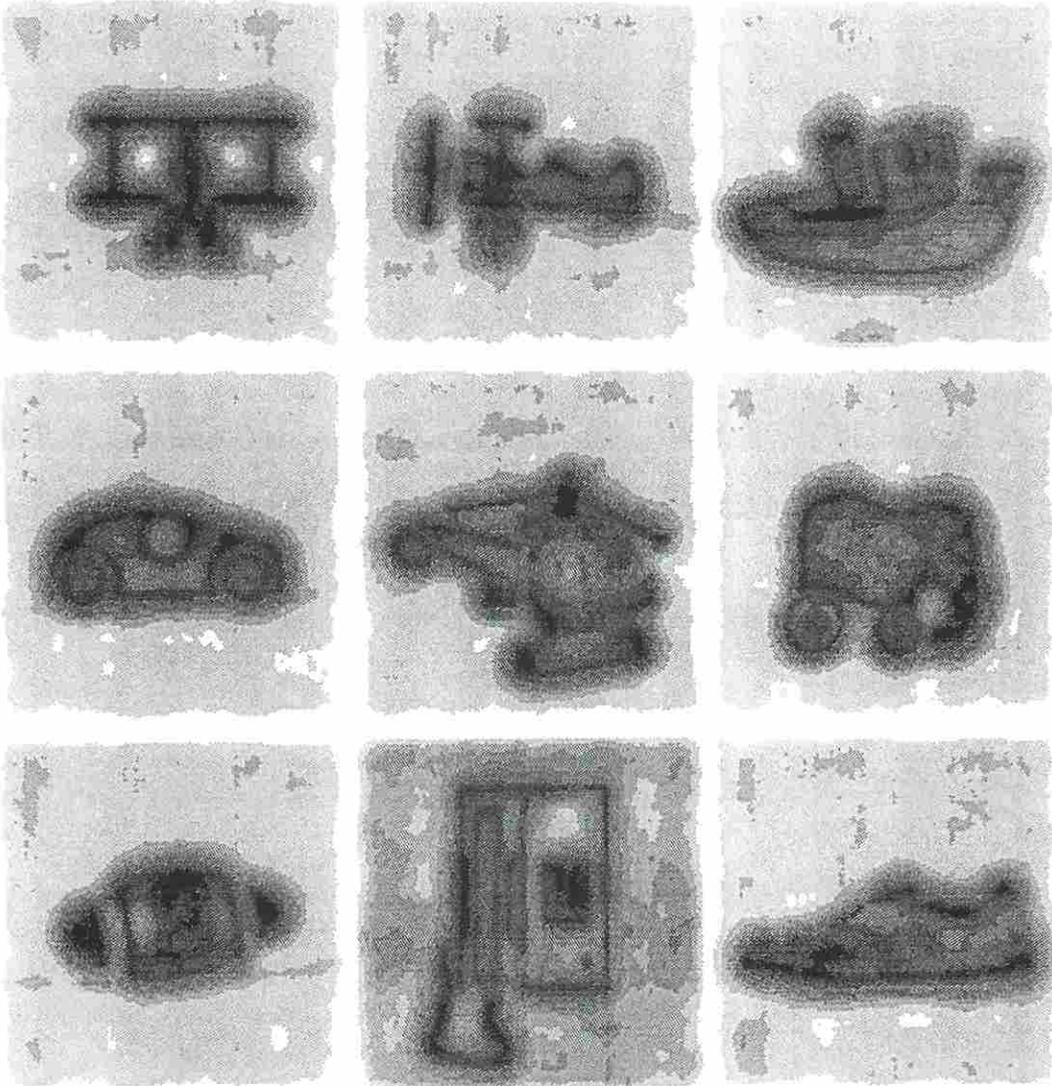


Figure 5.24 - Jet magnitudes of the model objects from Figure 5.22.

The jet magnitudes of the model objects are shown in Figure 5.24 and represent the saliency for each object. They were generated from complex jets using the MWD processing described in Chapter 2 with  $O=6$ ,  $S=4$ ,  $\lambda_{\min}=4$  pixels,  $\sigma=\pi/2$ , and with orientation column tuning on. The jets used for feature vectors employed the same parameters except were one octave down, i.e.,  $\lambda_{\min}=8$  pixels.

Approximately 1100 jets were arbitrarily selected<sup>15</sup> from Morlet blocks<sup>16</sup> for use as training vectors for a self-organizing vector map. The AVQ processing described in Chapter 2

<sup>15</sup> Selection of jets was subject to the constraint that their magnitudes exceed a threshold of 50% of the difference between the maximum and minimum jet values for a given Morlet block. The value of 50% was a somewhat arbitrary selection in itself as the system is extremely robust with respect to this threshold and typically functions well at values from 30% to 70%.

<sup>16</sup> The jets were taken from Morlet blocks that were formed from the model objects images of Figure 5.22 as well as 30 similar images.

was used to train the 15x15 vector map shown in Figure 5.25. A neighborhood radius of 3.0 and a peak update gain constant of 0.3 were used. The update gain was made to be a function of jet saliency and decreased linearly with iteration. Map wraparound was enabled so that the left side of the map was connected to the right and the top was connected to the bottom. Training occurred for 40000 iterations. Figure 5.26 shows the running average of quantization error over iteration. Figure 5.27 shows a 2D histogram of the trained vector map with respect to the training jets; a large value in the histogram (black) indicates a large number of training jets mapping to that model vector in the vector map.

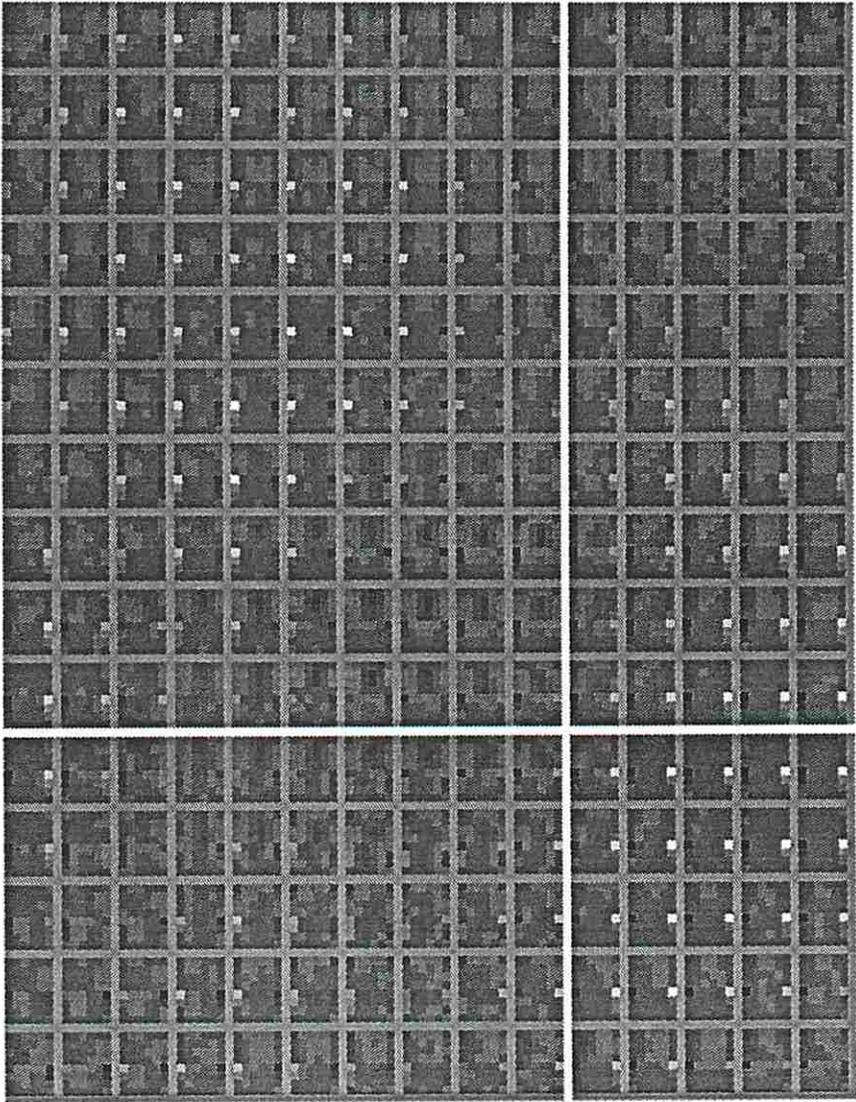


Figure 5.25 - 15x15 vector map trained using 1080 jets from 39 images.

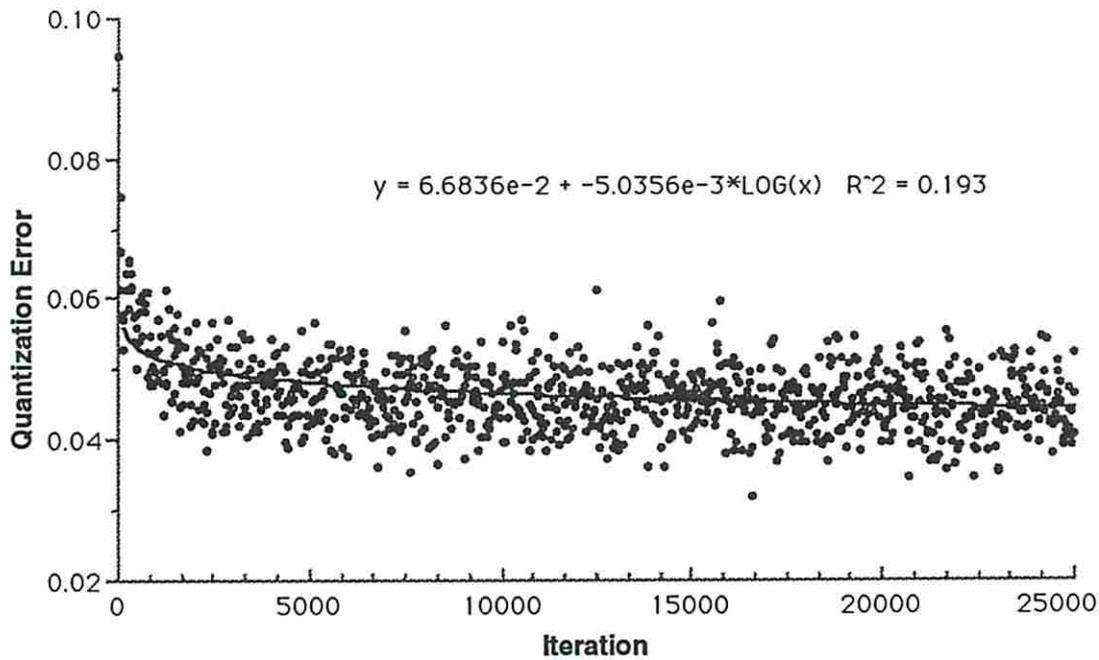


Figure 5.26 - Running average of quantization error over iteration for the vector map in Figure 5.25. A total of 40000 iterations were used to train the map; for presentation purposes, only 25000 iterations are shown. Also for presentation purposes, the initial quantization error of 0.320 at iteration 0 is eliminated. Each data point is determined at the end of 100 iterations.

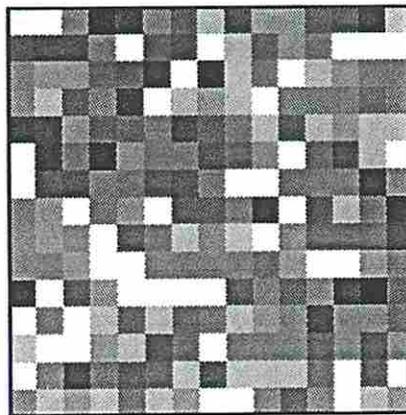


Figure 5.27 - 2D histogram of trained vector map (Figure 5.25) with respect to the 1080 training jets. Black is high, white is low.

After the model vectors have been learned, jets from the Morlet blocks of the model objects are quantized. Figure 5.28 shows the feature images for each of the model objects. Because there are a possible 225 features and only 32 available colors, there are several features mapped to each color. This makes it a little difficult, but not impossible, to get an idea of the feature distribution for each model object.

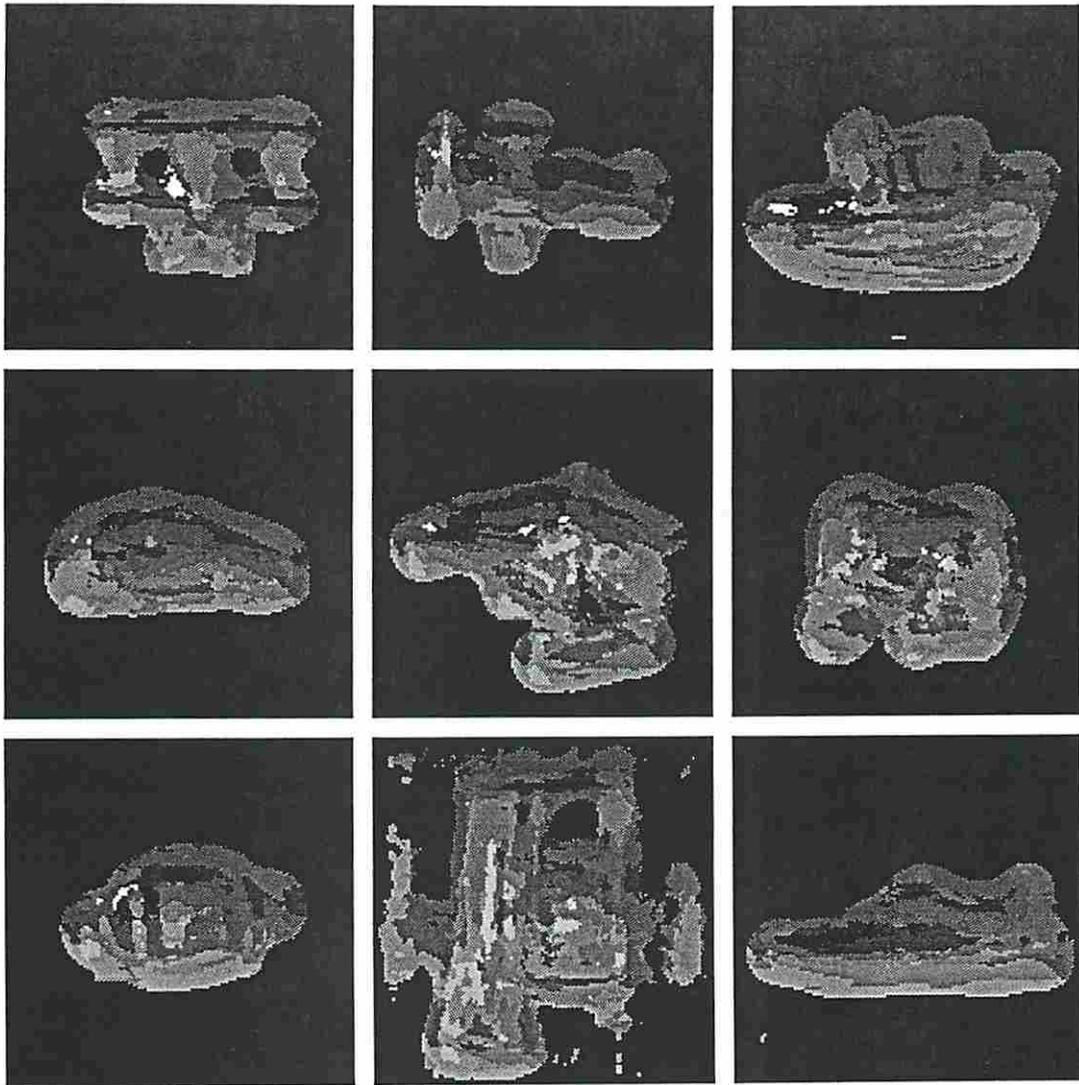
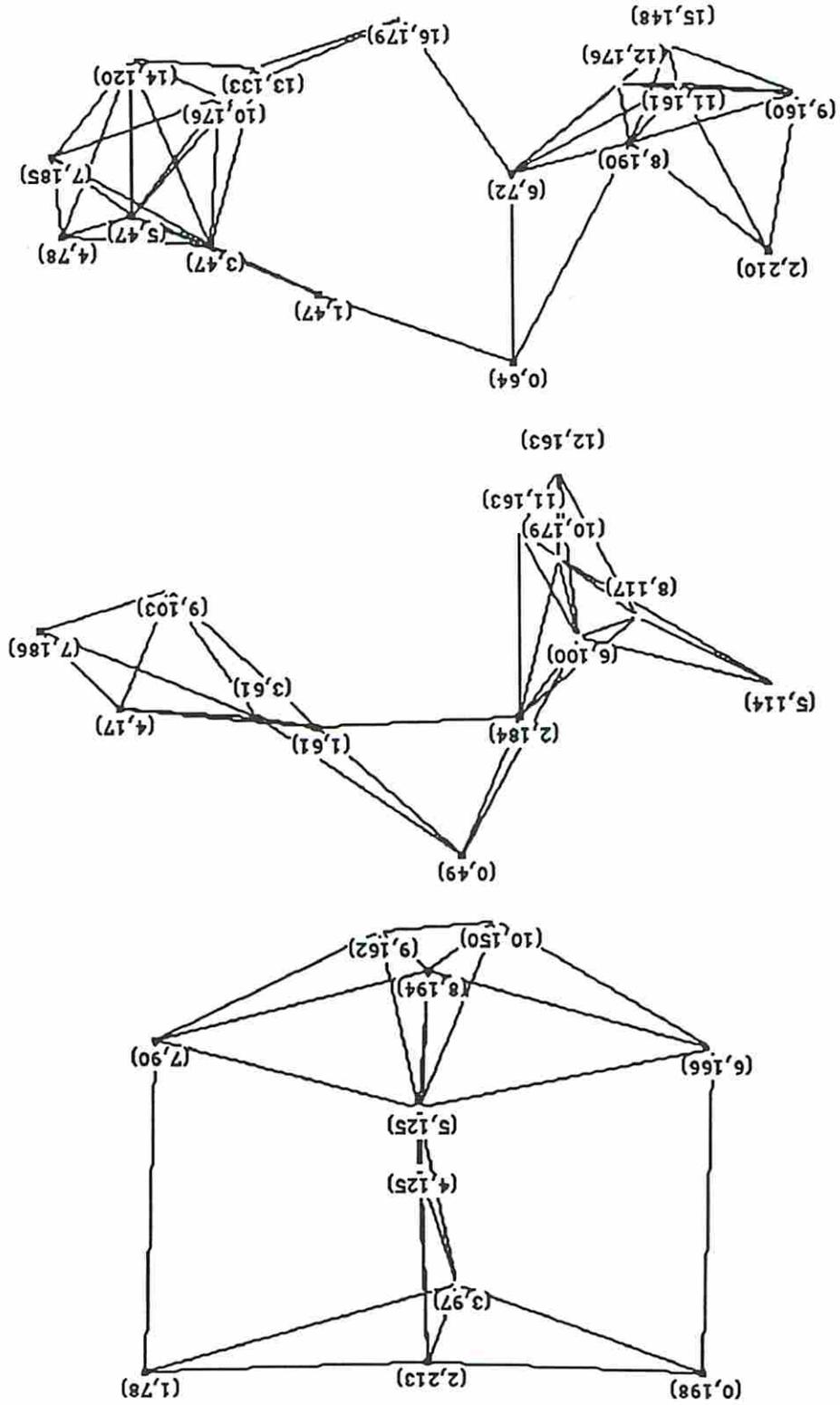


Figure 5.28 - Feature images of the model objects. Because there are 225 possible features and only 32 colors available, several features are mapped to each color.

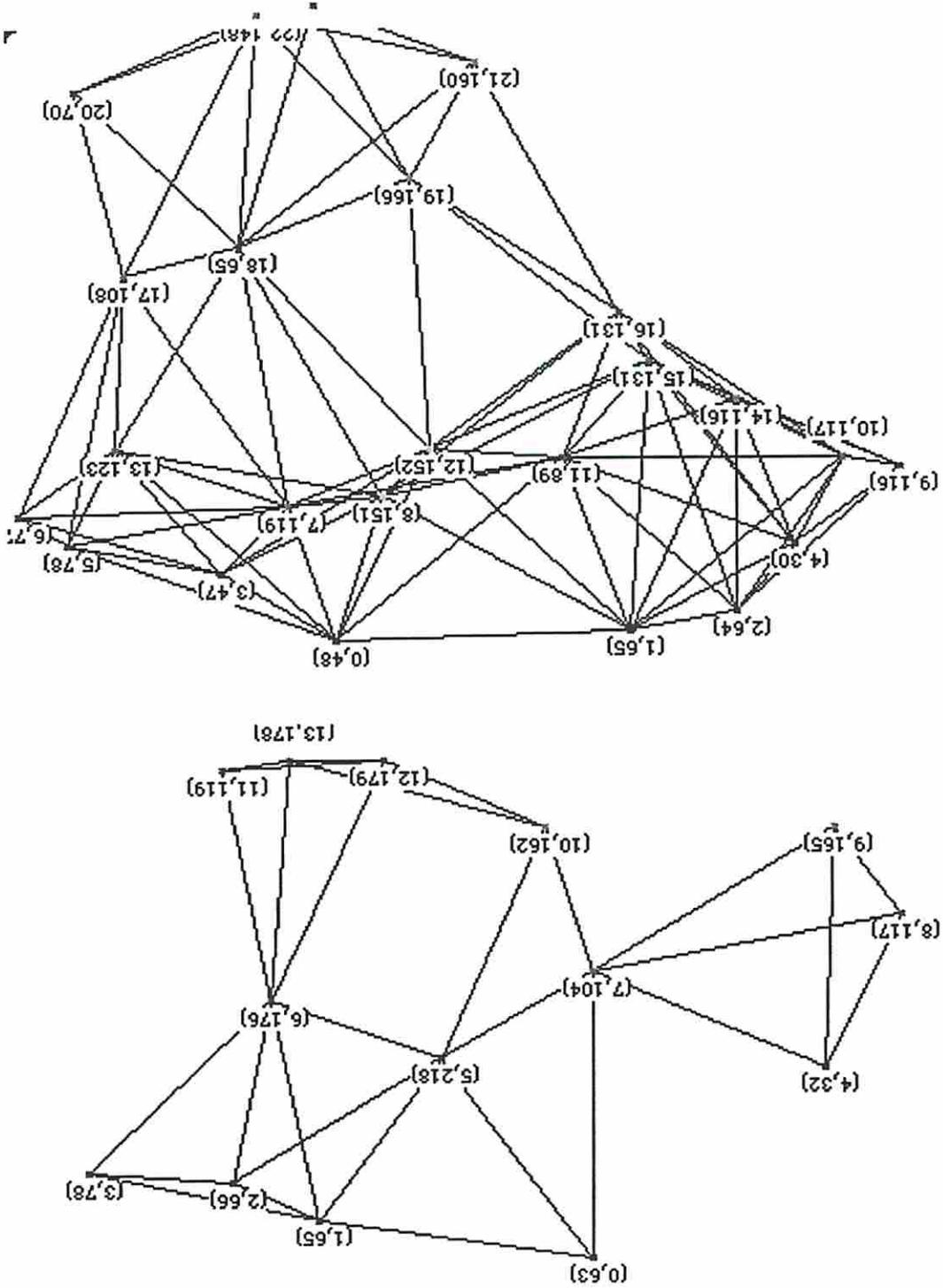
The processing described in Chapter 3 was used to generate graphs for subsequent matching; a subset of these graphs is shown in Figures 5.29 and 5.30. The minimum distance between nodes was set at 5 pixels and the maximum distance was 30 pixels. The minimum number of inbound neighbors was 3 while the maximum was 8. Candidate nodes were required to have a corresponding saliency value of greater than 50% of the maximum minus minimum values. As was the case earlier, the value of the saliency threshold was not critical; values between 30% and 70% work quite well.  $\rho$ , used in the computation of the  $\Gamma$ -array, was set to 0.70. The  $\Gamma$ -array was filtered twice with a 3x3 median filter to eliminate spurious local maxima.

Figure 5.29 - Object graphs of biplane at 0°, biplane at 90°, and car. Each node is labeled in the following manner: (node ID, feature label).



The first test of the system in an end-to-end recognition mode was with the model images (Figure 5.22) used as input. The system was easily able to recognize each of the nine models. Figures 5.31 through 5.35 show the g-cell activity over iteration for each of the model input

Figure 5.30 - Object graphs of the boat and the helicopter. Each node is labeled in the following manner: (node ID, feature label).



cases. (Recall that g-cell activity is a measure of the degree of graph matching with respect to each model object.) From those figures it may be noted that recall for this test was perfect.

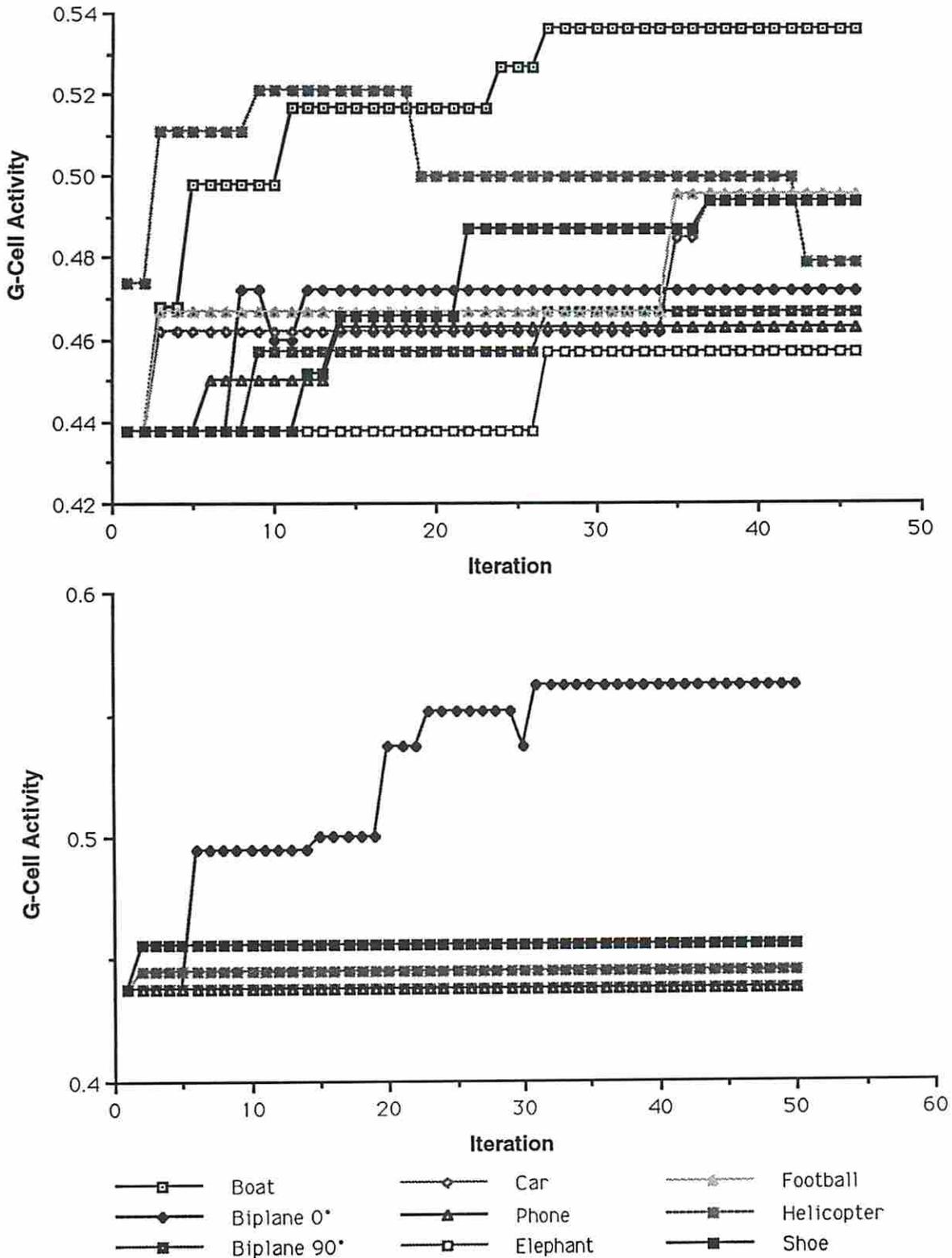


Figure 5.31 - Plots of g-cell activity for the boat and biplane 0° models. G-cell activity is an indicator as to the degree of graph match or recall. The upper plot corresponds to the boat and the lower to the biplane.

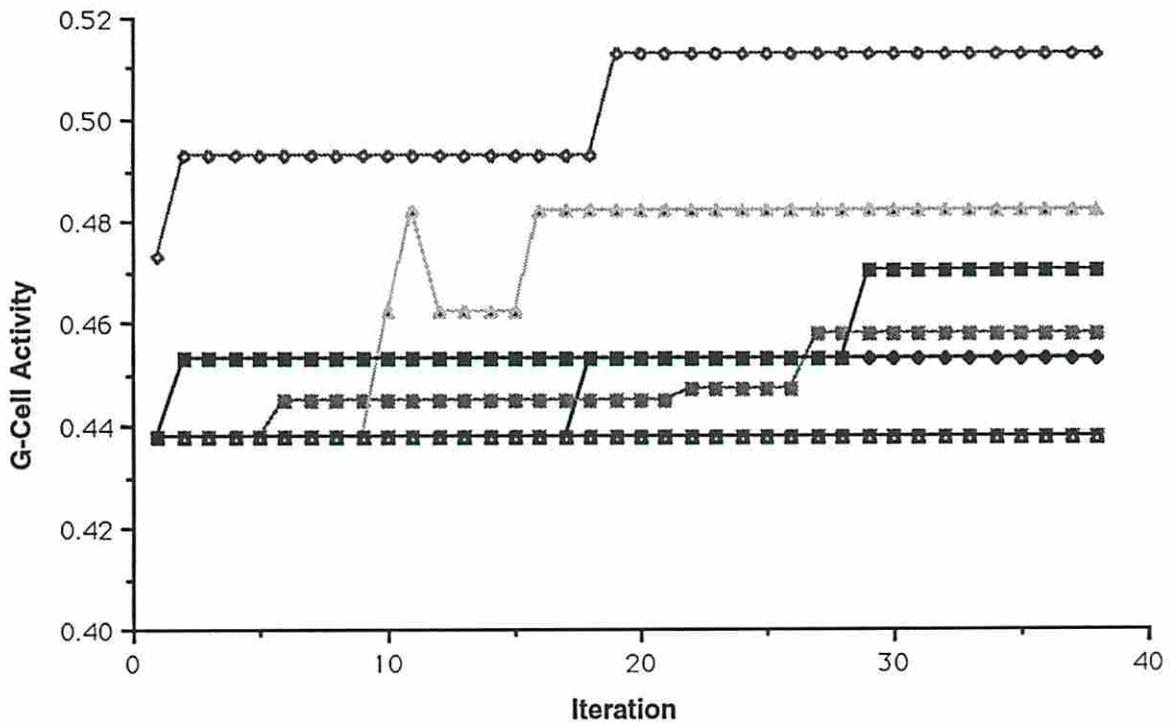
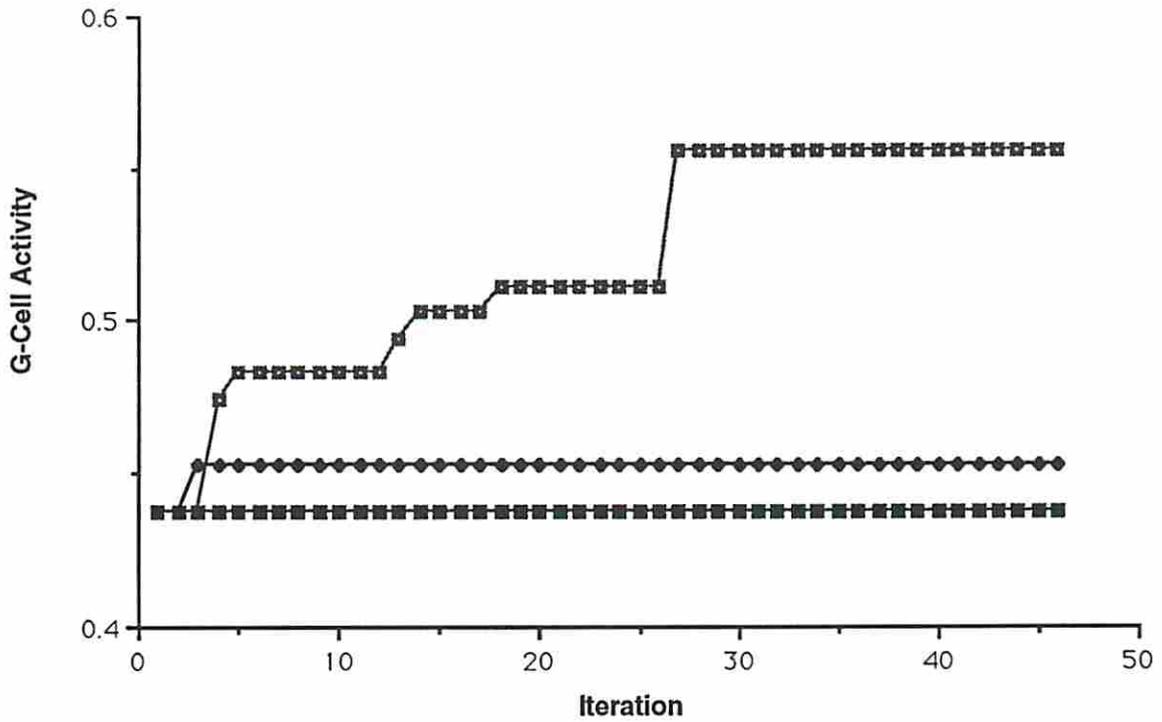
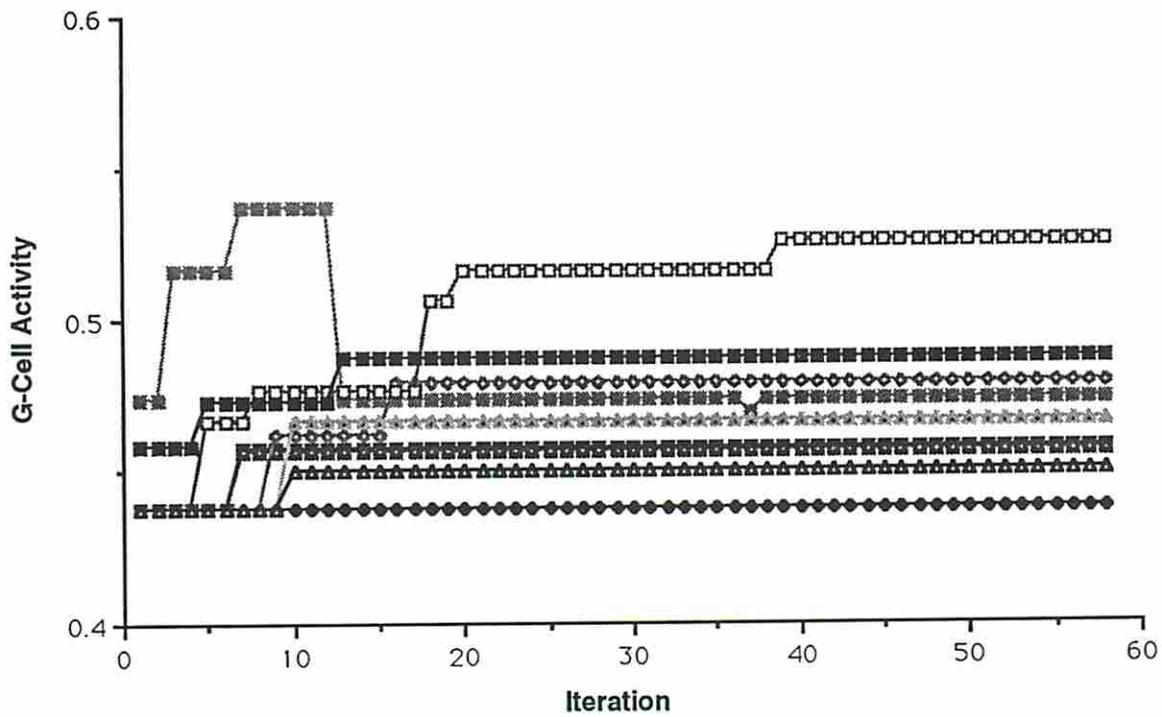
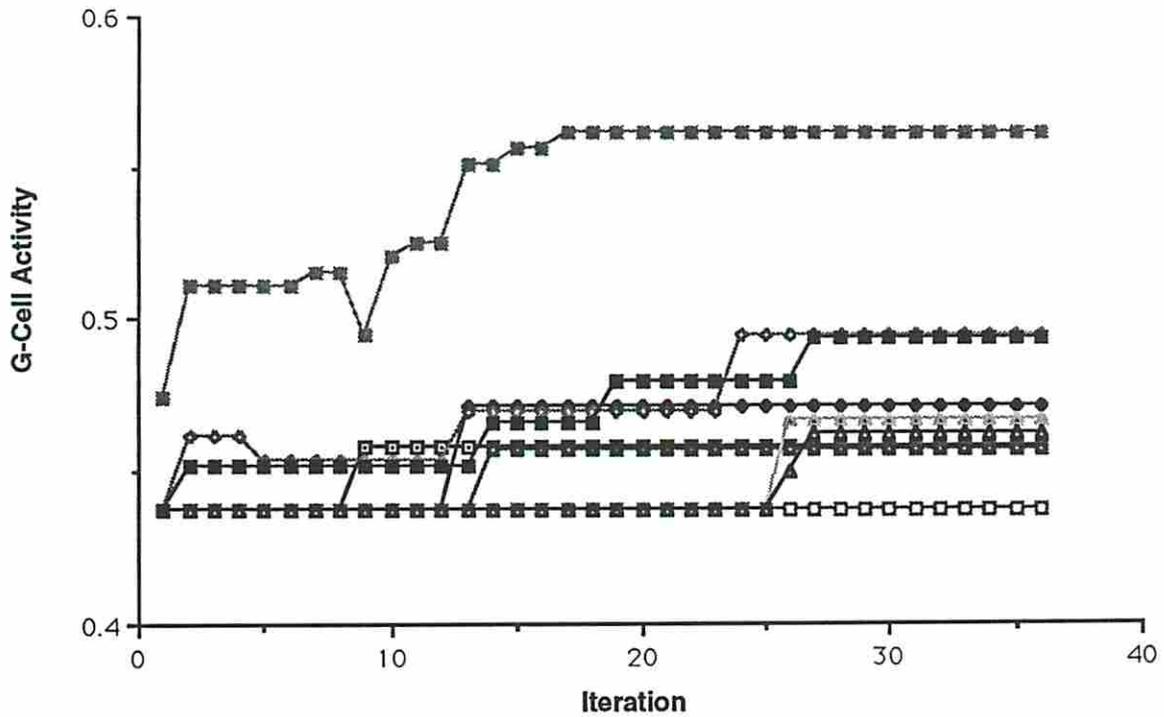
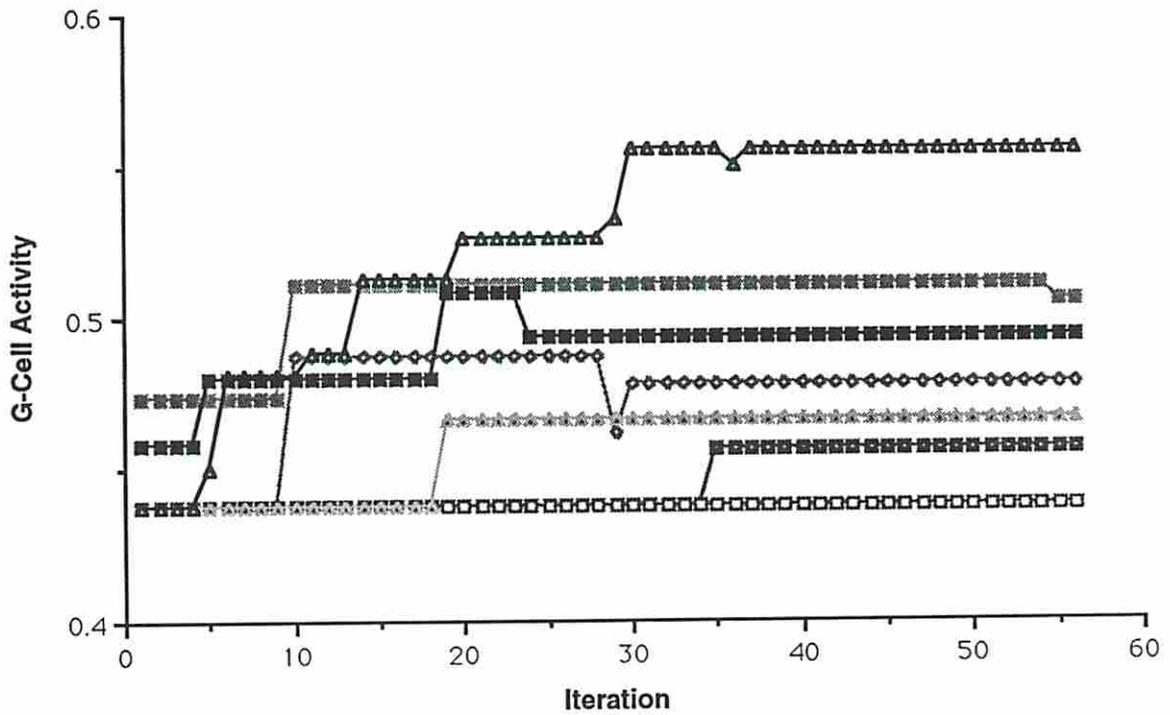
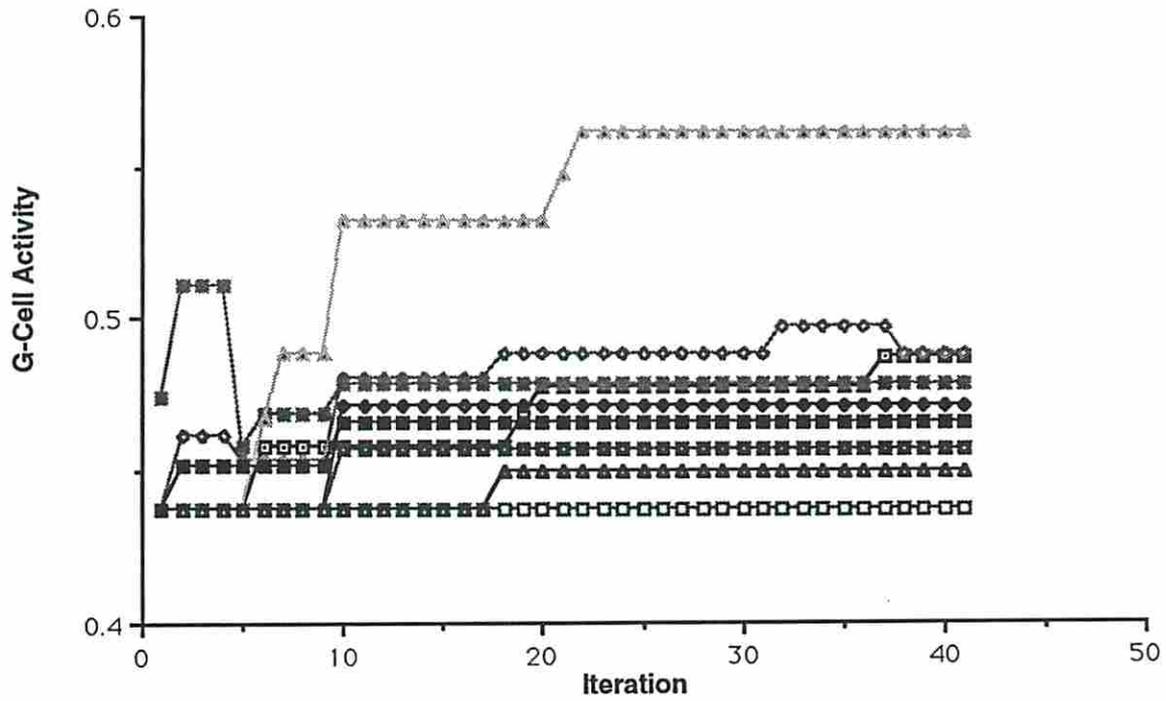


Figure 5.32 - Plots of g-cell activity for the biplane 90° and car models. The upper plot corresponds to the model biplane 90° and the lower to the model car.



- |     |             |     |          |     |            |
|-----|-------------|-----|----------|-----|------------|
| —□— | Boat        | —◇— | Car      | —★— | Football   |
| —●— | Biplane 0°  | —▲— | Phone    | —■— | Helicopter |
| —■— | Biplane 90° | —□— | Elephant | —■— | Shoe       |

Figure 5.33 - Plots of g-cell activity for the helicopter and elephant models. The upper plot corresponds to the model helicopter and the lower to the model elephant.



- |     |             |     |          |     |            |
|-----|-------------|-----|----------|-----|------------|
| —□— | Boat        | —◇— | Car      | —*— | Football   |
| —◆— | Biplane 0°  | —△— | Phone    | —■— | Helicopter |
| —■— | Biplane 90° | —□— | Elephant | —■— | Shoe       |

Figure 5.34 - Plots of g-cell activity for the football and phone models. The upper plot corresponds to the model football and the lower to the model phone.

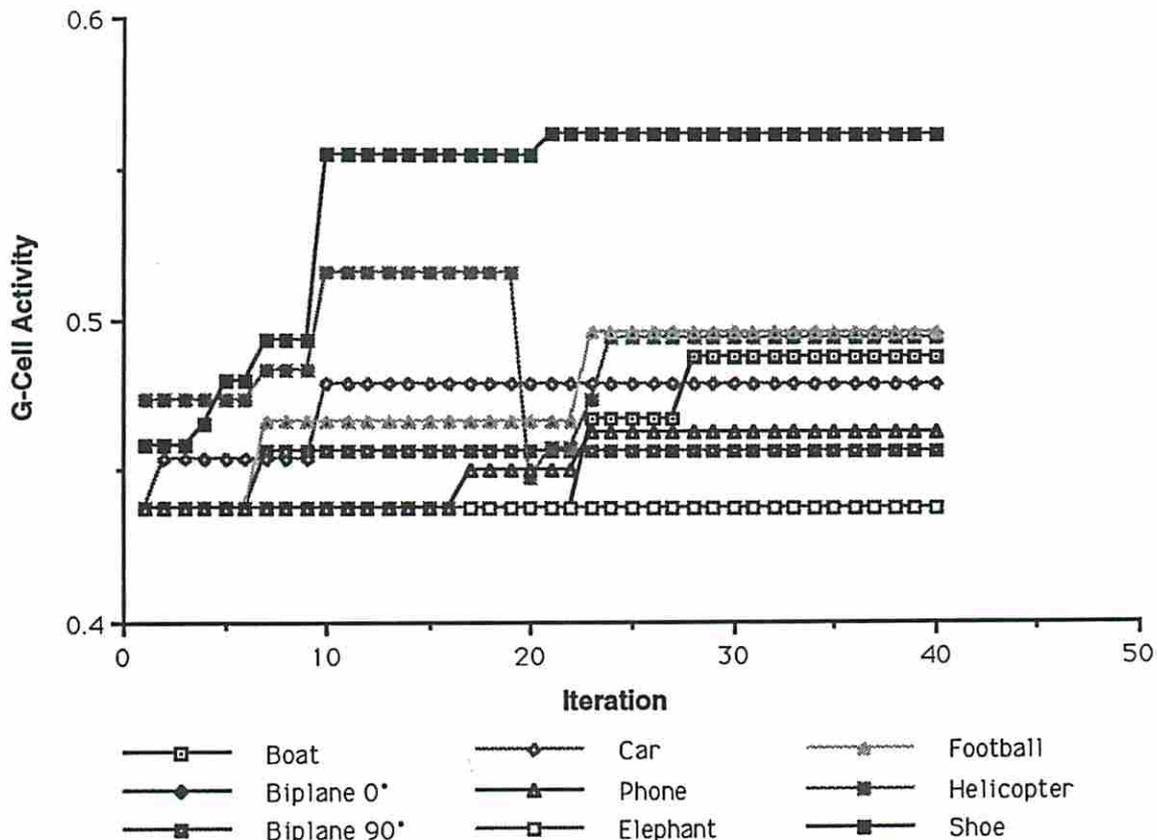


Figure 5.35 - Plot of g-cell activity for the model shoe.

The next system test demonstrated the ability of the system to recognize an object at varying aspect angle. The object chosen was the biplane which had 0° and 90° aspect models already stored in the database. The system was presented with 19 views of the biplane from 0° to 90° in steps of 5°. Figure 5.36 shows the 19 biplane images and Figure 5.37 shows the response of the system to those inputs. Note that the system correctly recognizes the input as biplane 0° for aspects 0° through 25° and as biplane 90° for aspects 60° through 90°. Between aspects of 25° and 60°, the system is no longer consistently accurate although responses of biplane 0° and biplane 90° dominate.

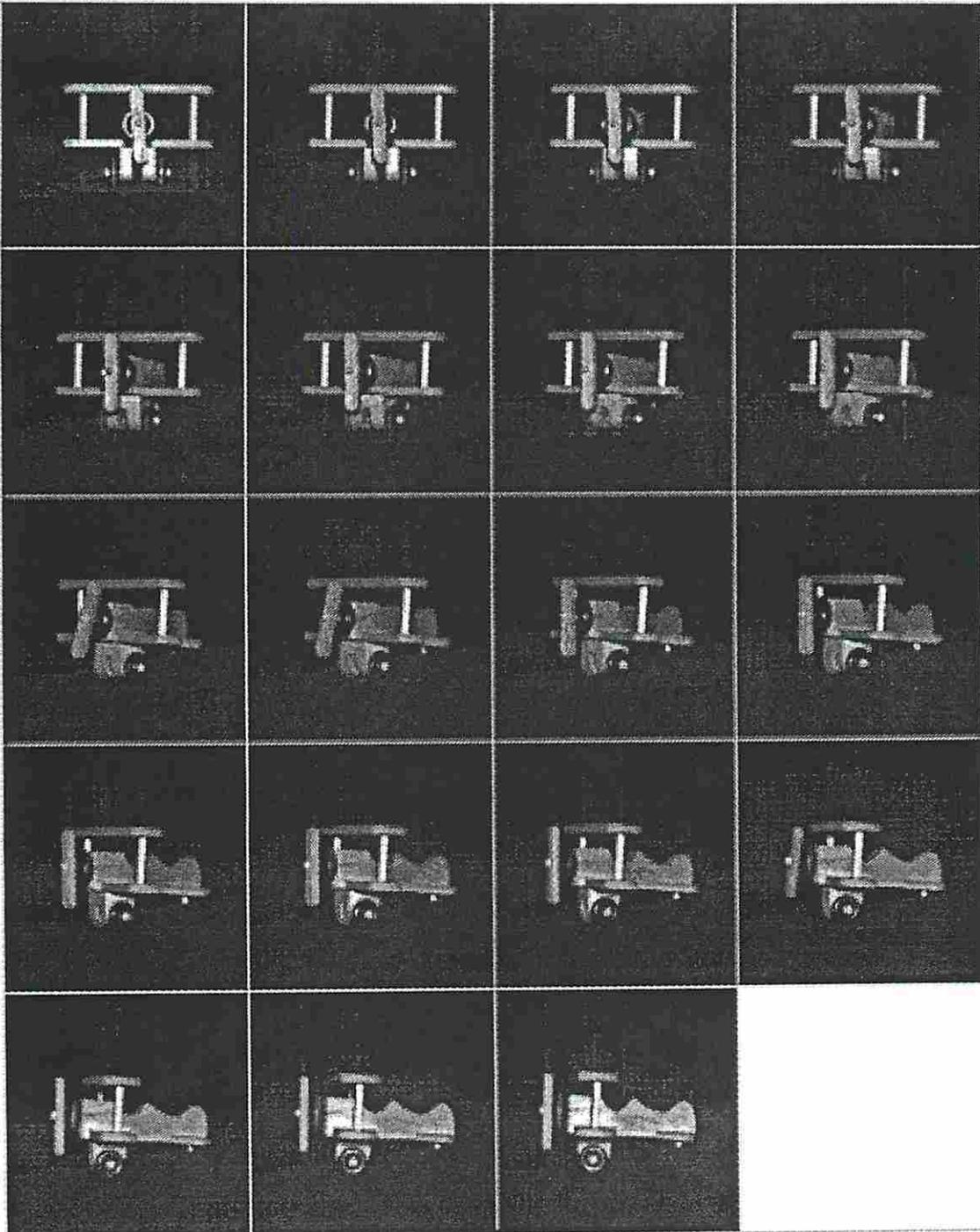


Figure 5.36 - Aspect test inputs: toy biplanes from 0° to 90° in increments of 5°.

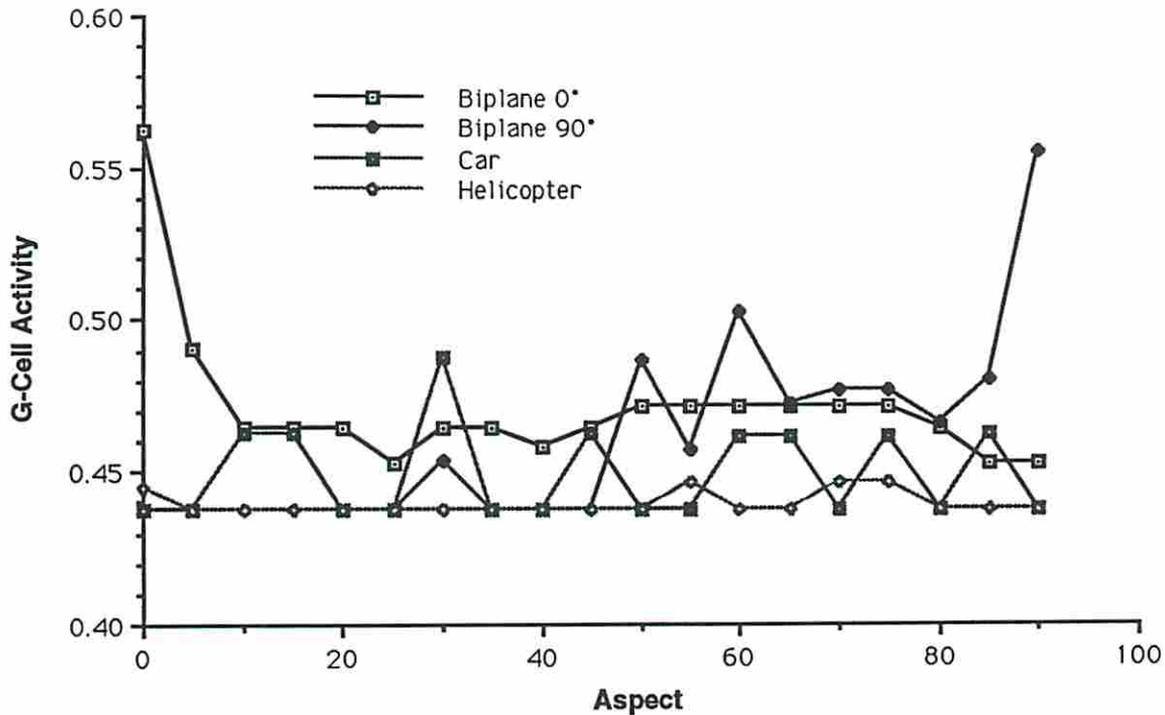


Figure 5.37 - System response to aspect inputs of Figure 5.36. The system correctly recognizes the inputs for 25-30° aspect variances and misses only occasionally elsewhere in the 90° interval. (Responses for five of the model objects are not shown here as none of those five showed significant activity in this test.)

The third system test involved various distortions on the input objects: scale, in-plane rotation, lighting direction, partial occlusion, and additional aspect changes. Figure 5.38 shows scale, rotation, lighting, and aspect variations on the biplane 90° model and aspect variations, only, on the car model. Figures 5.39 through 5.42 show the system responses to those inputs. In each instance, with one exception, the system recalls the correct model. The failure of the rotation case was expected and is due to the poor rotational invariance of the features. It should also be noted that the failure was not a spectacular one: the second strongest response for the rotated input corresponds to the correct model. Figures 5.43 through 5.45 show four partially occluded helicopters and the corresponding system responses. In each occluded case, the system easily recalls the model helicopter.

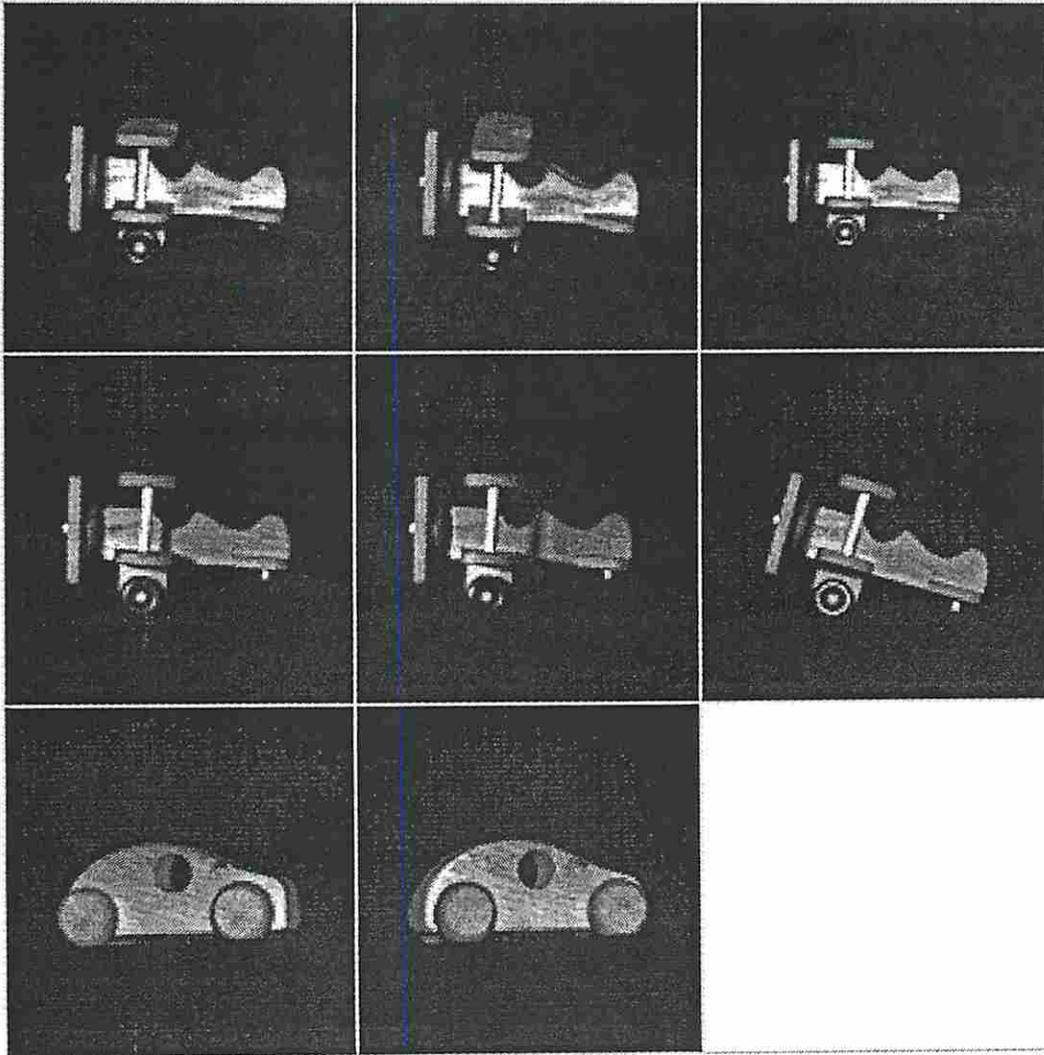
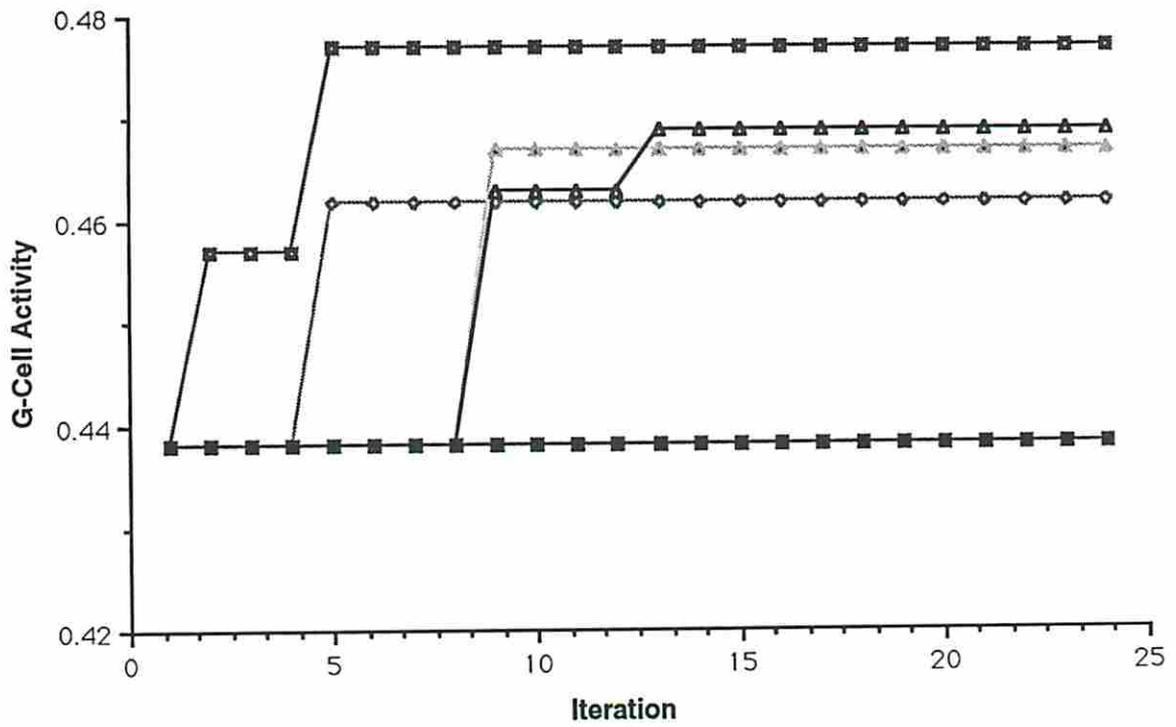
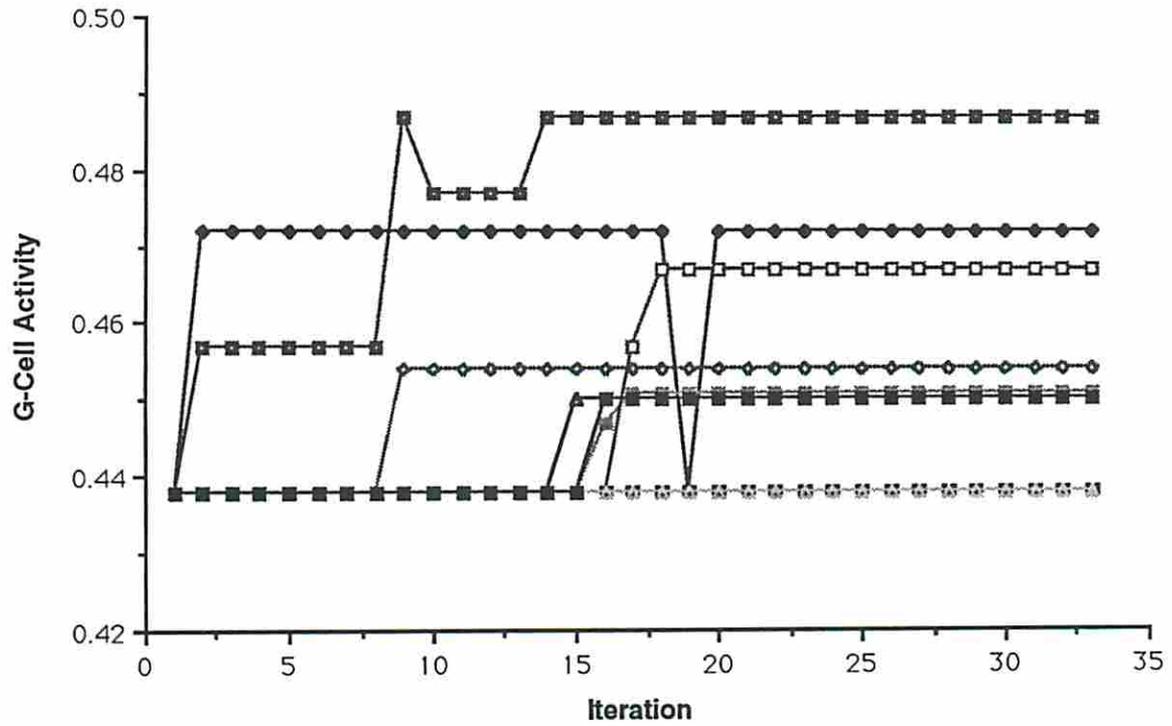


Figure 5.38 - Test inputs distorted by (a) elevation aspect of  $15^\circ$ , (b) elevation aspect of  $30^\circ$ , (c) scale reduction of 25%, (d) lighting direction shift of  $15^\circ$ , (e) lighting direction shift of  $30^\circ$ , (f) in-plane rotation of  $17^\circ$ , (g) azimuth aspect of  $80^\circ$  ( $-10^\circ$  relative), and (h) azimuth aspect of  $100^\circ$  ( $+10^\circ$  relative).





- |     |             |     |          |     |            |
|-----|-------------|-----|----------|-----|------------|
| —□— | Boat        | —◇— | Car      | —★— | Football   |
| —●— | Biplane 0°  | —▲— | Phone    | —■— | Helicopter |
| —■— | Biplane 90° | —□— | Elephant | —■— | Shoe       |

Figure 5.40 - Biplane match results under variable lighting direction. Input objects are the distorted biplanes with lighting direction of 15° (top) and with lighting direction of 30° (bottom).

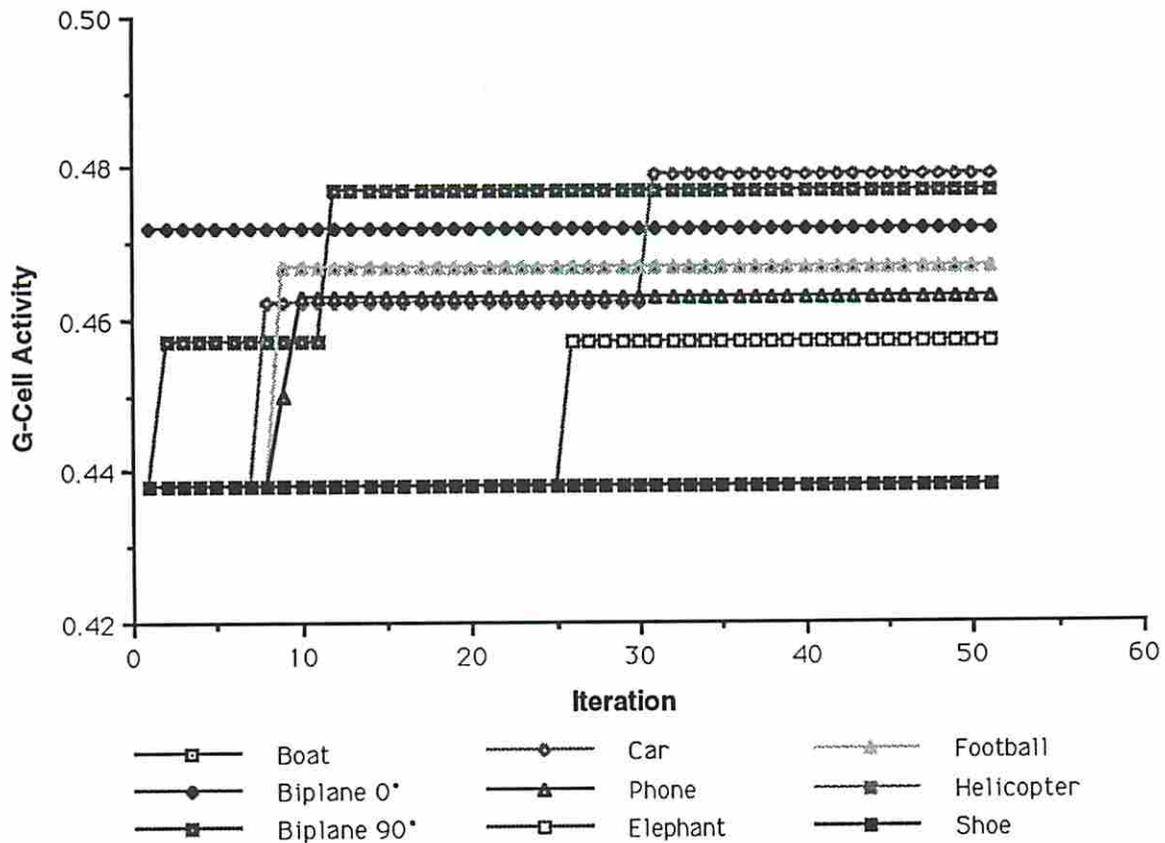
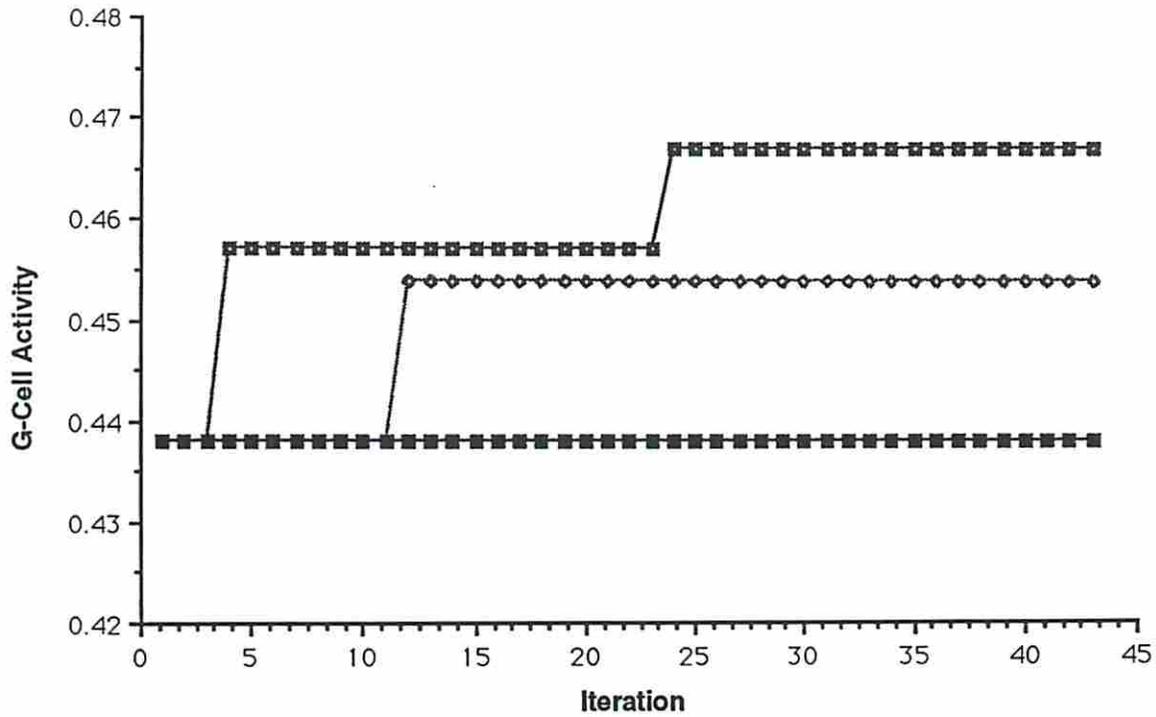
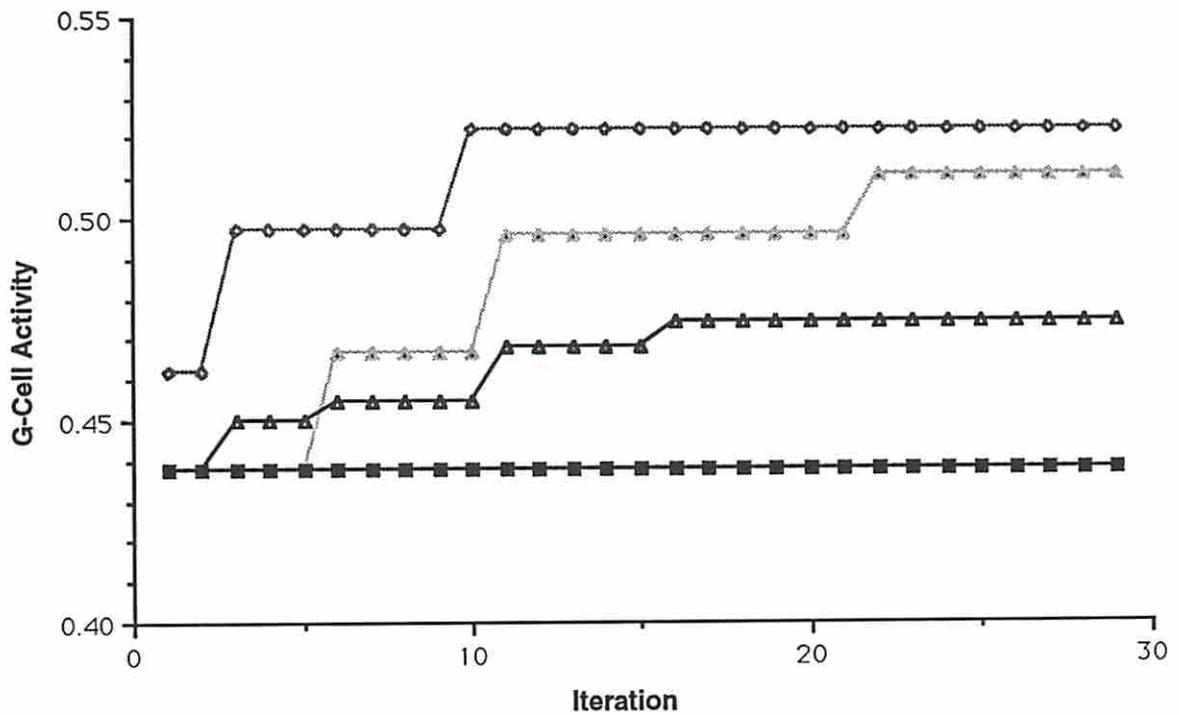
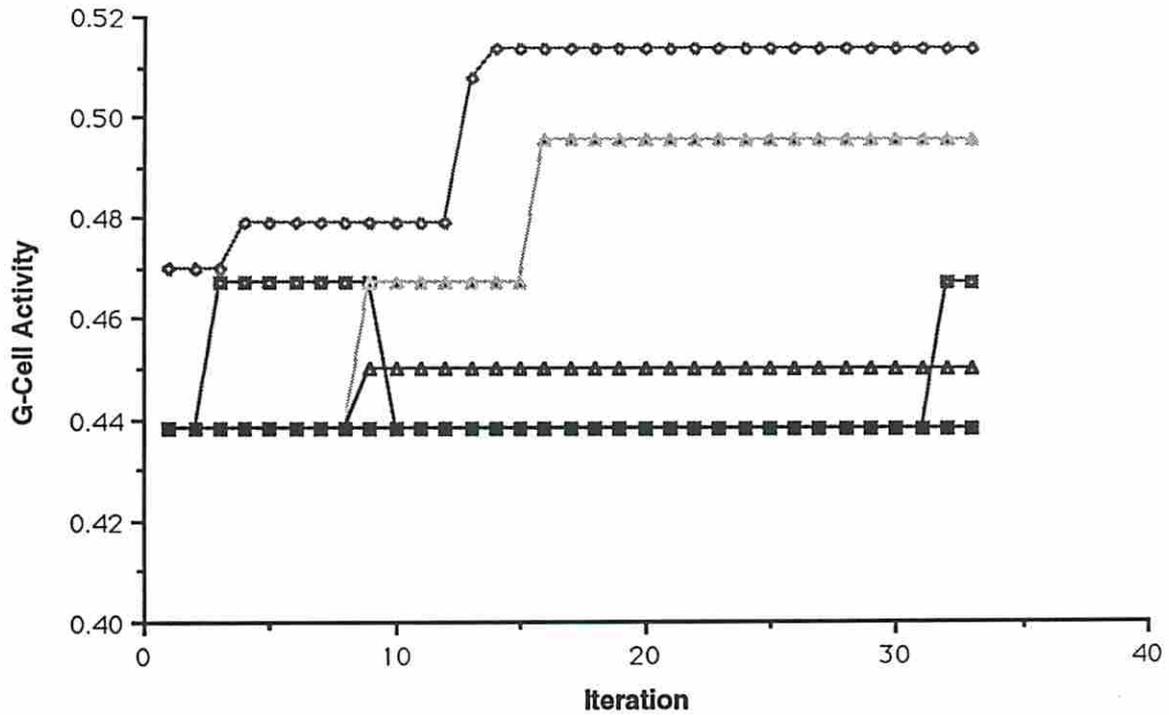


Figure 5.41 - Biplane match results under scale and rotation distortion. Input objects are the distorted biplanes with scale reduction of 25% (top) and with an in-plane rotation of 17° (bottom).



- |     |             |     |          |     |            |
|-----|-------------|-----|----------|-----|------------|
| —■— | Boat        | —◇— | Car      | —△— | Football   |
| —●— | Biplane 0°  | —▲— | Phone    | —■— | Helicopter |
| —■— | Biplane 90° | —□— | Elephant | —■— | Shoe       |

Figure 5.42 - Car match results for perspective distortion. Input objects are distorted cars with azimuth aspect of 80° (top) and with azimuth aspect of 100° (bottom).

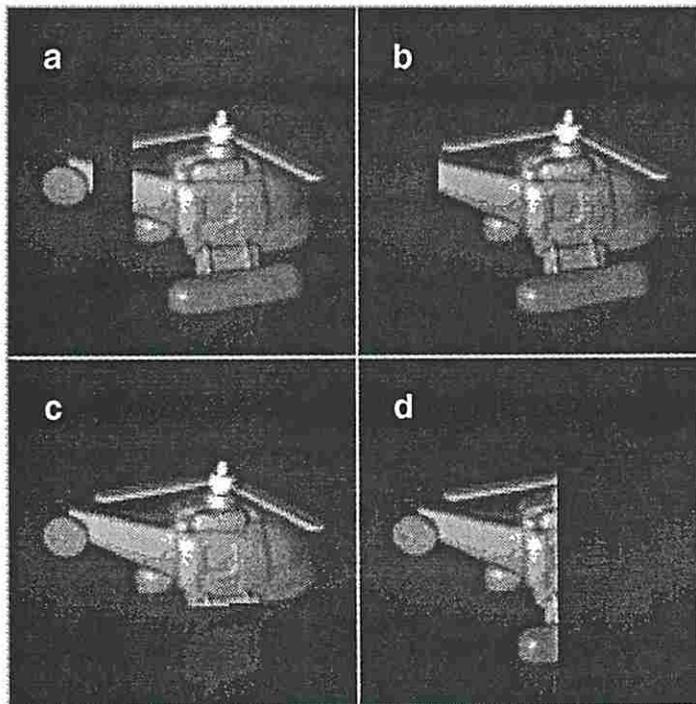
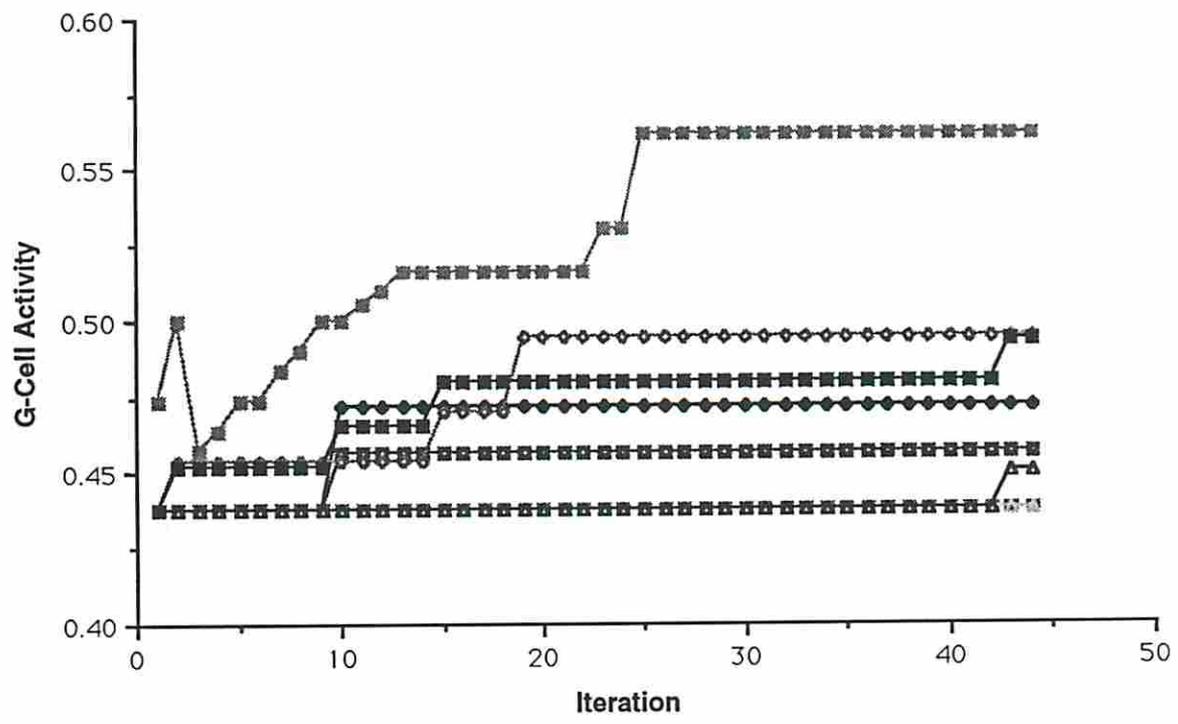
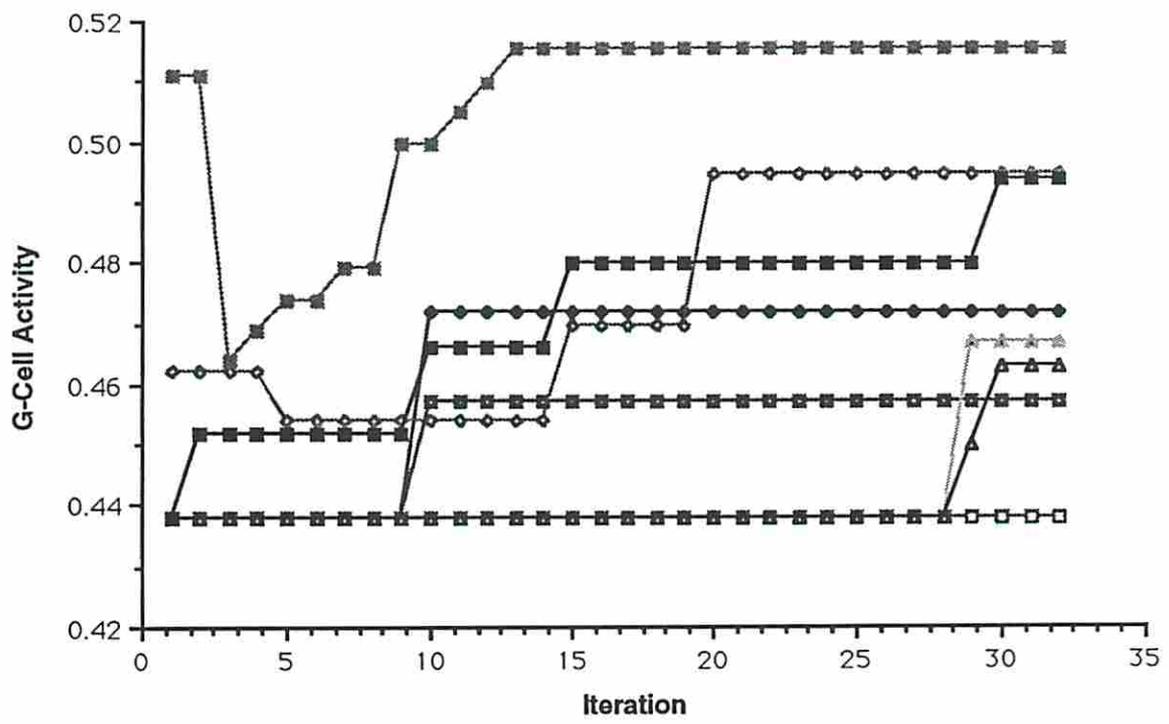


Figure 5.43 - Test inputs of helicopters distorted by partial occlusion.



- |     |             |     |          |     |            |
|-----|-------------|-----|----------|-----|------------|
| —□— | Boat        | —◇— | Car      | —☆— | Football   |
| —●— | Biplane 0°  | —△— | Phone    | —■— | Helicopter |
| —■— | Biplane 90° | —□— | Elephant | —■— | Shoe       |

Figure 5.44 - Helicopter match results under partial occlusion. Input objects are the helicopters of Figure 5.43a (top) and Figure 5.43b (bottom).

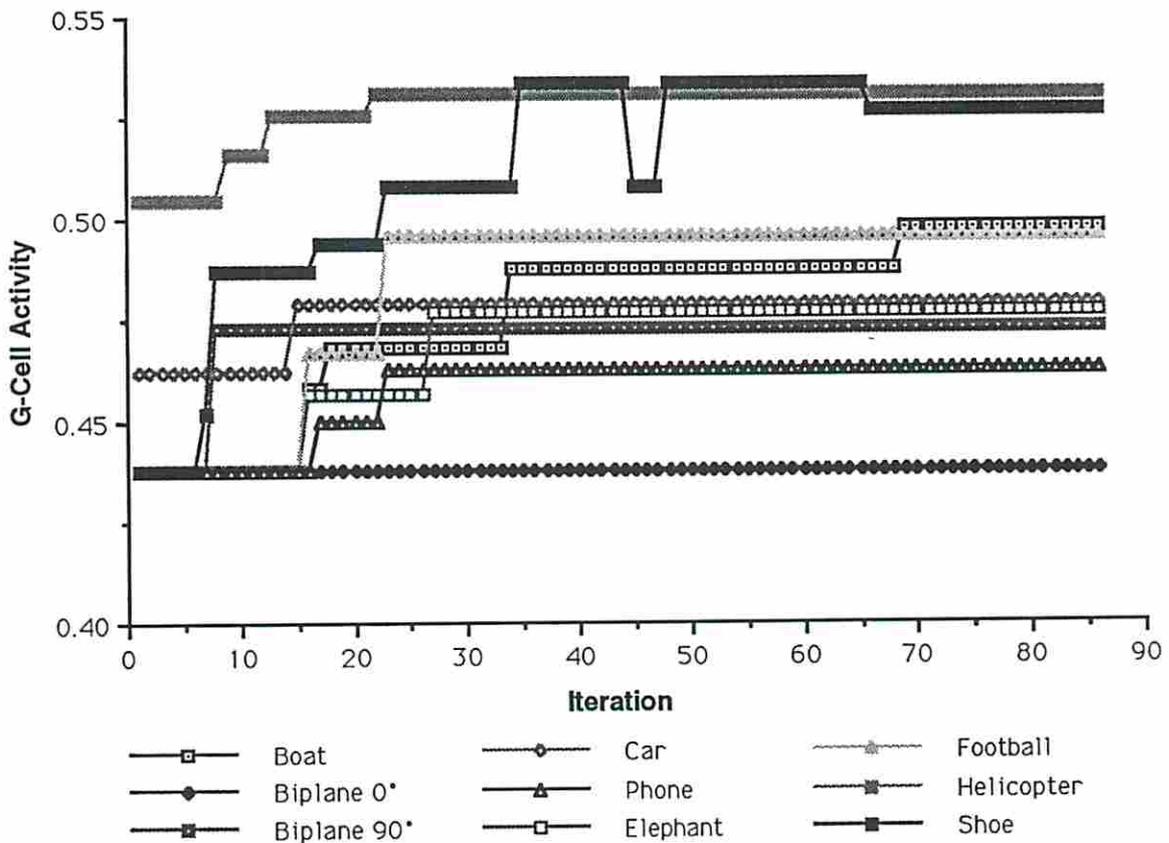
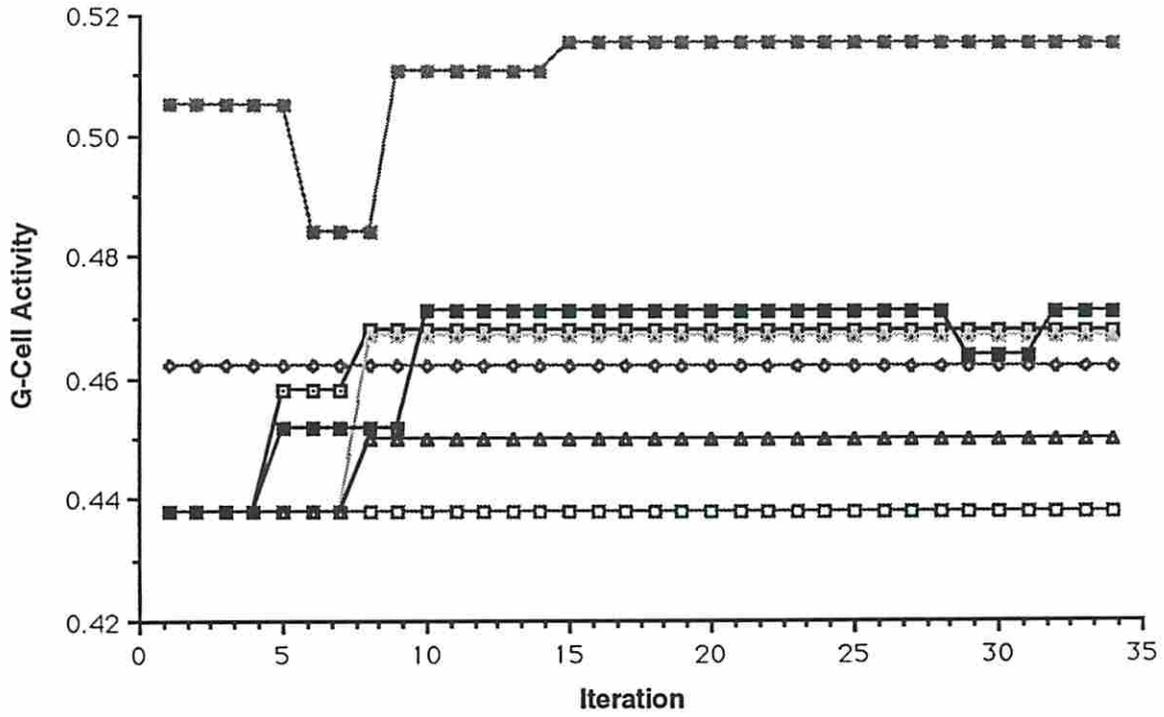


Figure 5.45 - Helicopter match results under partial occlusion. Input objects are the helicopters of Figure 5.43c (top) and Figure 5.43d (bottom).

The final system test looked at the effects of non-zero mixing ratios. The relative mixing factor for the model plane was set to 1.0, i.e., all shareable nodes are shared by all graphs that can share them. Because not all graphs contain the same exact sets of features, the absolute pattern mixing factors are expected to be less than 100%. This is borne out in Table 5.2 which shows the absolute pattern mixing factor for each model subgraph.

Model	Absolute Pattern Mixing Factor
Boat	0.13
Biplane 0°	0.07
Biplane 90°	0.03
Car	0.10
Helicopter	0.09
Elephant	0.12
Football	0.08
Phone	0.04
Shoe	0.10
<b>Avg</b>	<b>0.09</b>

Table 5.2 - Absolute pattern mixing factors for the test database when the relative mixing factor is set to 1.0.

The effect of a non-zero mixing factor on the recognition process was noticeable though not catastrophic. All undistorted models, when input to the system, were correctly and easily recognized. When the system was presented with the distorted versions of the models, it usually failed to correctly recognize the input although the correct model subgraph was usually one of the top three candidates for correct recognition.

### 5.3 Summary of Results

We have demonstrated the efficacy of our system in the role of 2D object recognition. Specifically, the system was provided with nine greyscale objects from which a database was formed. When provided as input, the system was able to correctly classify each model object.

In addition, the system was shown to be robust with respect to a variety of distortions and variances. The system demonstrated robustness to perspective by correctly recognizing a test object over at least 25° of aspect angle both in azimuth and elevation. Scale robustness was shown when the system recognized a test object reduced 25% in size. The system correctly recognized four cases of a test object occluded in four different ways, including one in which the obscuration was over 50%. The system performed well under variable illumination conditions recognizing the test object under shifts in lighting direction of 15° and 30°. The system failed to recognize a test object with approximately 17° of in-plane rotation; in this case, the response corresponding to the correct object was the second strongest.

The effect of non-zero mixing factors was explored by forcing the model graph to a 100% relative mixing factor which corresponded to an average absolute pattern mixing factor of 9%. Performance at 100% relative mixing was degraded: the system correctly recognized the undistorted model objects but failed in most distortions.

## Chapter 6 - Conclusion

### 6.1 Summary

We have developed and demonstrated a system that performs object recognition using adaptive feature extraction and dynamical link graph matching. The system was tested with a database of nine objects derived from greyscale images. Undistorted objects and objects distorted by scale, perspective, rotation, partial occlusion, and lighting direction were presented to the system. All were correctly recognized except one. The exception was the rotationally distorted object; in that case, the second strongest response came from the correct model. The system was also tested using a relative mixing factor of 100%. Undistorted object recognition remained completely intact while distorted object recognition was degraded.

The adaptive feature extractor consists of an edge enhancer, a Morlet wavelet decomposition (MWD) module, and an adaptive vector quantization (AVQ) module. The edge enhancer uses a 5x5 Laplacian to bring out edges and reduce variable lighting effects. The MWD uses Gabor functions as (Morlet) wavelets to decompose an input image into  $N^2$  components per kernel. Each Morlet wavelet has a characteristic orientation and spatial frequency; our MWD uses six orientations and five spatial frequencies. Jets (feature vectors of Morlet wavelet components) are used to characterize the image from the perspective of the jet's position. Jet magnitudes are used as a saliency measure where the larger magnitudes indicate greater salience.

The AVQ module operates in two modes: learning and quantization. In the learning mode, a large number of jets are passed to the AVQ module which learns their distribution with a self-organizing vector map. The learning need take place only once or whenever the distribution of the jets has significantly changed. After learning is complete (i.e., after the vector map has stabilized), the second mode, quantization, may be invoked. In vector quantization, the goal is to find the model vector from the vector map that will best represent an input jet. If the system has learned the distribution of the jets and if the input jet fits the distribution, then the selected model vector will be a good approximation of the input jet. Because there are a finite number of model vectors, it is possible to represent the input jet with only a scalar label that corresponds to the best fit model jet. In our system,  $N^2$  jets are passed from the MWD to the vector quantizer which, in turn, creates a feature image of  $N^2$  model vector labels. The feature image is passed to the next processing module which generates a graph.

Two types of graphs are employed: single-object input graphs and multi-object model graphs. The former is used as input to the graph matcher whereas the latter forms the database.

Non-zero mixing factors may be applied to model graphs (i.e., single-object model subgraphs may have shared nodes) and while they result in increased storage capacity, they also lead to a degradation of the recall process.

The graph formation module converts a feature image into a labeled and locally connected graph. Features from the feature image are selected as node labels based on their saliency, their label accuracy (quantization error), and their continuity (similarity to neighbors). In addition, a minimum distance constraint is applied that requires a minimum separation between candidate nodes. After the node labels have been selected, they are connected according to maximum distance, minimum number of neighbors, and maximum number of neighbors constraints. Because the graphs must possess some degree of robustness to scale, the distance constraints are made a function of the size of the object. Completed graphs are passed to the graph matcher for comparison with model subgraphs and thus recognition may occur.

The graph matcher is based on Dynamical Link Architecture and consists of three neural planes with various intra- and inter-connections. Graphs are implemented as neural networks in which the labeled nodes are artificial neurons and the edges are synapses. The input plane holds the input graph, the model plane holds the model (database) graph, and the grandmother plane holds grandmother cells. Grandmother cells, or g-cells, signal the degree of match for each stored pattern. Two types of links are employed. J-links are the signal carrying connections commonly associated with the synaptic function but are variable during non-learning operations, i.e., J-links are dynamical links. For each J-link, there is a T-link. T-links provide an upper bound on their respective J-links. In our system, only the connections within the model plane are dynamic; all other links are fixed at the T-link value.

The input plane neurons are connected to model plane neurons such that only those neurons with identical or highly similar labels are joined. Model plane neurons are connected to g-cells such that all the nodes of a single-object model subgraph are joined to a unique g-cell; each g-cell, therefore, signals the combined activation of a single model pattern.

The recall procedure is a two phase operation: activation and readout. The activation phase begins with the external activation of a small ensemble of input plane nodes. These nodes transmit their activations to similarly labeled nodes in the model plane. Connected nodes in the model plane that are simultaneously active have their J-links strengthened thereby binding them together more tightly. Conversely, connected model plane nodes that are of opposite polarity (i.e., one active and one inactive) have their dynamical links weakened thereby disconnecting them. This is the essence of the graph matching solution: subgraphs of model plane nodes that match subgraphs of input plane nodes are bound together whereas nodes that do not fit the active input plane ensembles are disconnected.

The second phase, the readout phase, occurs immediately after each activation phase. The objective of this phase is to measure the degree of binding that has taken place in each model subgraph and thus provide information as to the degree of the match. The readout phase begins with the full activation of the input plane and hence a substantial activation of the model plane. After activity in the model plane has stabilized, the input plane is deactivated. Because model plane interconnections are symmetric, there is the possibility of model plane nodes exciting each other in a stable manner. This possibility is only a reality if the J-links have reached a sufficient strength so as to allow input activations to exceed the neural threshold. After

activations have stabilized again (without input plane activation), those model plane nodes that are active provide indications as to the degree of binding that has occurred. The model plane nodes transmit their activations to their respective pattern g-cells which respond as a continuous function of their inputs. Monitoring g-cell activity, then, provides an indication as to the degree of match for each pattern.

The next cycle begins by restarting the activation phase; a new input plane ensemble is activated and the dynamics are repeated. The process continues in this cyclic fashion until g-cell activity is determined to have converged at which point the g-cells may be read and the winner declared.

The entire system may be implemented neurally. Neural edge enhancers currently exist in the form of regularization nets [Poggio, Torre, and Koch, 1985] and boundary contour systems [Grossberg and Mingolla, 1987]. The Morlet wavelet decomposition may be implemented with neurons possessing Gabor-shaped receptor fields, similar in manner to Nature's implementation [Jones and Palmer, 1987]. Orientation column tuning, fashioned after the description by [Hubel and Wiesel, 1974] may be implemented by competitive connections. Self-organizing vectors maps have been described in neural terms by [Willshaw and von der Malsburg, 1976] and [Kohonen, 1988]. Neural-based graph generation has been described by [Reiser, 1991]. The DyLink graph matcher is currently implemented as a neural network.

## 6.2 Observations, Limitations, and Extensions

It is probably the case that, for most systems developed for degree purposes, there are more limitations and possible extensions than there are accomplishments. It is often the purpose of such systems to provide proofs-of-concept rather than to function in ambitious real world environments. It is our belief that we have developed a system of the former variety and that further development would be necessary for it to be categorized as one of the latter. It is with that in mind that we itemize some of the current limitations and possible solutions.

### 6.2.1 Multiview Recognition

Perhaps the most obvious requirement for a more generalized visual recognition system is that it be able to recognize objects from a wide range of aspects. We have demonstrated an ability along those lines by using multiple stored views of the same object and showing robustness to perspective. While the demonstration was not intended to prove 3D object recognition, it points to the ability of the system to achieve that goal given certain improvements. There are two ways in which the system can be made responsive to 3D. The first is as was demonstrated in Chapter 5, i.e., by using multiple object graphs for the same object. This approach is, however, highly wasteful of computational resources in that object graphs from similar views will contain similar subgraphs thereby requiring duplication of those subgraphs for each object graph. This brute force approach may work well in limited domains with small numbers of objects but is not expected to scale well. The preferred method would be to use a single 3D

aspect graph for each object. Of specific value to the system would be the methodology employed by [Reiser, 1991] in which 3D graph representations are built by incorporating only those features that remain stable over perspective. Such a methodology would be a welcomed replacement of the current graph generation process which uses only indirect measures of stability and is non-neural. The use of 3D aspect model graphs would, unfortunately, require modifications to the readout architecture and dynamics of the current graph matcher. These modifications are expected to be minor. Their purpose would be to require readout values to be based on contiguous subgraph activation and not multiple disjunct activations throughout the graph.

### 6.2.2 Negative Recognition

Another observation regarding the system is that it does not explicitly supply a negative recognition response. Instead, the system indicates the strength of the matching process and provides relative information regarding the ordering of candidates. We do not consider this behavior to be a limitation; indeed, it shows a measure of flexibility. Simple implementations of the system may threshold g-cell outputs thus providing a capacity for negative recognition results. More complex systems may use our system as a lower-level element in which results from several g-cells are combined to produce an inferred decision.

### 6.2.3 Expectation Driven Perception

As currently implemented, the g-cells function only in a feedforward capacity, monitoring the strength of the  $J^M$ -links. The originally proposed architecture, however, provided for the existence of connections from g-cells back to model plane neurons, i.e.,  $J^{MG}$ -links. Such connections were actually implemented early in the research but were subsequently disabled to reduce the scope of the analysis. The  $J^{MG}$ -links, augmented with weak winner-take-all  $J^G$ -links, were designed to improve the speed of recognition. The idea is that as one model-subgraph became dominant, its g-cell would become stronger than the others. The stronger g-cell would inhibit other g-cells while, at the same time, feeding its excitation back to its model-subgraph. (Early experiments did, in fact, bear this dynamic out.) If the g-cell's were properly biased, the losing g-cells would feedback inhibition to their model-subgraphs and the system would declare, rather quickly, a winner.

One very powerful side-effect of such a scheme (sans competitive  $J^G$ -links) is that it allows top-down input to guide the matching dynamics. Top-down control would function in the following manner. An input object is applied to the system. Based on other, perhaps contextual, information a higher level process determines that there is some likelihood of the object being a widget. That higher level process activates the g-cell (corresponding to widget) commensurate with the degree of likelihood. The g-cell, through the  $J^{MG}$ -links, provides some degree of activation to its model graph thereby biasing the system in the direction of recognizing a widget. With such a top-down influence, the system would demonstrate expectation-driven perception [Arbib, 1989], that is, the system would tend to recognize what it expected to see.

## 6.2.4 Implementation Issues

Our work is implemented on a Macintosh IIfx. The processing rate is currently limited by the Morlet wavelet decomposition which takes approximately 30 minutes per image at six orientations, five spatial frequencies, and with zero-padding enabled. All other processing is on the order of seconds. (AVQ training is performed off-line and is effectively accomplished in less than 20,000 iterations taking approximately 30 minutes.) Obviously, such an implementation for a real-world application is unacceptable. One of the benefits of the neural approach, however, lies in its parallelism and thus in its implied speed. MWD has been demonstrated at approximately frame-rate speeds by [Anderson, 1990] using a DataCube. In addition, optical implementations of Morlet wavelet transforms are feasible [Hughes, 1989]; such computations are limited only by the speed of the detector array readout device and is on the order of microseconds. As noted earlier, the self-organizing vector map is inherently a neural network: each vector in the vector map is represented by the synaptic weights of a single neuron. Such a system, therefore, performs vector quantization in parallel. We have used a 15x15 vector map in our system; off-the-shelf hardware necessary to efficiently implement that approach currently exists. The graph generation module, while not a neural paradigm, is highly amenable to both parallel and distributed processing. The graph matcher, being a neural paradigm, is also highly parallelizable and amenable to distributed processing. We therefore conclude that a fast, perhaps real-time implementation of our system is feasible.

## 6.2.5 Scale and Rotation Robustness

Chapter 5 demonstrated a wide variety of robustness to distorted input. Lacking, both in theoretic and empirical terms, however, is robustness to rotation. In addition, we emphasize here that the system only possesses a certain degree of robustness to scale and is not invariant<sup>17</sup>. Both of these limitations are due exclusively to the fact that the low-level features, i.e., the Morlet jets, are intentionally sensitive to rotation and spatial frequency. In the case of scale, because each wavelet has a finite bandwidth, a feature will move in and out of a filter's frequency range as its size changes [Buhmann, Lades, and von der Malsburg, 1990]. Consider the jet image representation of Figure A.1b (Appendix A). Buhmann et al. showed that a variation in scale results in a shift of component values along the spatial frequency axis. Thus, size shifted features may be represented by similar jets with the only difference being a shift in spatial frequency. Similarly, a change in rotation results in a shift along the orientation axis. A jet representing a rotated feature, then, contains the same information but in a lexicographically different arrangement.

The system need not suffer from scale and rotational dysfunction. A modification to the system would employ multiple recognition iterations after the MWD using mutated vector arrangements to determine the best rotational and spatial frequency positionings. Such a scheme would demonstrate the same linear-time de-rotation effects found in the human visual

---

<sup>17</sup> Indeed, the system has demonstrated failure in the case of a 50% scale reduction of the Biplane 90° object of Figure 5.22.

system [Shepard and Metzler, 1971]. A faster alternative would be to employ parallel implementations of the post-MWD system in a genetic-algorithm sort of approach.

## 6.2.6 Computational Complexity

### 6.2.6.1 Space Complexity

Data densities vary in our system from high density in the earliest processing stages to lower densities in the final stages. See Figure 6.1. Specifically, if the input image is  $n \times n$  ( $x$  8 bits) the Morlet wavelet decomposition returns a data structure that is  $n \times n \times S \times O \times 2$  ( $x$   $b$  bits) where  $S$  and  $O$  are the number of spatial frequencies and orientations used by the decomposition (4 and 6, respectively) and the factor of two is due to the use of complex numbers. For our work,  $n=128$  and  $b=32$  which corresponds to 4-byte floating point values. ([Anderson, 1990] has demonstrated 8-bit integer operation in his system but it is unclear to us at this point whether that result is relevant to our application.)

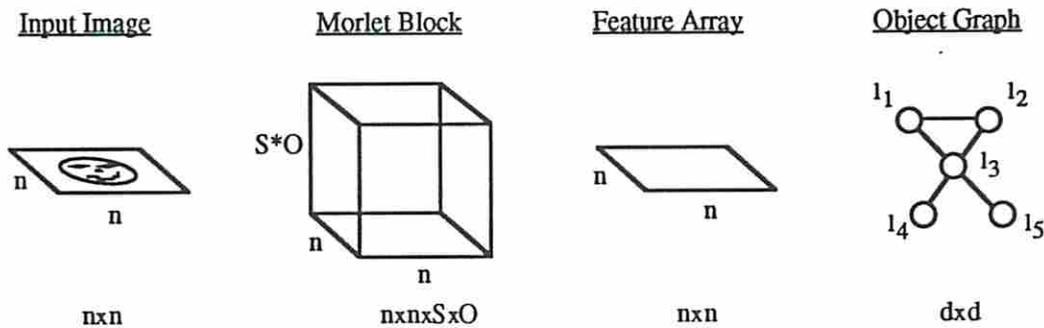


Figure 6.1 - Data densities for the system as implemented. The Morlet block and the feature array scale as a constant function on the image size. The highest complexity data structure for the graph representation is the connection matrix which scales  $O(d^2)$  where  $d$  is the number of nodes in the graph.

The vector quantization stage uses an  $m \times m$  ( $15 \times 15$ ) model vector map to quantize the  $S \times O \times 2$  dimensional jets produced by the MWD. The resulting data array of  $n \times n$  feature labels is currently implemented using 2-byte integers, however, two compatible data reduction schemes are possible. In the first scheme we may take advantage of the previously computed feature saliencies and form features only as specified by the saliency-driven portion of the graph generation module. If that processing module typically produces on the order of  $d$  nodes per graph and has a  $\Gamma$ -array radius of  $r_\Gamma$ , then the AVQ module may be expected to produce only  $d \cdot \pi \cdot r_\Gamma^2$  feature labels instead of  $n^2$ . For our work,  $d$  was typically 15 and  $r_\Gamma$  was arbitrarily set to 5 thus the system, using this scheme, would typically produce 1178 labels instead of 16384.

The second scheme would losslessly compress the feature labels using Huffman or a similar method. As results in Chapter 5 indicate, specifically Figure 5.27, the jet distributions are not smooth, i.e., they have high entropy; the system would therefore benefit from such a compaction approach.

The final object representation is a graph consisting of typically  $d$  labeled nodes each with approximately  $e$  edges. Our implementation of a graph uses a  $d$  dimensional vector of labels, a  $d$  dimensional vector of relative saliencies, and a  $d \times d$  connection matrix. The model plane connection matrix, which grows as more models are added as  $O(d^2)$  on the number of nodes, is sparsely populated and might benefit from more compact representations (e.g., linked lists). It is not clear to us at what point the space-time tradeoff would make such compactifications viable but it is expected to be a function of the computational platform as well as the implementation.

### 6.2.6.2 Time Complexity

Characterizing the early processing modules in terms of their computational complexity is relatively straightforward. As noted in Chapter 2, the wavelet decomposition occurs with the use of a fast Fourier transform that is  $O(n^2 \log_2 n)$  on the image size. Jet saliency requires  $O(n^2 * S * O)$  operations per image. Vector quantization requires that an inner product be computed between an input vector and each of  $m^2$  model vectors where all vectors are of dimension  $S * O$ ; the number of operations for vector quantization is thus bounded by  $O(n^2 * m^2 * S * O)$ . Section 6.2.4 describes ways in which these algorithms may be parallelized, thereby reducing the temporal complexity at the expense of increased processing requirements.

The graph generation module computes feature stability and uses computations already made in earlier processing stages for jet saliency and feature accuracy. Feature stability (Equation 3.1) is measured by jet continuity and requires, at most,  $O(n^2 * r_c^2 * S * O)$  operations per image where  $r_c$  is the small circular radius of the neighborhood,  $\mathfrak{N}_{ij}$ . As suggested by Section 6.2.6.1, it is possible to reduce the complexity of this operation by computing Equation 3.1 only in the neighborhood of salient features. The resulting complexity is  $O(d * r_{\Gamma}^2 * r_c^2 * S * O)$  operations where  $d * r_{\Gamma}^2 \ll n^2$ .

While subgraph isomorphism (SI) is known to be in the class of NP-complete, the graph matching module of our system is not a solution to that problem<sup>18</sup>. Although we have not attempted to explicitly ascertain the computational complexity of the graph matcher, our experience is that it does not exhibit non-polynomial behavior. The discussion that follows does not contain a rigorous proof of convergence<sup>19</sup> but generally describes our experience in that area and how effects such as gallery scaling and graph mixing may affect convergence.

From Figure 4.2 it may be noted that the graph matcher is operated by a sequence of parallel operations in which the system is perturbed and then allowed to settle to equilibrium. Two types of convergence may be noted: neural and pattern. Neural convergence is observed when the system equilibrates after a perturbation such as a spotlight or floodlight activation.

---

<sup>18</sup> The SI problem, among other things, assumes the use of unlabeled graphs and finds all matching subgraphs, i.e., the *best* solution. Our graph matcher uses labeled graphs (which significantly reduces ambiguity) and finds only *good* solutions.

<sup>19</sup> Indeed, it is likely that to obtain such a proof, the system would have to be so simplified as to preclude its proper functioning.

Experimental results indicate that the neural convergence time is generally quite small and invariant: the system typically reaches a neural equilibrium in about 2 or 3 iterations<sup>20</sup>.

The second level of convergence, pattern convergence, is a higher level process that is determined to have occurred when the output from the strongest g-cell remains fixed for some specified number of system iterations. Pattern convergence signifies that one pattern has dominated the recall process for a "long" time. In all of our work, the pattern convergence window is 20 iterations. Perusing the recognition results in Chapter 5, it may be observed that, on average, pattern convergence occurs in about 20 system iterations (excluding the 20 iteration convergence window) and is roughly independent of distortion. See Table 6.1. Distortion manifests itself in the second moment such that increased distortion leads to a greater standard deviation of pattern convergence time about the average. These results, in conjunction with the fact that the patterns are disjunct, suggest that the convergence rate will be unaffected by gallery size. This is because, as implemented, the recall process for each pattern is separate, i.e., it is unaffected by the recall of other patterns. In other words, g-cell activation for a specific pattern is not influenced by the existence or activity of any other pattern and so convergence rate may be independent of inter-graph ambiguity.

Object class	Mean	Std Dev
Undistorted	18	5
Distorted	22	15
High Mixing Ratio	70	48

Table 6.1 - Observed convergence behavior of the graph matcher for different types of objects. It is noted that convergence rate is relatively unaffected by object distortion and is adversely affected by a high mixing ratio.

Such is not the case, however, when the model graphs overlap and so a significant increase in convergence time is noted when the mixing factor (Equation 4.27) is increased. Specifically, Table 6.1 shows the effect on convergence time with the relative mixing factor set to 1.0. That result, combined with the observation in Chapter 5 that a high mixing ratio leads to extremely poor distortion performance, suggests that the system be operated in a low or zero mixing factor mode.

### 6.2.7 Gallery Scaling Issues

In Chapter 5, we demonstrated good recognition results using a 9-object greyscale gallery. While the use of a limited size model base may be adequate for proof of concept and certain restricted applications, nine objects is approximately an order of magnitude below what is useful for general recognition applications. In scaling upward the size of the gallery, the

---

<sup>20</sup> In the vast majority of instances, convergence is to a stable attractor. In exceedingly rare circumstances, the system converges to a (binary) periodic attractor.

critical issue is ambiguity, specifically, ambiguity as seen from the viewpoint of the graph matcher: a model base of unambiguous graphs will result in good recognition results regardless of the size of the gallery. The difficulty, of course, is to maintain discriminability in the face of numerous or similarly appearing objects. Ultimately, the problem becomes one of selecting distinct and specific features while at the same time providing the ability to generalize features and thereby remain robust to distortion.

The approach to feature extraction described in Chapter 2, along with the feature generalizing attributes of the graph matcher, provide the necessary combination of feature distinction and generalization. In particular, the feature extractor learns perceptually salient features that are derived from Morlet wavelet decomposition while the similarity based connections of the graph matcher provide the ability to generalize. [Lades et al., 1991] have demonstrated the efficacy of using Morlet jets as features through their ability to allow recognition of individual faces from a gallery of 88 faces. That system, in contrast to ours, uses jets in their raw, unquantized form and so may more accurately represent features for a specific image. It is thus observed that a limiting factor for our features may be due to the error introduced in the quantization process. Fortunately, this limitation can be controlled by allowing the learned model vector map to expand in size and/or dimension as a function of the convergent quantization error. In other words, as the gallery size is increased and more demands are placed on the feature extractor to provide distinguishable features, a model vector map of larger size and/or dimension may be used. The larger map allows the quantized features to span a larger feature space with the same accuracy (as specified by quantization error) or, alternatively, to span the same feature space with increased accuracy. Using this approach, we expect our system to scale as well as the face recognition system noted above.

### 6.2.8 Dealing with Complex Presentations

The function of our system was inspired by the "what" visual process that exists in the primate brain. That process, said to occur in humans in visual cortex (V1 through V4) and inferotemporal cortex, is principally responsible for object specification or recognition. This is as opposed to the "where" visual process the terminus of which is in posterior parietal cortex and is responsive to object location and motion [Mishkin et al., 1983] [Sagi and Julesz, 1985]. Certain advantages are had by such a what/where scheme and include robustness to translation, rotation, and scale without the combinatoric explosion described by [Tsotsos, 1990].

A fundamental architectural assumption made by our system is that good object segmentation has occurred before presentation of the object to the recognition system. The segmentation could be presumed to include the various methods attributed to the human visual system: color, motion, stereo, texture, etc. The input to the object recognizer, therefore, is expected to contain little clutter and cases to the contrary could be described as camouflaging (i.e., a breakdown of the visual segmentation process such that inadequate information is supplied to the recognition units).

A special case of the camouflage problem is one in which the segmenter is unable to resolve multiple objects that are overlapping and thus presents the mixture to the recognizer. Although we have not carried out experiments to explicitly determine the response of our

system to such input, it is reasonable to expect that the system outputs would correspond to the presented objects and their degree of uniqueness. In other words, because a local feature extractor is used and because the graph matcher detects and builds upon binary subgraphs of those local features, patterns of object features that are not inherently corrupted by the occlusions will cause responses by appropriate model patterns. Presentation of a partial object, therefore, still allows the graph matcher to recognize the object based on the revealed parts.

If, for example, the toy biplane and the toy elephant were overlapped and simultaneously presented to the system, we would expect the system to respond primarily with the biplane and elephant models. It is left up to a "where" system to determine the relative position of each. As the amount of overlap is reduced, i.e., as the objects become more distinguishable, we accordingly expect stronger responses from the biplane and elephant models. In another example, we assume that the two input objects are of the same type, say, biplanes. In this case, the system will respond with a single, strong biplane response; it is again left up to a "where" system to determine the number and locations of the objects.

### 6.2.9 Fusion

The final extension deals with a limitation specific not only to our system, but to the majority of visual recognition systems developed to date: the exclusive use of uni-modal data in feature extraction and classification. Natural visual systems of sufficient complexity to perform the generalized recognition tasks that the artificial vision community is interested in solving, invariably process data in a variety of visual modalities. It is unrealistic to believe that computer vision systems will be in any way competitive with these biological systems as long as they restrict themselves to a single visual mode. Our system was initially developed for the purpose of visual recognition using fusion where the fusion was to occur at the feature-level. Indeed, the entire design of the system is predicated on its extension to multi-modal processing. Details of that extension are too numerous to expound upon here and may be found in the thesis proposal. The main idea, however, is that from the standpoint of the graph matcher (which performs the actual classification), a graph is merely a collection of topologically related features. There is no bias as to the origin or meaning of the features, only to similarity relationships between labels. The fusion, per se, occurs during graph formation where features from several modalities that are known to be spatially proximate are linked together by edges (or T-links). The resulting structure is a multi-modal object graph and represents the object in the modalities from which it was constructed. The graph matcher, containing now a multi-modal model graph, matches the input multi-modal graph to subgraphs in the database entirely without modification to the graph matcher.

---

In this dissertation, we have described and demonstrated a mid-level object recognition system. Although limited in its present form, it introduces a powerful approach to object recognition that is built on adaptive feature extraction and dynamical link graph matching. The current system, complemented by various modifications (some of which having been presented in the preceding paragraphs), provides a significant alternative to existing artificial recognition systems.

## Appendix A - Data Structures for the Feature Extractor

This Appendix introduces data structures fundamental to the feature extractor. It should be noted that this paper did not originate many of the terms described here. They are included, however, to provide a common basis for unambiguous explanation.

We define the *image* class as the class of 2D arrays where each array element is scalar. The size of images are arbitrary for the purposes of this paper but are typically 128x128. There are many instantiations of the image class and they are generally defined as they are introduced. Examples of image instantiations are greyscale images, jet magnitude images, and feature images.

A *Morlet jet* (or *jet*) is defined as a vector of *Morlet wavelet components*. See Figure A.1a. A Morlet wavelet component is the computational result of filtering the input image with a Gabor filter that is centered at a specific point in the image. In our system, the Gabor kernels used in the convolution are self-similar (i.e., they are *Morlet wavelets*) and are unique in orientation and spatial frequency.

A 2D array of Morlet jets is called a *Morlet block* (or *block*). See Figure 2.3b. Each jet in a Morlet block corresponds to a pixel in the originating image.

A *feature vector* is typically considered to be a vector of elements where each element contains a numeric description of the feature assigned to that place in the vector. We also use that definition of feature vector but restrict the feature types to be Morlet wavelet components. At some point in the processing, specifically during adaptive vector quantization, the terms feature vector and jet may appear to be synonymous. There is, however, a subtle difference in that we restrict the definition of a jet to be the direct computational results of a (Morlet wavelet) decomposition of an image about a single point whereas a feature vector may be either a jet or a vector derived from a jet.

We define a *vector map* to be an array of feature vectors (similar to the feature maps of [Kohonen, 1988]). Our specific definition of a vector map requires the dimension of the map to be two and square and that a neighborhood topology based on Euclidean distance be employed. We use vector maps to discretize the vector space that represents our feature set which is accomplished with adaptive vector quantization.

A *jet image* is less a data structure than a graphic representation of one. Specifically, it is a Morlet jet rearranged into a 2D depiction with filter orientation running vertically and spatial frequency running horizontally. See Figure A.1. Jet images are useful in their ability to graphically represent the contents of a jet. A larger data representation, called a *jet image array*, is a 2D array of jet images and is used to graphically depict Morlet blocks and vector maps.

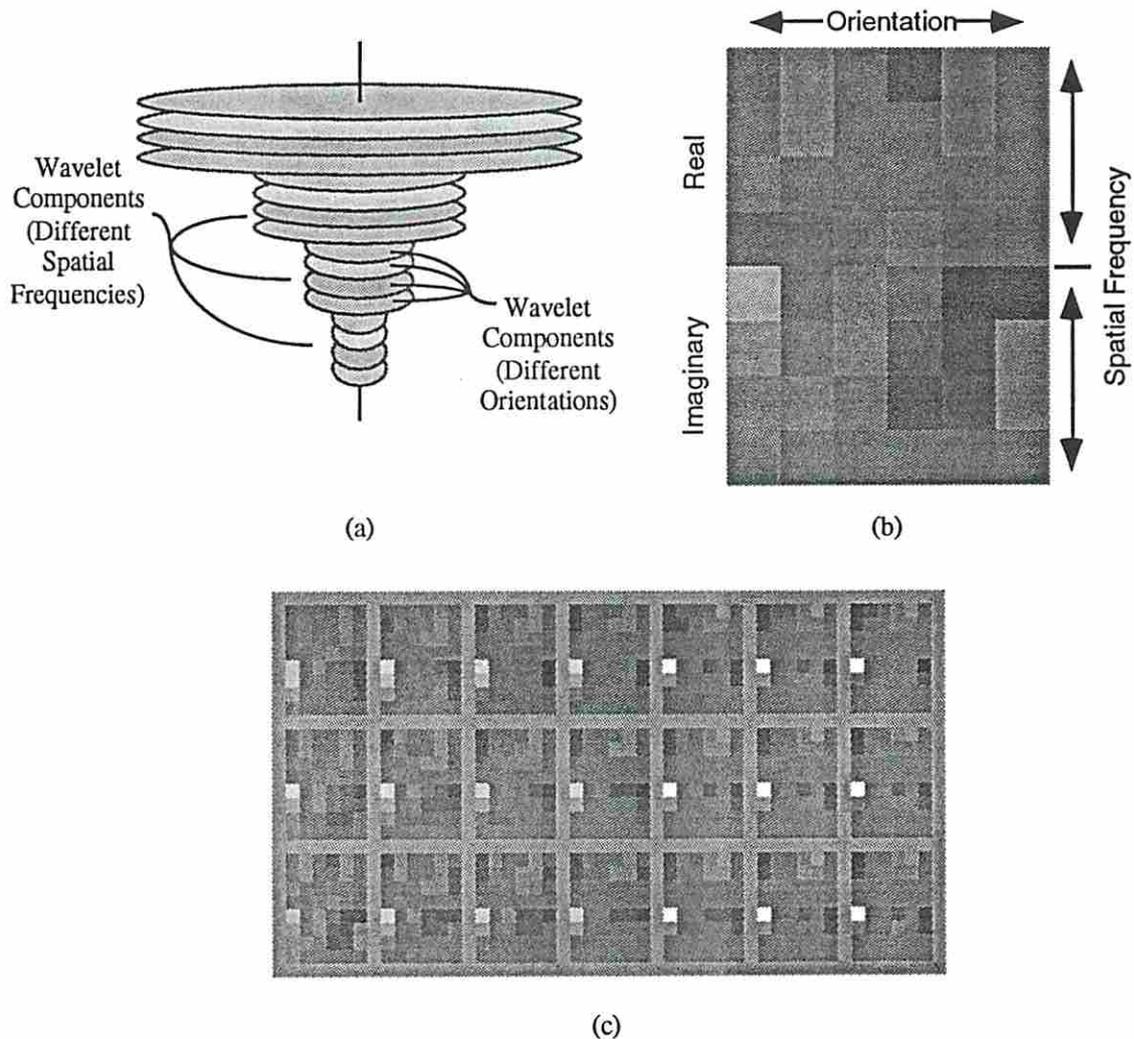


Figure A.1 - Graphic jet depictions. (a) jet schematic, (b) jet image example, (c) jet image array example. In (a), the varying disk size signifies the varying receptor field sizes at the different spatial frequencies. In (b) and (c), each square represents a jet component value. Because the jet is a complex quantity, the jet image is segregated into real and imaginary parts.

## Appendix A - Data Structures for the Feature Extractor

This Appendix introduces data structures fundamental to the feature extractor. It should be noted that this paper did not originate many of the terms described here. They are included, however, to provide a common basis for unambiguous explanation.

We define the *image* class as the class of 2D arrays where each array element is scalar. The size of images are arbitrary for the purposes of this paper but are typically 128x128. There are many instantiations of the image class and they are generally defined as they are introduced. Examples of image instantiations are greyscale images, jet magnitude images, and feature images.

A *Morlet jet* (or *jet*) is defined as a vector of *Morlet wavelet components*. See Figure A.1a. A Morlet wavelet component is the computational result of filtering the input image with a Gabor filter that is centered at a specific point in the image. In our system, the Gabor kernels used in the convolution are self-similar (i.e., they are *Morlet wavelets*) and are unique in orientation and spatial frequency.

A 2D array of Morlet jets is called a *Morlet block* (or *block*). See Figure 2.3b. Each jet in a Morlet block corresponds to a pixel in the originating image.

A *feature vector* is typically considered to be a vector of elements where each element contains a numeric description of the feature assigned to that place in the vector. We also use that definition of feature vector but restrict the feature types to be Morlet wavelet components. At some point in the processing, specifically during adaptive vector quantization, the terms feature vector and jet may appear to be synonymous. There is, however, a subtle difference in that we restrict the definition of a jet to be the direct computational results of a (Morlet wavelet) decomposition of an image about a single point whereas a feature vector may be either a jet or a vector derived from a jet.

We define a *vector map* to be an array of feature vectors (similar to the feature maps of [Kohonen, 1988]). Our specific definition of a vector map requires the dimension of the map to be two and square and that a neighborhood topology based on Euclidean distance be employed. We use vector maps to discretize the vector space that represents our feature set which is accomplished with adaptive vector quantization.

A *jet image* is less a data structure than a graphic representation of one. Specifically, it is a Morlet jet rearranged into a 2D depiction with filter orientation running vertically and spatial frequency running horizontally. See Figure A.1. Jet images are useful in their ability to graphically represent the contents of a jet. A larger data representation, called a *jet image array*, is a 2D array of jet images and is used to graphically depict Morlet blocks and vector maps.

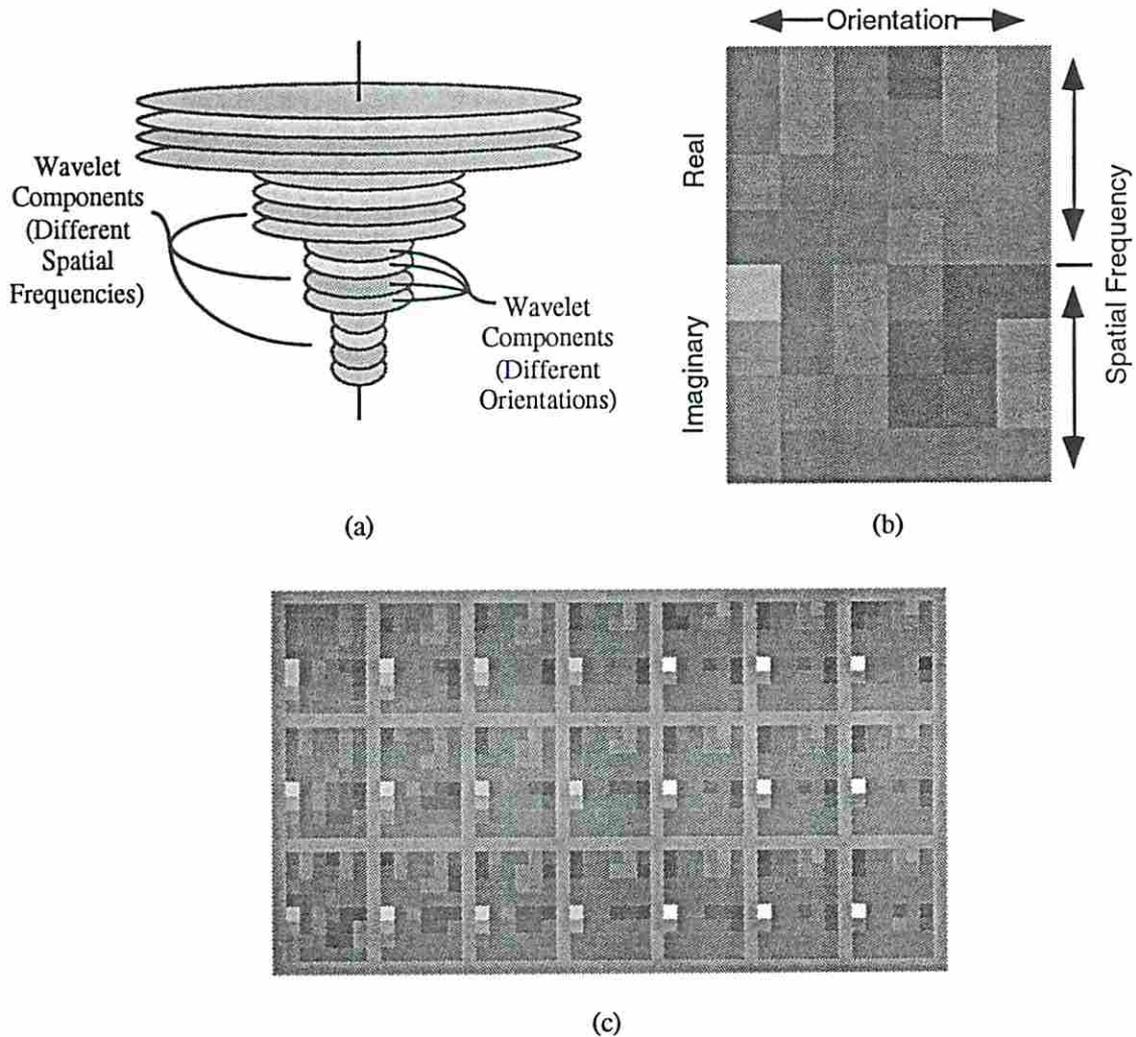
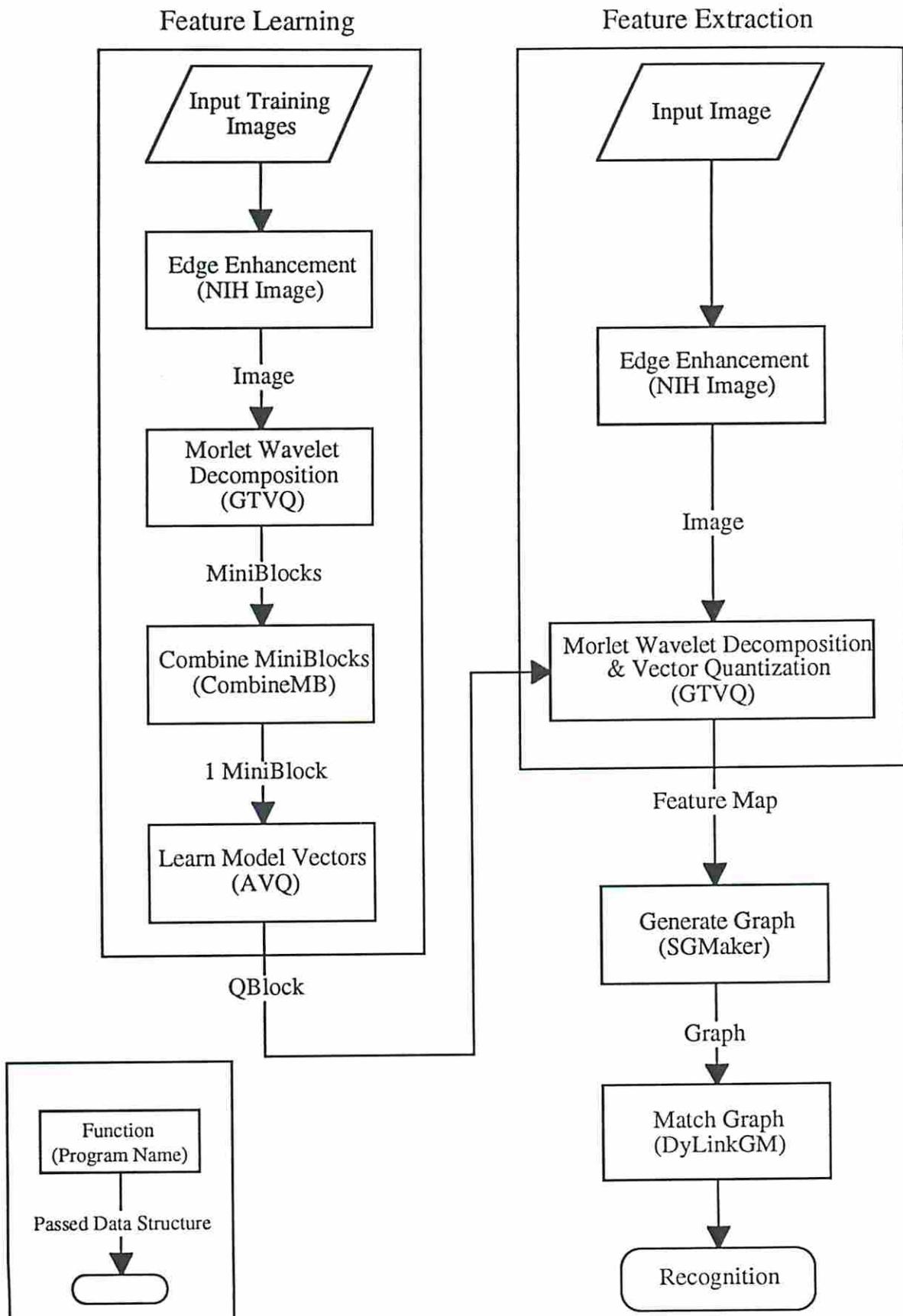


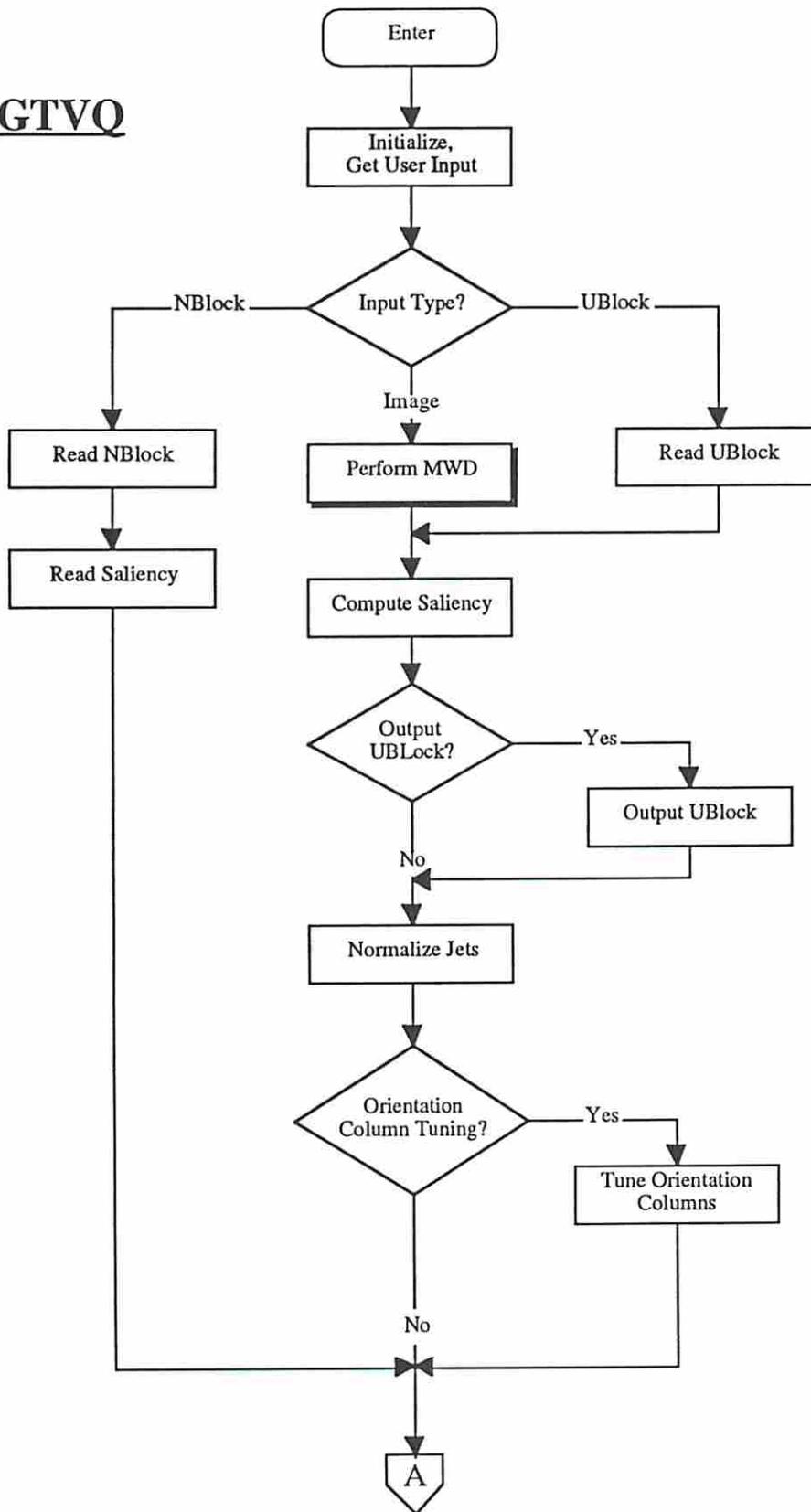
Figure A.1 - Graphic jet depictions. (a) jet schematic, (b) jet image example, (c) jet image array example. In (a), the varying disk size signifies the varying receptor field sizes at the different spatial frequencies. In (b) and (c), each square represents a jet component value. Because the jet is a complex quantity, the jet image is segregated into real and imaginary parts.

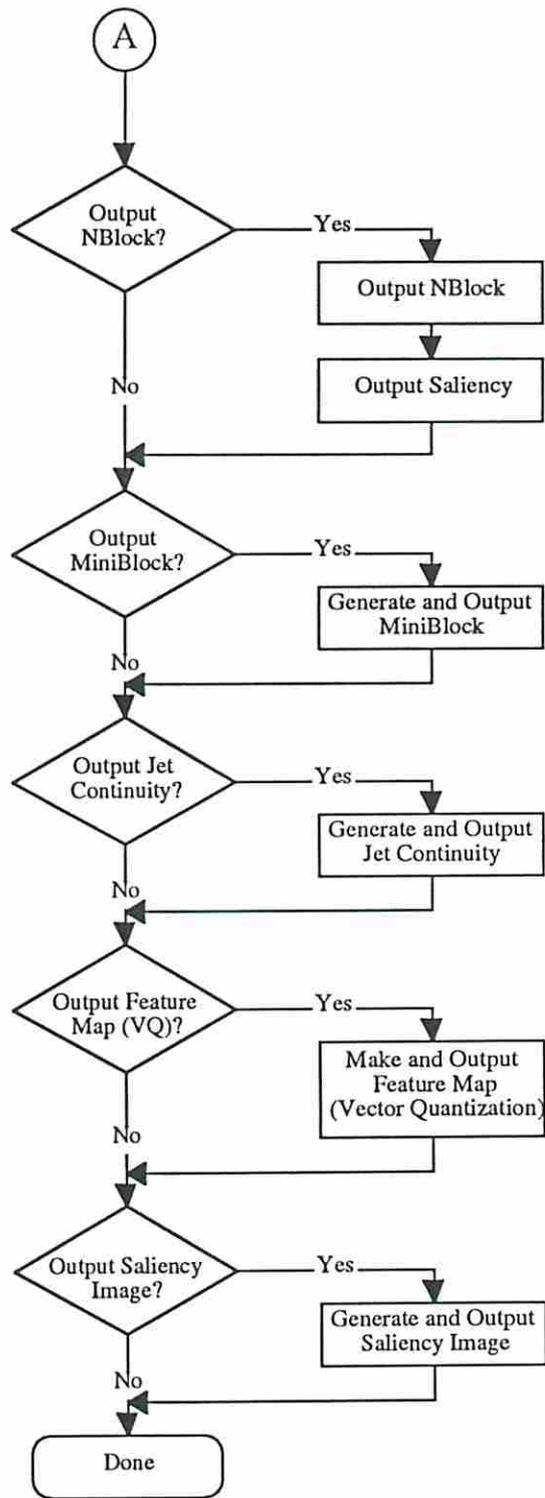
## Appendix B - System Flowchart



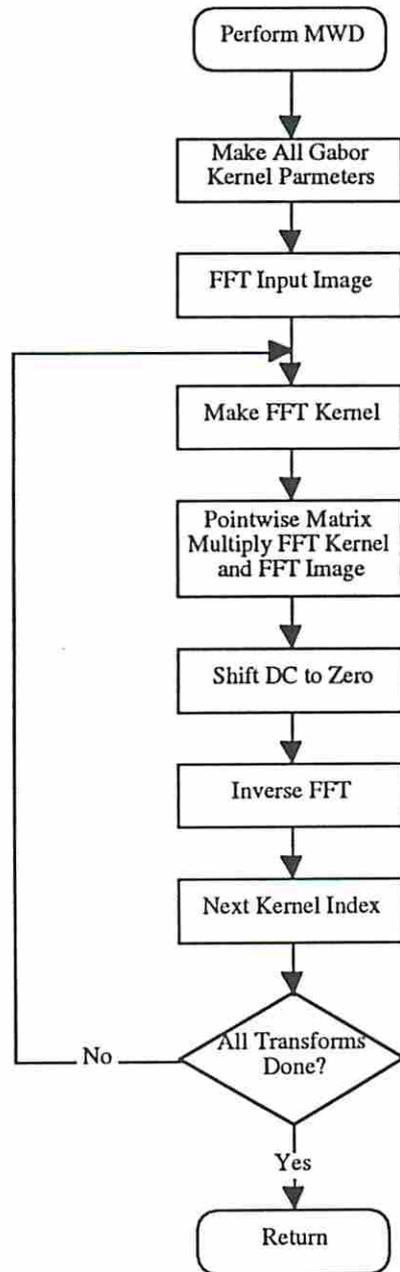
## **Appendix C - Program Flowcharts for GTVQ**

# GTVQ



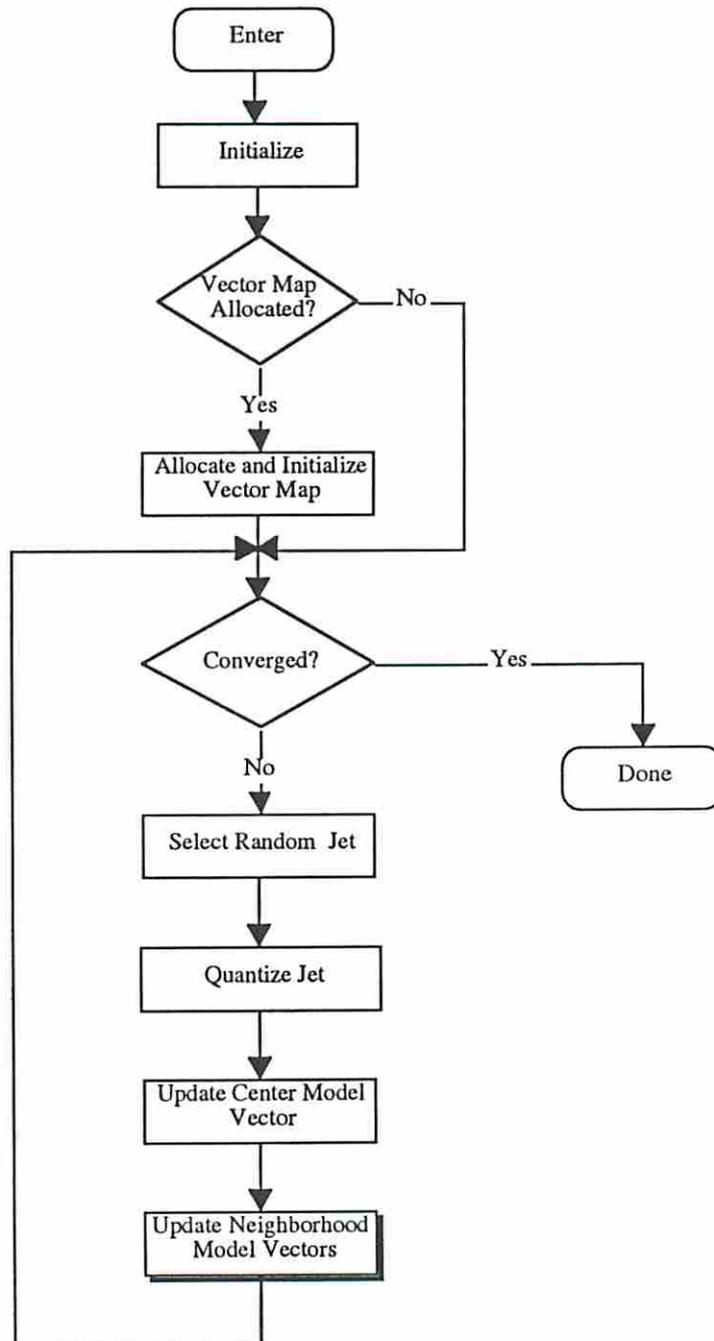


# GTVQ/Perform MWD

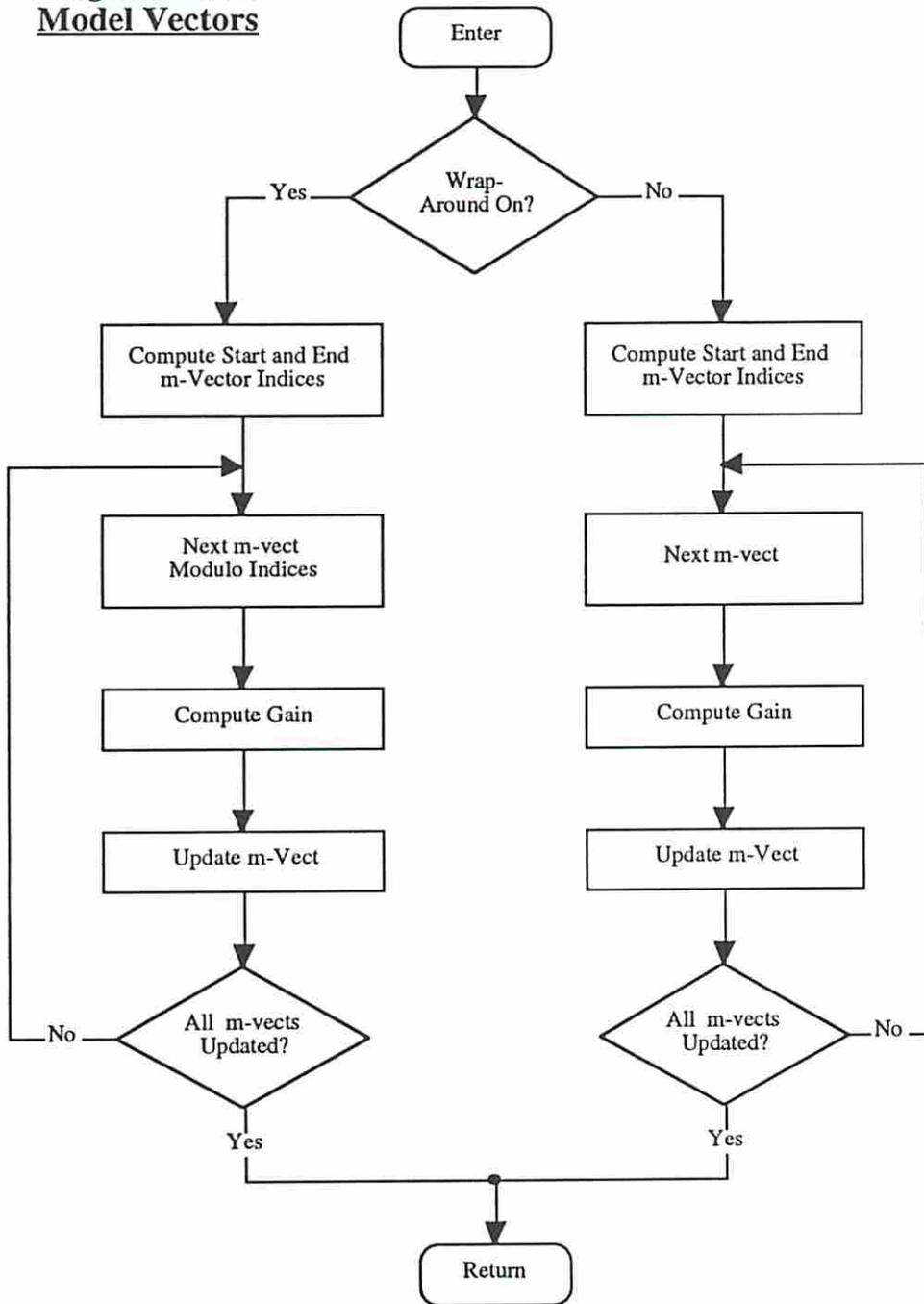


# Appendix D - Program Flowcharts for AVQ

## AVQ Learning

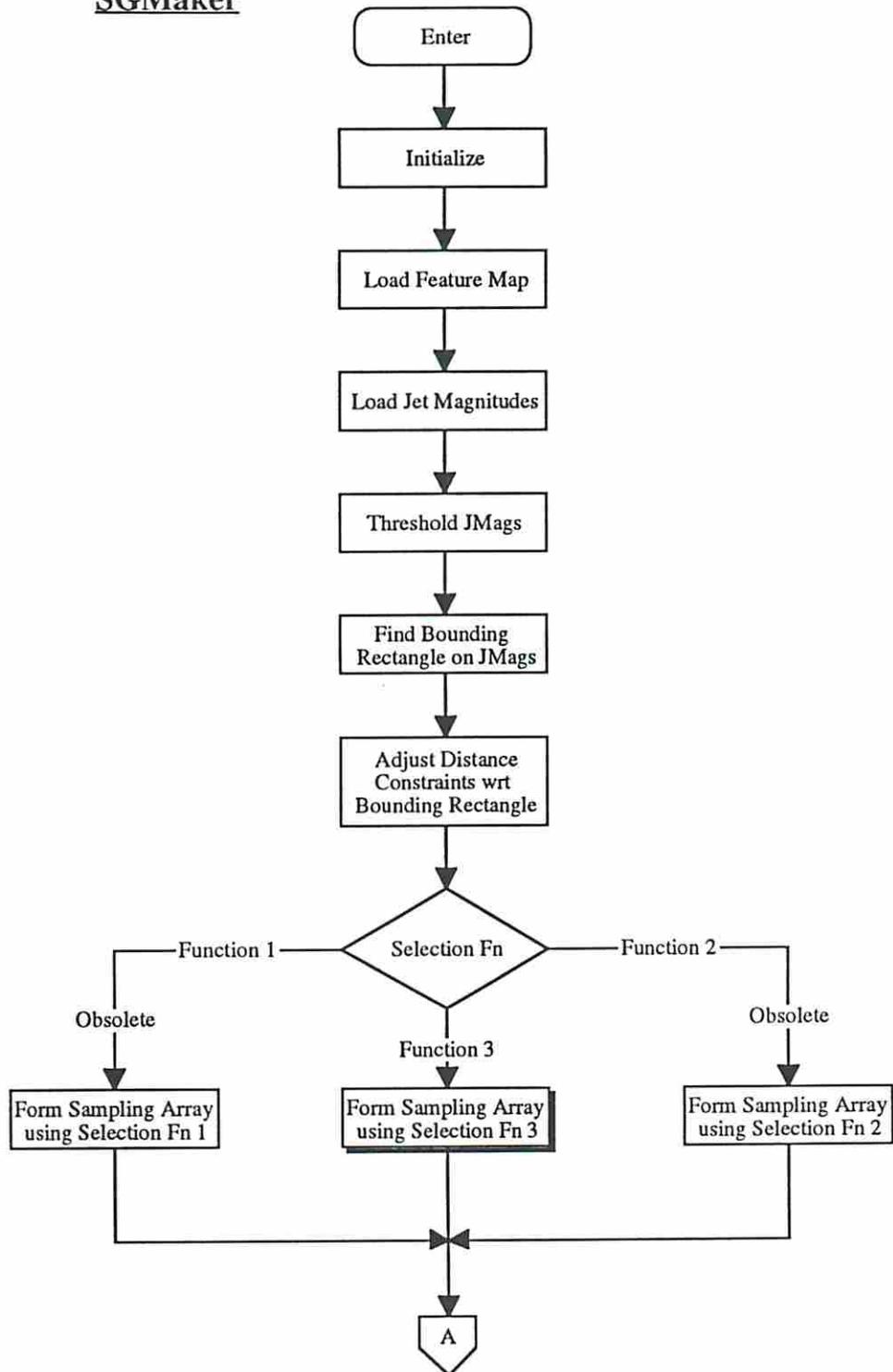


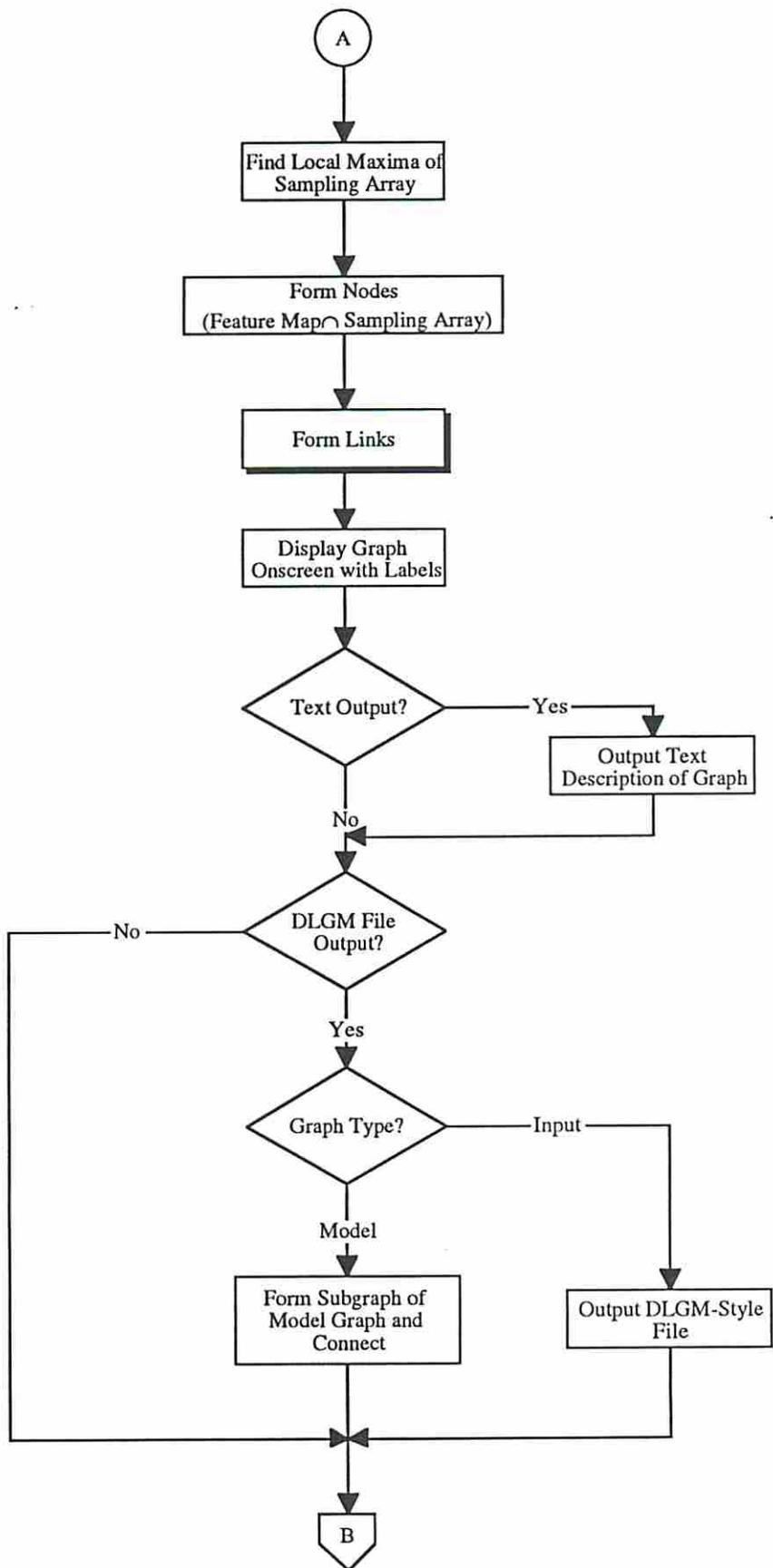
AVQ  
Learning/Update  
Neighborhood  
Model Vectors

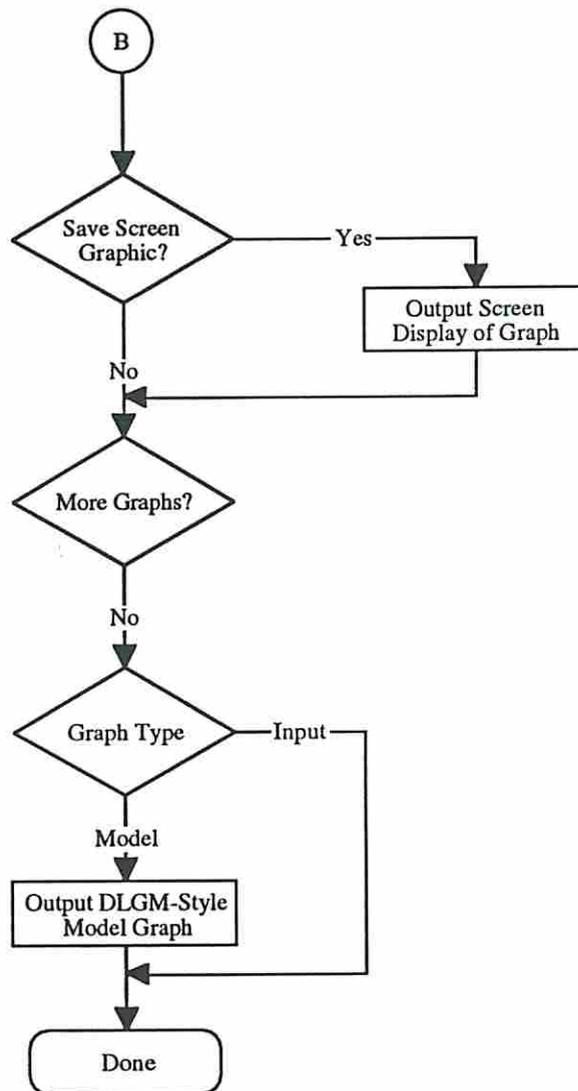


## **Appendix E - Program Flowcharts for SGMaker**

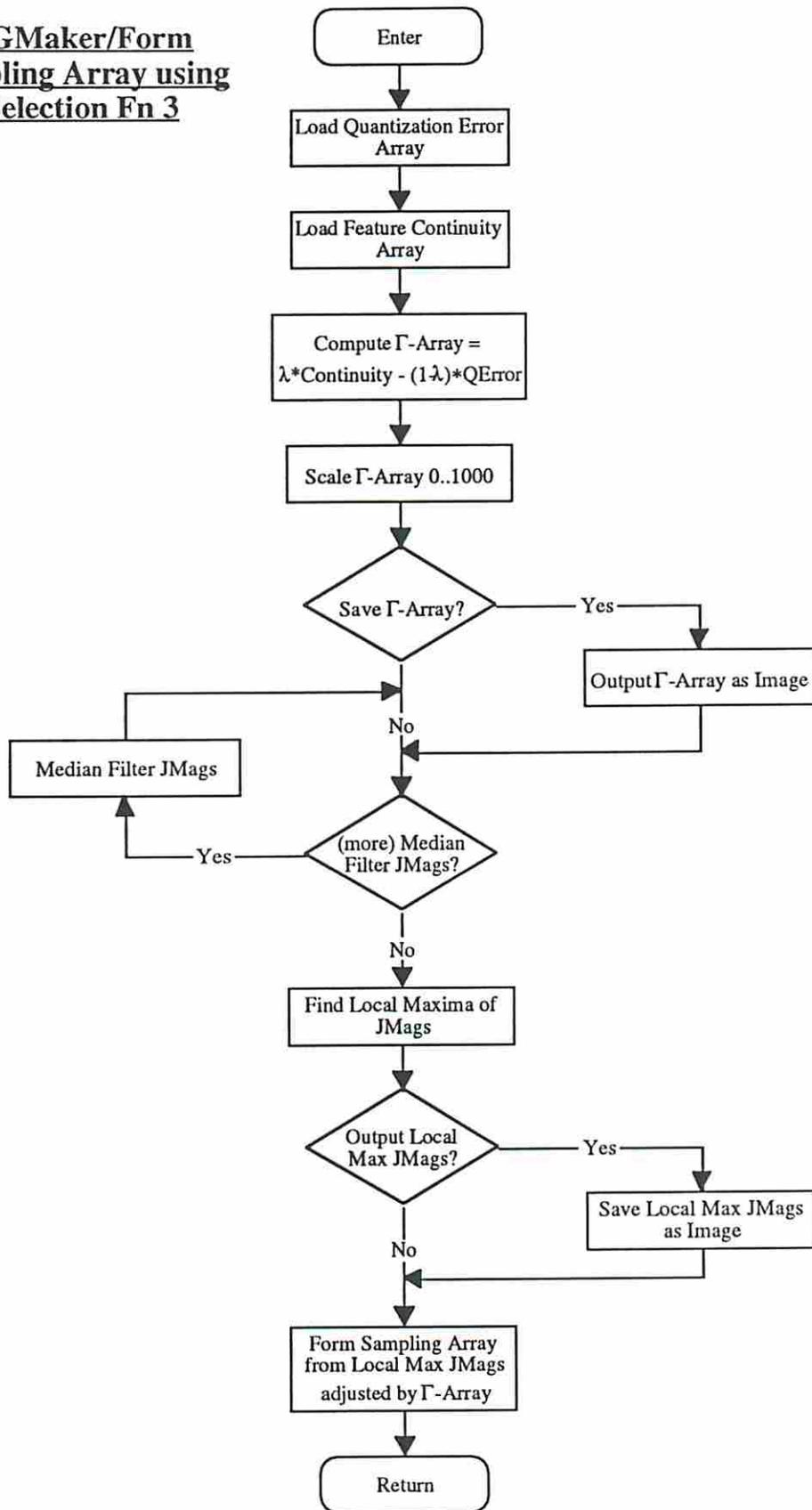
# SGMaker



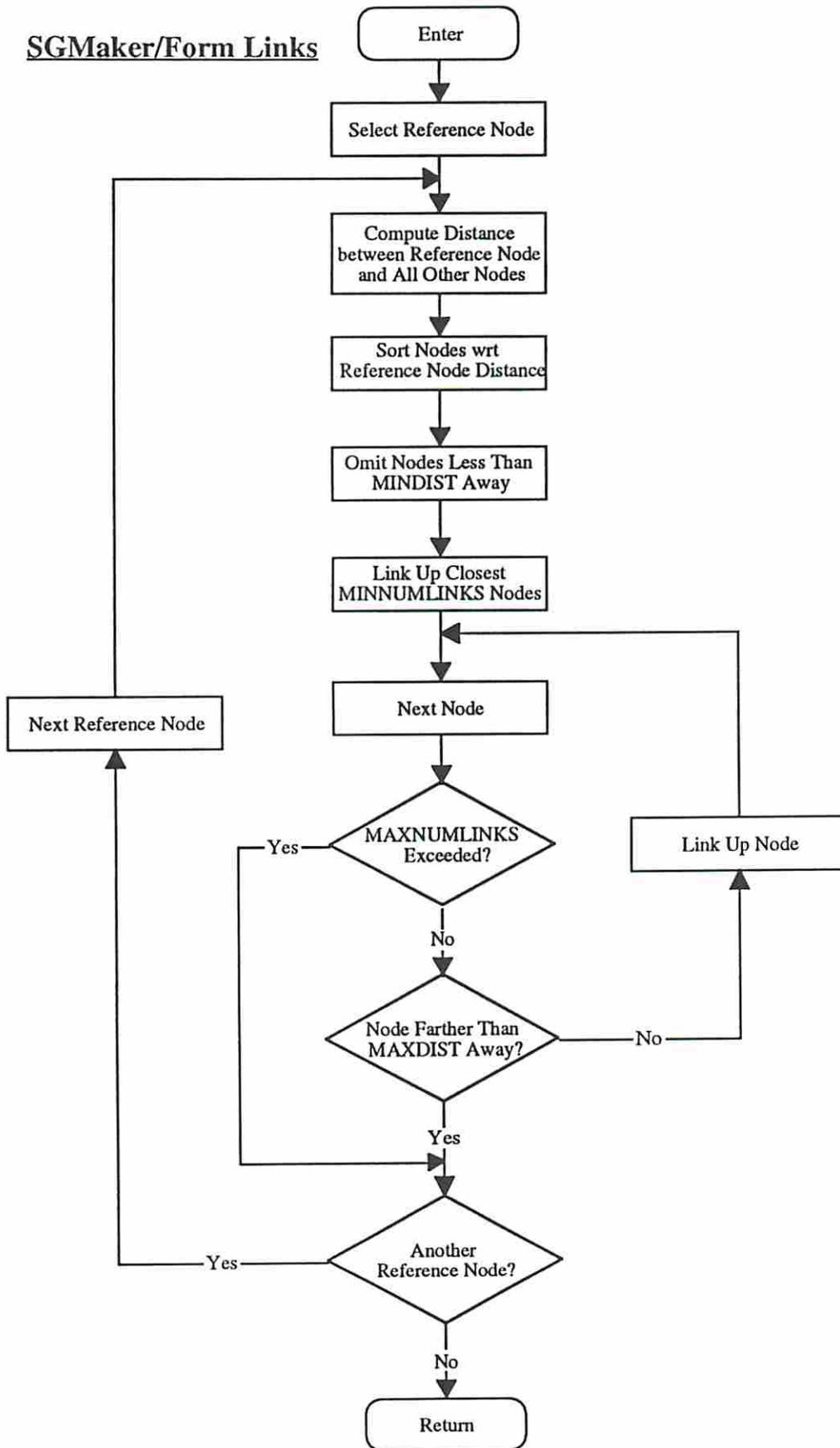




**SGMaker/Form**  
**Sampling Array using**  
**Selection Fn 3**

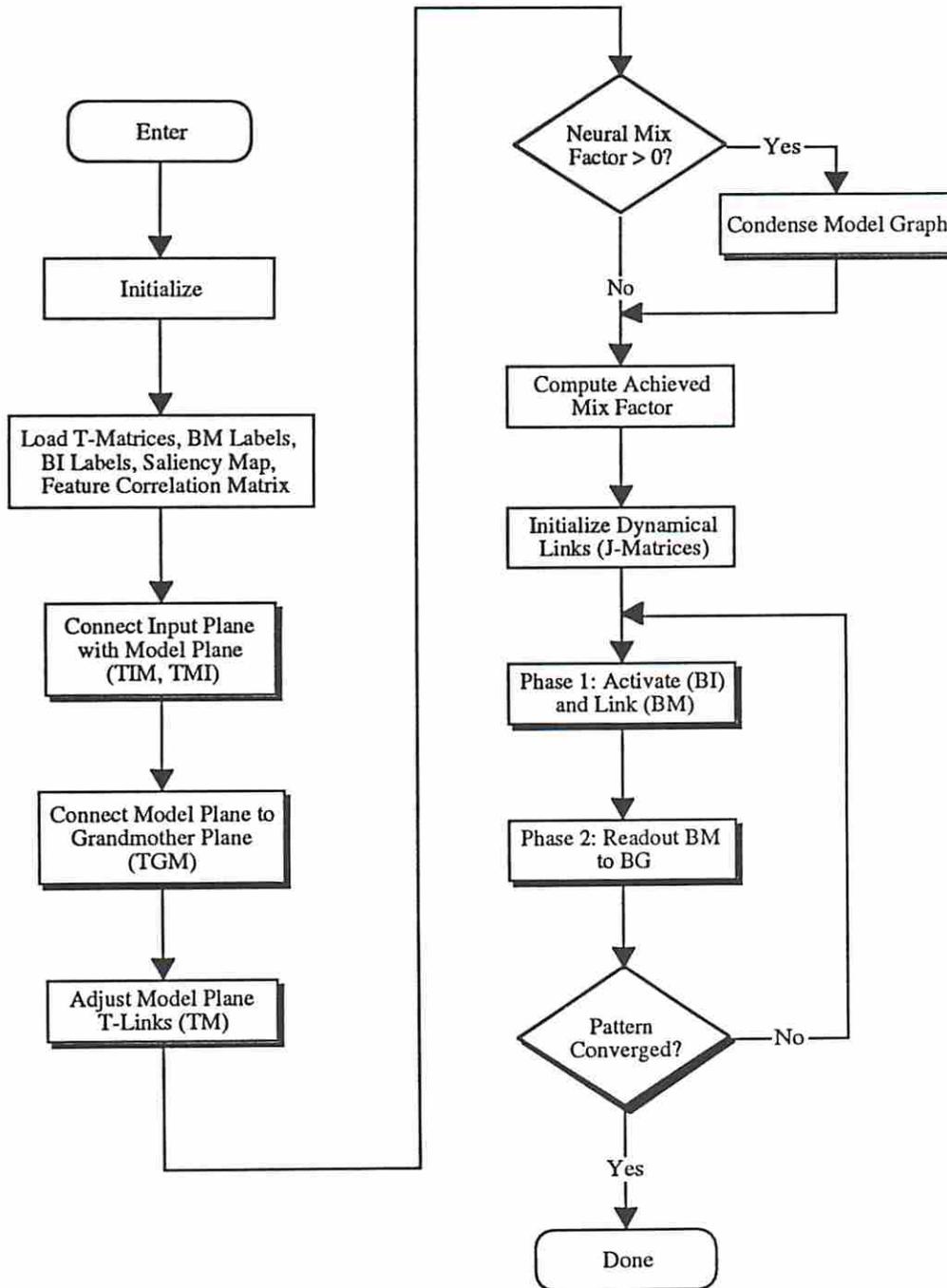


**SGMaker/Form Links**

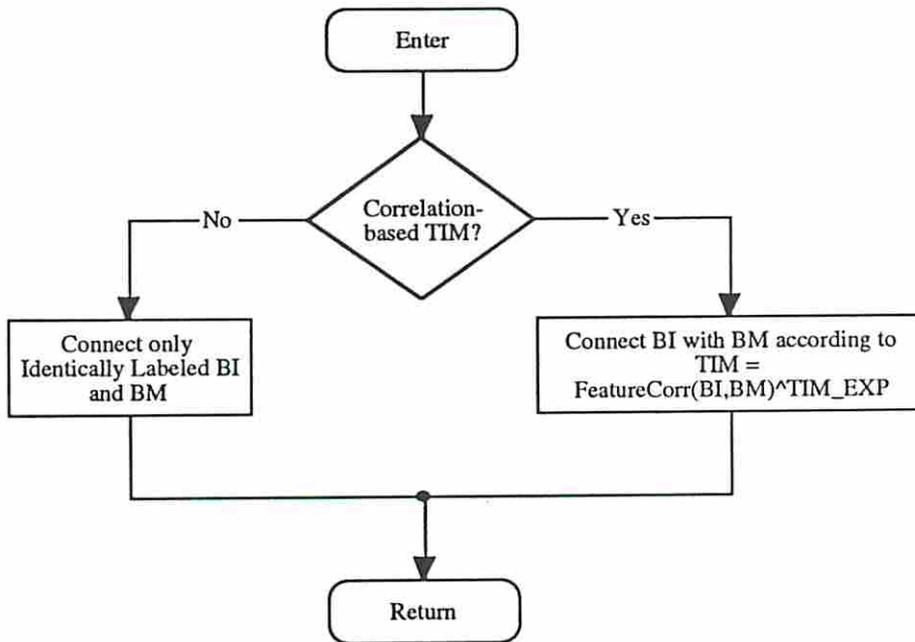


## **Appendix F - Program Flowcharts for DyLink GM**

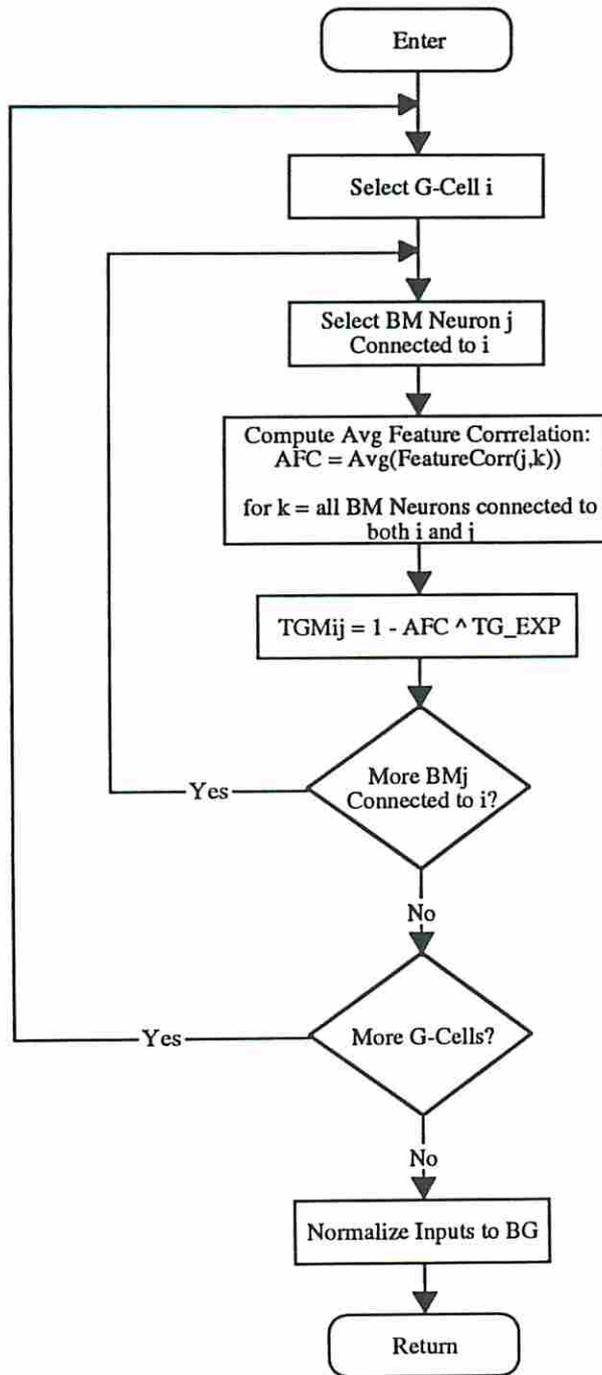
# DyLink GM



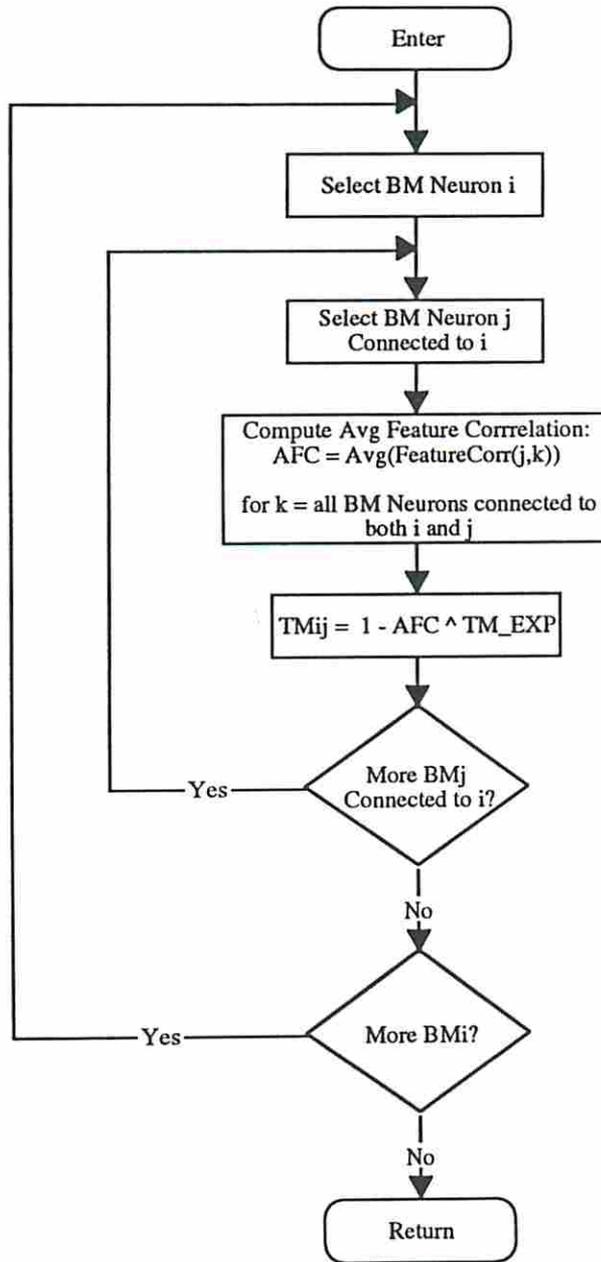
DyLink GM/Connect  
Input Plane with Model  
Plane (TIM, TMI)



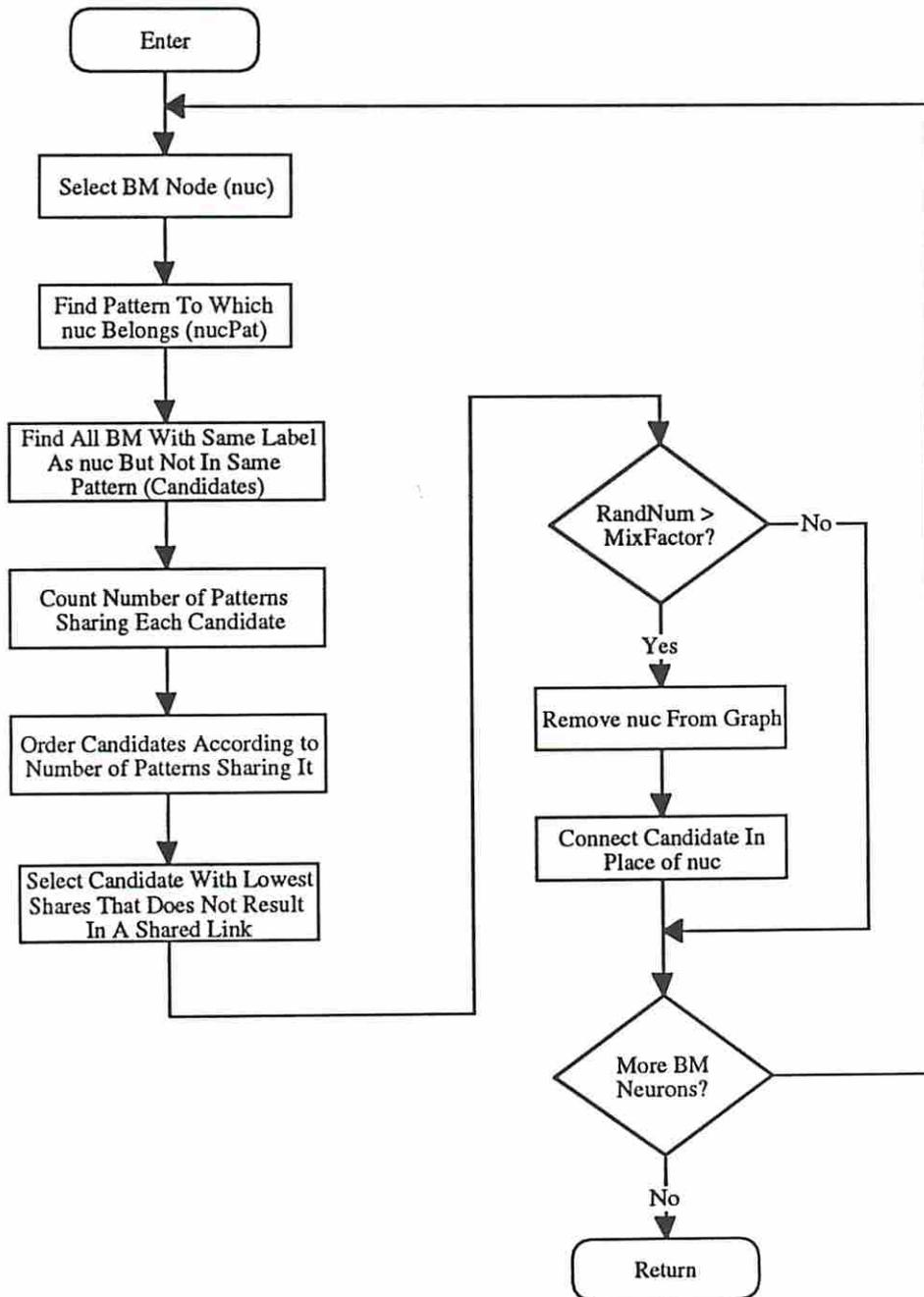
DyLink GM/Connect  
Model Plane to  
Grandmother Plane  
(TGM)



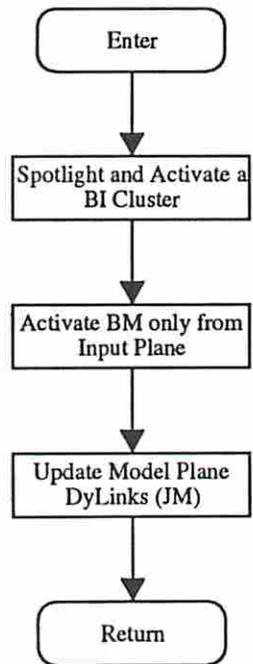
DyLink GM/Adjust  
Model Plane T-Links  
(TM)



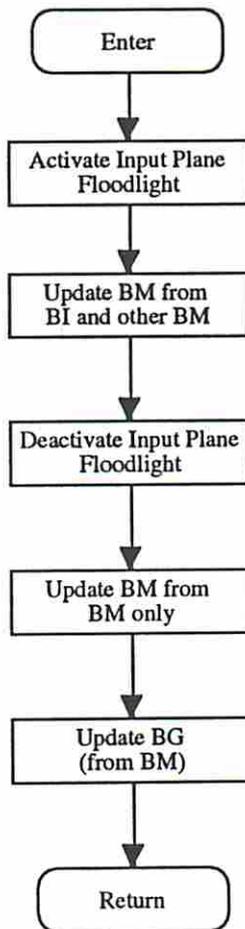
# DyLink GM/Condense Model Graph



**DyLink GM/Phase 1:**  
**Activate (BI) and Link**  
**(BM)**



**DyLink GM/Phase 2:**  
**Readout BM to BG**



## Appendix G - Dialog Boxes for GTVQ

This Appendix contains the Macintosh dialog boxes for the program GTVQ as well as a description of some of the parameters and options. GTVQ performs Morlet wavelet decompositions and vector quantizations. When the programs were originally written, use of the term "Gabor" for describing Gabor derived data structures was dominant. Since then, as noted in Chapter 2, the terminology has evolved such that these structures are now referred to as "Morlet" or "Morlet wavelet" structures. Because the dialog boxes still contain the word "Gabor", our descriptions of them here use both Gabor and Morlet terms to remind the reader of the correct terminology.

Upon entering GTVQ, the first dialog box encountered is General Parameters. It is from this box that major I/O and processing options are selected thereby introducing other dialog boxes. Below, we describe the parameters of each dialog box.

### General Parameters

#### Input Data Selection

This set of radio buttons specifies the type of input: TIFF image, unnormalized Gabor (Morlet) block, normalized Gabor (Morlet) block, or none of the above. Selecting TIFF image forces the program to perform a Morlet wavelet decomposition. Selection of either type of Gabor (Morlet) block as input bypasses wavelet decomposition. The "None" option is used for debugging.

#### Output Data Selection

This set of checkboxes specifies the format in which the wavelet-decomposed image is output. An unnormalized Gabor (Morlet) block is the standard data structure described in Chapter 2. If each jet in the block is normalized to unity, it is called a normalized Gabor (Morlet) block. A mini-block contains an array of normalized jets, the positions of which have no significance. Mini-blocks are typically used to contain a set of select jets chosen on the basis of jet magnitude.

## Other Processing

Jet Continuity - compute and output a jet continuity image.

Jet Magnitude Output - output a jet magnitude image.

Orientation Tuning - perform orientation column tuning. Available only if the input data type was Image.

Vector Quantization - perform vector quantization on the Gabor (Morlet) block. The program will later ask for the mini-block containing the model vectors.

Component Histogram - analyzes the Gabor (Morlet) block and presents a histogram of Gabor (Morlet) components.

## Misc. Parameters

Continuity Radius - radius of neighborhood in jet continuity processing.

## Wavelet Decomposition Parameters

Convolution Method - Complex or Real FFT.

Orientations - number of filter orientations in the decomposition.

Spatial Frequencies - number of filter spatial frequencies in the decomposition.

Highest SF Wavelength - wavelength (in pixels) of highest spatial frequency filter in the decomposition.

Dist between SFs - multiplicative spatial frequency increment. E.g., a value of 2.0 causes the frequency to double (one octave) at each successively higher spatial frequency.

Sigma - frequency independent Gaussian rolloff ( $\sigma$ ).  $\sigma = \pi/x$  where user enters  $x$ .

Zero-Pad FFT - when enabled, prevents aliasing by doubling the effective sampling rate.

Output Magnitude Convolution Image - output the complex magnitude results of each filter.

Output Real Convolution Image - output the real part of the results of each filter.

Output Imaginary Convolution Image - output the imaginary part of the results of each filter.

## Gabor (Morlet) Block Output Options

These parameters determine the spatial frequency indices used in jet construction and jet magnitude computation. The box is otherwise self-explanatory.

## Mini-Block Parameters

These options and parameters are used in the selection of jets for inclusion in a mini-block. Typically, the mini-blocks so formed are used as training vectors in Adaptive Vector Quantization.

Threshold - percent of difference between largest and smallest jet magnitude, over which each jet is considered as a candidate for selection.

Selection Window Radius - radius about the image center outside of which all jets are excluded from selection. -1 selects the entire Gabor (Morlet) block.

Jet Selection - methods by which jets may be selected.

Thresholded Values - selects all jets whose magnitude exceeds threshold described above.

Local Maxima - selects only those jets whose magnitudes are local maxima. Sparseness is imposed by requiring a minimum distance between selected jets.

Global Maximum - selects only the jet with the largest jet magnitude.

## VQ Options and Parameters

This box contains parameters that affect vector quantization.

SNR Threshold - percent of difference between largest and smallest jet magnitude, over which each jet is quantized.

Output Feature Map - output an image in which the value of each pixel corresponds to the label of the model vector to which the respective jet was quantized.

Output Error Map - output an image in which the value of each pixel corresponds to the error associated with the quantization of the respective jet to the closest model vector.

**General Parameters**

<b>Input Data Selection</b> <input checked="" type="radio"/> Image (TIFF) <input type="radio"/> UnNormalized Block <input type="radio"/> Normalized Block <input type="radio"/> None	<b>Output Data Selection</b> <input type="checkbox"/> UnNormalized Block <input type="checkbox"/> Normalized Block <input type="checkbox"/> Mini-Block	<b>Other Processing</b> <input type="checkbox"/> Jet Continuity <input type="checkbox"/> Jet Magnitude Output <input checked="" type="checkbox"/> Orientation Tuning <input type="checkbox"/> Vector Quantization <input type="checkbox"/> Component Histogram
--	---	---

**Misc. Parameters**  
Continuity Radius:

**Log Entry:**

**Wavelet Decomposition Parameters**

Convolution Method:  
 **Complex FFT**     **Real FFT**

---

**Orientations:**              
**Spatial Frequencies:**      
**Highest SF Wavelength:**   
**Dist Between SFs:**          
**Sigma:**                     $\pi /$

**Zero-Pad FFT (Takes longer)**

---

**Output Magnitude Convolution Images**  
 **Output Real Convolution Images**  
 **Output Imaginary Convolution Images**

**Gabor Block Output Options**

(0 is lowest frequency)

Jets  
Output spatial frequency indexes  to

Jet Magnitudes  
Compute for spatial frequency indexes  to

**Mini-Block Parameters**

Threshold (%) on  
(JMagMax - JMagMin):

Selection Window Radius:

---

**Jet Selection:**

Use Thresholded Values

Use Local Maxima of Jet Mags

Minimum Dist  
between Maxima:

Use Global Maximum of Jet Mags

**DQ Options and Parameters**

**General Parameters**

SNR Threshold:

**Output**

Feature Map

Error Map

## Appendix H - Dialog Boxes for AVQ

This Appendix contains the Macintosh dialog boxes for the program AVQ as well as a description of some of the parameters and options.

Upon entering Program AVQ, the first dialog box encountered is General Parameters. It is there that the user selects the 2D vector map learning (*Kohonen 2D Learning*) discussed in Chapter 2. All other options in the General Parameters box are for analysis purposes and are unimportant to the dissertation. Then second box encountered is the 2D Kohonen Parameters box which contains all of the parameters for the 2D vector map learning. Those parameters and options are described below:

Vector Map Rows & Cols - Dimensions of the vector map.

Quantization Threshold - Threshold over which training jets are allowed to affect the vector map during learning. Threshold is percent of maximum training jet magnitude.

Alpha - Nominal gain.  $\alpha_k$  in Equation 2.7.

Neighborhood Radius - Radius of area effected by neighborhood updates.  $3\sigma$  point of Gaussian. See Equation 2.7.

Wrap Around - When enabled, creates a seamless vector map such that the left and right columns are neighbors as are the top and bottom rows.

Gain fn(JMag) - When enabled, multiplies the update gain by a term that is a function of the training jet magnitude. See Equation 2.7.

Initialize with Existing Map - When enabled, initializes vector map with a stored vector map. When disabled, initializes vector map with random vectors.

Linearly Decreasing Gain - When enabled, multiplies the update gain by a term that decreases linearly over iteration. See Equation 2.7.

Convergence Threshold - Number of iterations for learning.  $t_{max}$  in Equation 2.7.

Averaging Window Length - Window length on quantization running average. Used for display purposes only; has no effect on learning.

**Methods, General Parameters, and Options**

Kohonen 2D Learning  
 Map QBlock1 to QBlock2  
 Cluster Analysis  
 Spare 1       Show Mapping

---

Condense Mapping  
 Show Condensed Mapping

**2D Kohonen Parameters**

**General Parameters**

Vector Map Rows & Cols:   
 Quantization Threshold (%):

**Learning Parameters**

Alpha:   
 Neighborhood Radius:   
 Wrap Around       Initialize with existing map  
 Gain fn(JMag)       Linearly decreasing gain

**Convergence Criteria**

Convergence Threshold:   
 Averaging Window Length:

## Appendix I - Dialog Box for SGMaker

This Appendix contains the only Macintosh dialog box for the program SGMaker (Sparse Graph Maker) as well as a description of some of the parameters and options. SGMaker is responsible for converting feature maps into sparse graphs. Parameters and options relating to SGMaker are herewith described:

Selection Function - Chooses the Selection Function that specifies the most salient, reliable, and accurate nodes. Selection Functions 1 and 2 are obsolete and Selection Function 4 is a spare. The algorithm and results described in the dissertation are due to Selection Fn 3.

Minimum Distance - Specifies the minimum distance allowed between two nodes. The node with the greater selection value survives.

Saliency Fn Threshold - Only features pixels with saliency in excess of this threshold are considered as candidates for nodes. Expressed as a percent of saliency maximum minus minimum. Increases processing efficiency by reducing the number of elements processed.

Selection Fn Lambda -  $\rho$  in Equation 3.3. Governs emphasis of continuity with respect to accuracy.

Median Filter Size - Size of the 2D median filter mask (on a side) used on the saliency array.

Filter Applications - Number of applications of the 2D median filter to the saliency array.

Max Distance Between Nodes - Obvious.

Min Number of Neighbors - Obvious.

Max Number of Neighbors - Obvious.

Text Data Output - Incurs a textual description of the graph; output as a Word file.

Graph Matcher Files (DLGM) Output - Creates graph data file directly useable by the graph matching program, DyLinkGM.

BI Graph(s) - Indicates that the graph(s) to be created are single-object input graphs.

BO Graph - Indicates that the graph to be created is a multiple-object model graph.

Labeled Graph Graphic Output - Output a TIFF file that is a graphic image of the graph, with IDs and feature labels for nodes.

Unlabeled Graph Graphic Output - Output a TIFF file that has a graphic image of the graph, without IDs or feature labels for nodes.

No Graph Graphic Output - Obvious.

Staccuracy Image Graphic Output - Output a TIFF file of the  $\Gamma$ -array.

Sample Array Graphic Output - Output a TIFF file of the Sampling Array.

JMag Local Graphic Output - Output a TIFF file of the saliency local maxima.

Batch Input - Process all feature maps in a given folder (subdirectory).

Log Output - Dump console to a log file.

**Graph Maker**

**Nodes**  
 Selection Fn 1    Selection Fn 2    Selection Fn 3    Selection Fn 4  
Minimum Distance:       Median Filter Size:   
Saliency Fn Threshold (%):       Filter Applications:   
Selection Fn Lambda ( $\leq 1.0$ ):

**Edges**  
Max Distance between Nodes:   
Min Number of Neighbors:   
Max Number of Neighbors:

**Data Output**  
 Text (Word)  
 Graph Matcher Files (DLGM)  
 BI Graph(s)    BO Graph

**Graphic Output**  
 Labeled Graph       Staccuracy Image  
 Unlabeled Graph       Sample Array  
 No Graph               JMag Local Max

**Other**  
 Batch Input  
 Multi-Batch  
 Log Output

## **Appendix J - Dialog Boxes for DyLink GM**

Initialization	
<p><b>Neural Activation Parameters</b></p> <p><u>BI and BIJ cells</u></p> <p><input checked="" type="radio"/> Binary      Threshold: <input type="text" value="0.5"/>  <input type="radio"/> Graded      Gain: <input type="text" value="0.5"/>  Decay: <input type="text" value="1.0"/></p> <p><u>Grandmother cells</u></p> <p>Threshold: <input type="text" value="0.5"/>  Gain: <input type="text" value="0.5"/>  Decay: <input type="text" value="1.0"/> ?</p>	
<p><b>Dulink Parameters</b> ?</p> <p>Initial Jij (w/Tij): <input type="text" value="0.1"/></p> <p><input type="radio"/> Update Alg 1  <input type="radio"/> Update Alg 2  <input checked="" type="radio"/> Update Alg 3</p>	
<p><b>Convergence Criteria Parameters</b></p> <p>Activity Threshold: <input type="text" value="0.5"/> ?  Pattern Threshold Delta: <input type="text" value="0.001"/>  Pattern Threshold Iteration Window: <input type="text" value="20"/></p>	
<p><b>Architectural Options</b> ?</p> <p><input type="radio"/> ILTMI      <input type="radio"/> Directed Edges  <input checked="" type="radio"/> CBTMI      <input checked="" type="radio"/> Undirected Edges</p> <p>-----</p> <p>TMI Scaling Exponent: <input type="text" value="80"/>  TM Scaling Exponent: <input type="text" value="4"/>  TGM Scaling Exponent: <input type="text" value="20"/></p> <p>-----</p> <p>Input Plane Spotlight Size: <input type="text" value="2"/></p> <p>-----</p> <p>Target Neural Mixing Factor: <input type="text" value="0.001"/></p>	
<p><b>Simulation Parameters</b> ?</p> <p>Euler Step Size: <input type="text" value="1.0"/>  Random Number Seed: <input type="text" value="0"/></p> <p><input checked="" type="checkbox"/> Use files in Default Input f  <input checked="" type="checkbox"/> Use same files as last run  <input type="checkbox"/> Console to Logfile</p>	
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

### DyLink Update Algorithm 1 Configuration

As implemented, this algorithm is for use only with BINARY neurons. Use with graded neurons will not cause a crash but will also not provide the desired results.

Update Constant:

OK

### DyLink Update Algorithm 2 Configuration

$\alpha$ :  Decay term coefficient.  
 $\kappa$ :  Correlation term coefficient.  
 $\gamma$ :  Normalization term coefficient.  
(Not implemented)

$$\dot{J}_{ij} = -\alpha(J_{ij}) + \kappa \left( \frac{T_{ij}}{\sum_j T_{ij}} * \sigma_i * \sigma_j \right)$$

OK

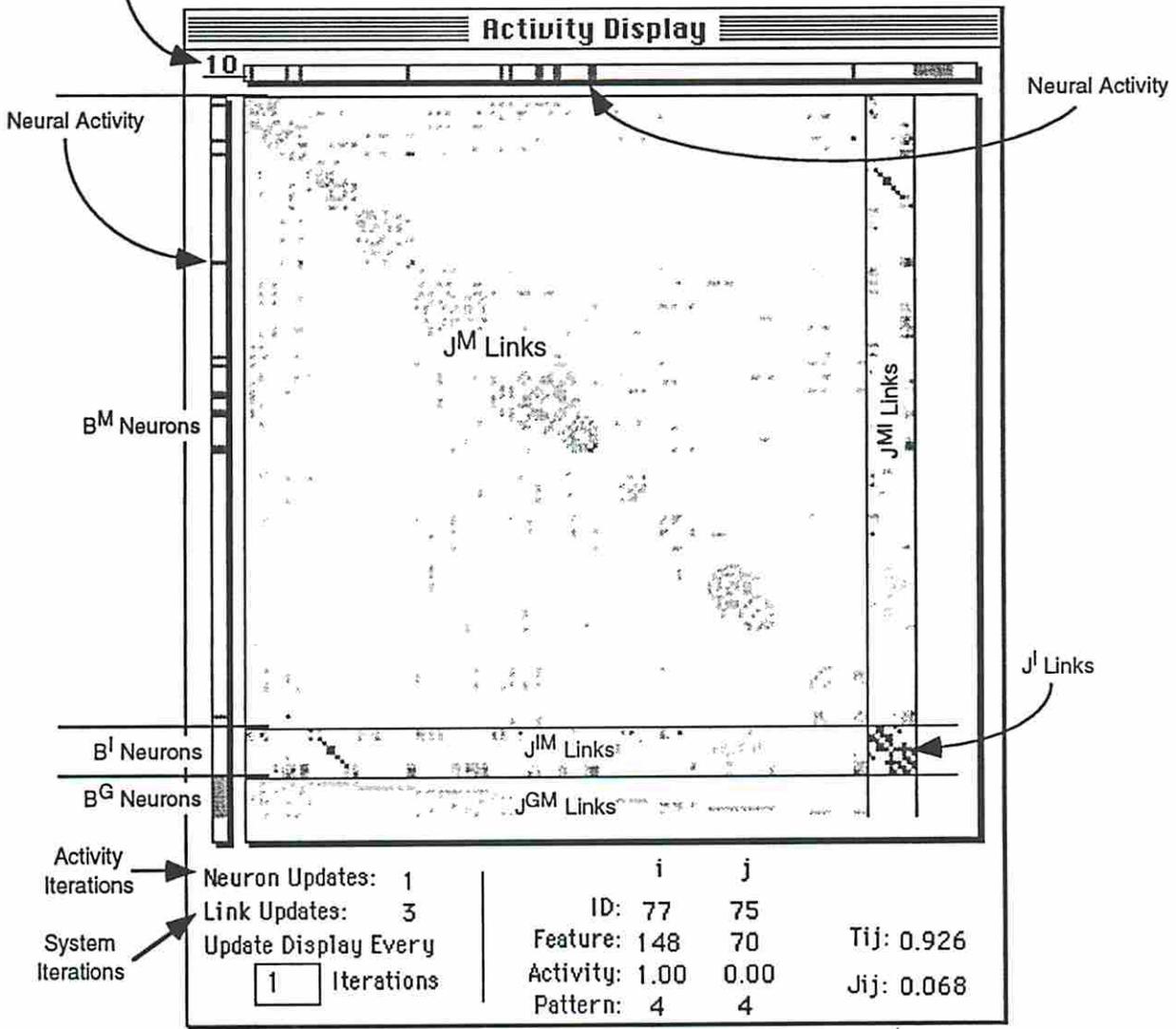
### DyLink Update Algorithm 3 Configuration



D1:  Positive Delta J Coefficient  
D2:  Negative Delta J Coefficient  
P:  Peak Value of Inverted Parabola

OK

Input plane hot neuron ID  
(underline indicates dwell)



Data on Neurons i and j  
(as selected by mouse)

Data on T<sub>ij</sub> and J<sub>ij</sub>  
(as selected by mouse)

## References

- C. Anderson, "Pyramids in Machine Vision at JPL", First International Symposium on Measurement and Control in Robotics, pp F1.3.1-F1.3.4, Houston, June 20-22, 1990.
- M. Arbib, "The Metaphorical Brain 2: Neural Networks and Beyond", John Wiley and Sons, 1989.
- F. Atteneave, "Some Informational Aspects of Visual Perception", *Psychology Review*, Vol. 61, pp. 183-193, 1954.
- R. Bolles and R. Cain, "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method", *International Journal of Robotics Research*, Vol. 1, No. 3, pp. 57-82, 1982.
- E. Bienenstock and C. von der Malsburg, "A Neural Network for Invariant Pattern Recognition", *Europhysics Letters*, Vol. 4, No. 1, pp. 121-126, 1987.
- J. Buhmann, J. Lange, C. von der Malsburg, "Distortion Invariant Object Recognition by Matching Hierarchically Labeled Graphs", *International Joint Conference on Neural Networks*, pp. 155-159, Washington DC, 1989.
- J. Buhmann, M. Lades, C. von der Malsburg, "Size Distortion Invariant Object Recognition by Hierarchical Graph Matching", *International Joint Conference on Neural Networks*, pp. 411-416, San Diego, 1990.
- J. Buhmann and H. Kühnel, "Vector Quantization with Complexity Costs", preprint, submitted to *IEEE Transactions on Information Theory*.
- D. Burr, M. Morrone, and D. Spinelli, "Evidence for Edge and Bar Detectors in Human Vision", *Vision Research*, Vol. 29, No. 4, pp. 419-431, 1989.
- T. Cornsweet, "Visual Perception", Academic Press, 1970.
- F. Crick, "Function of Thalamic Reticular Complex: The Searchlight Hypothesis", *Proceedings of the National Academy of Science*, Vol. 81, pp. 4586-4590, 1984.
- J. Daugman, "Two-dimensional Spectral Analysis of Cortical Receptive Field Profiles", *Vision Research*, Vol. 20, pp. 847-856, 1980.

- J. Daugman, "Uncertainty Relation for Resolution in Space, Spatial Frequency, and Orientation Optimized by Two-dimensional Visual Cortical Filters", *Journal of the Optical Society of America*, Vol. 2, No. 7, pp. 1160-1169, 1985.
- J. Daugman and D. Kanne, "Image Statistics, Gases, and Visual Neural Primitives", *International Conference on Neural Networks*, Vol. IV, pp.163-175, Washington DC, 1987.
- J. Daugman, "Complete Discrete 2D Gabor Transforms by Neural Networks for Image Analysis and Compression", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 36, No. 7, pp. 1169-1179, July, 1988.
- J. Davidoff, "Cognition Through Color", MIT Press, 1991.
- K. Flaton and S. Toborg, "An Approach to Image Recognition Using Sparse Filter Graphs", *International Joint Conference on Neural Networks*, Vol. 1, pp. 313-320, June 1989.
- K. Flaton, "Object Recognition by Dynamical Link Graph Matching", *SPIE Conference on Applications of Neural Networks*, Orlando, April 1991.
- K. Fukushima, "Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition", *Neural Networks*, Vol. 1, No. 2, pp. 119-130, 1988.
- D. Gabor, "Theory of Communication", *J.I.E.E.*, Vol. 93, pp. 429-459, 1946.
- R. Gonzalez and P. Wintz, "Digital Image Processing", 2<sup>nd</sup> edition, Addison-Wesley, 1987.
- S. Grossberg and E. Mingolla, "Neural Dynamics of Surface Perception: Boundary Webs, Illuminants, and Shape from Shading", *Computer Vision, Graphics, and Image Processing*, pp. 116-165, 1987.
- R. Hecht-Nielson, "Neurocomputing", Addison-Wesely, 1990.
- D. Hubel and T. Wiesel, "Receptive Fields and Functional Architecture in Two Nonstriate Visual Areas (18 and 19) of the Cat", *Journal of Neurophysiology.*, vol. 28, pp. 229-289, 1965.
- D. Hubel and T. Wiesel, "Sequence Regularity and Geometry of Orientation Columns in the Monkey Striate Cortex", *J. Comput. Neurol.*, vol. 158, pp. 267-293, 1974.
- Hughes Aircraft Company, "Binary Optic Graph Matching Artificial Neural Network", (Proposal), Ref. No. G8984, FP-87-71-565, March, 1989.
- J. Jones and L. Palmer, "The Two-Dimensional Spatial Structure of Simple Receptive Fields in Cat Striate Cortex", *Journal of Neurophysiology*, Vol. 58, No. 6, pp. 1187-1258 , December, 1987.

- A. Kalvin, E. Schonberg, J. Schwartz, and M. Sharir, "Two-Dimensional, Model-Based Boundary Matching using Footprints", *International Journal of Robotics Research*, Vol. 5, No. 4, pp. 38-55, 1986.
- T. Kohonen, "Self-Organization and Associative Memory", Springer-Verlag, 1988.
- P. Kuner, "Matching of Attributed and Non-Attributed Graphs by Use of a Boltzmann Machine Algorithm", *First IEE International Conference on Artificial Neural Networks*, London, pp. 369-373, 1989.
- Y. Lamdan and H. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme", *IEEE International Conference on Computer Vision*, pp. 218-249, Tampa, Florida, December, 1988.
- M. Lades, J. Vorbrüggen, J. Buhmann, W. Konen, C. von der Malsburg, and R. Würtz, "Distortion Invariant Object Recognition in the Dynamic Link Architecture", preprint, submitted to *IEEE Transactions on Computers*, 1991.
- B. Manjunath and R. Chellappa, "A Unified Approach to Boundary Perception: Edges, Textures, and Illusory Contours", preprint, submitted to *IEEE Transactions on Neural Networks*, 1991.
- M. Mishkin, L. Ungerleider, and K. Macko, "Object Vision and Spatial Vision: Two Cortical Pathways", *Trends in Neurosciences*, Vol. 6, pp. 414-417, 1983.
- B. Parvin and G. Medioni, "A Dynamic System for Object Description and Correspondence", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 393-399, June 1991.
- T. Poggio, V. Torre, and C. Koch, "Computational Vision and Regularization Theory", *Nature*, Vol. 327, pp. 314-316, 1985.
- W. Rasband, "Image", National Institutes of Health, image processing public domain program for Macintosh II.
- K. Reiser, "Learning Persistent Structure", Doctoral Thesis, University of Southern California, September 1991.
- D. Sagi and B. Julesz, "'Where' and 'What' in Vision", *Science*, Vol. 228, No. 6, pp. 1217-1219, 1985.
- R. Shepard and J. Metzler, "Mental Rotation of 3-Dimensional Objects", *Science*, Vol. 171, pp. 701-703, 1971.
- W. Tackett and W. Lincoln, "A Hierarchical Self-organizing Neural Network Architecture: Theory and Implementation", *Hughes Internal AIMS-ANN Engineering Notebook*, Vol. 1, Sect. 4.6, 1991.

- A. Treisman and G. Gelade, "A Feature-Integrated Theory of Attention", *Cognitive Psychology*, Vol. 12, No. 1, pp. 97-136, 1980.
- A. Treisman, "Handbook of Perception and Human Performance", John Wiley and Sons, 1986.
- J. Tsotsos, "Analyzing Vision at the Complexity Level", *Behavioral and Brain Sciences*, Vol. 13, pp 423-469, 1990.
- J. Ullmann, "An Algorithm for a Subgraph Isomorphism", *Journal of the ACM*, Vol. 23, No. 1, pp. 31-42, 1976.
- C. von der Malsburg, "The Correlation Theory of Brain Function", *Internal Report 81-2*, Department of Neurobiology, Max Planck Institute for Biophysical Chemistry, Göttingen, 1981.
- C. von der Malsburg and E. Bienenstock, "A Neural Network for Retrieval of Superimposed Connection Patterns", *Europhysics Letters*, Vol. 3, No. 11, pp. 1243-1249, 1987.
- D. Willshaw and C. von der Malsburg, "How Patterned Neural Connections Can Be Set Up by Self-organization", *Proc. R. Soc. Lond. B.*, Vol. 194, pp. 431-445, 1976.
- H. Wilson and J. Bergen, "A Four Mechanism Model for Spatial Vision", *Vision Research*, Vol. 19, pp. 19-32, 1979.