A study of Artificial Neural Networks for Motion-based Recognition

Psarrou, Alexandra

ISSN 1369-1961

Department of
Computer Science

Technical Report No. 732

A Study of Artificial Neural Networks for Motion-based Recognition

Alexandra Psarrou

QUEEN MARY
AND WESTFIELD COLLEGE
UNIVERSITY OF LONDON

January 1996

A Study of Artificial Neural Networks for Motion-based Recognition

by

Alexandra Psarrou

Queen Mary and Westfield College

SUBMITTED TO THE
DEPARTMENT OF COMPUTER SCIENCE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

at

University of London

ABSTRACT

Motion-based recognition is the approach taken for the recognition of moving objects or their activities based on direct information from the patterns of motion they exhibit in a sequence of images. This requires (a) the extraction of appropriate information from the images, (b) the use of this information in the creation of motion models, and (c) the matching of the information extracted from the images with that of the models.

This work explores a unified approach to motion-based recognition, where artificial neural networks are used for both the extraction of information from a sequence of images and the creation of motion models based on the extracted information. Through our study we evaluate the suitability of artificial neural networks for motion-based recognition and identify the architectures for performing specific tasks. In addition, as the operations required for motion-based recognition are computationally expensive we explore the parallel implementation of artificial neural network architectures on a SIMD machine.

To extract information from a sequence of images we use artificial neural networks to determine the discontinuities in an optic flow field and compute the centroid of objects. Subsequently, we show how artificial neural networks can be used to create non-linear motion models that are invariant to translation, rotation and spatial scale. Such motion models can be used to (a) predict and track the positions of moving objects in a sequence of images and (b) verify whether the observed movement in an image sequence corresponds to the motion model.

In particular, this thesis includes experimental results on the ability of (a) linear resistive networks and Hopfield models to compute the optic flow field from a sequence of images and determine discontinuities in the computed field, (b) feed-forward networks to learn from examples how to compute the centroid of objects and (c) partially recurrent neural networks to create motion models based on example sequences of geometrical positions of a moving object. To test the former we compute the apparent 2D motion produced by the movement of a rigid object using a hybrid parallel model. This model combines an algorithm for computing a smoothed motion field with the detection of motion discontinuities using a Hopfield model. The centroid of a class of non-rigid objects is computed from image intensities using feed-forward neural networks. The Hopfield models and the feed-forward networks are both implemented on a parallel SIMD machine. Finally, models of motion trajectories are built using certain variations of partially recurrent networks and the effects of memory and data representation in tracking object locations are examined.

Acknowledgments

I would like to thank my first supervisor Hilary Buxton for the trust she showed in me, and the advice and encouragement she has given me over the last seven years. Her help and constant support even after she left Queen Mary and Westfield College have been decisive in completing my thesis.

I am also grateful to my second supervisor Heather Liddell. Her encouragement and support especially during the last year have been invaluable. Without her precious comments on many drafts of this thesis, it would be impossible to read.

Special thanks to Shaogang Gong. Our long discussions on my thesis have been a source of strength and inspiration. I am especially grateful for his comments on the structure of my thesis and the numerous drafts. His help with the LaTeX layout and the postscript files is also greatly appreciated.

Thanks to Dennis Parkinson for carefully reading my thesis and his valuable comments on the work of Chapter 4.

Thanks to Stephen McKenna and Nick Lambrou who proofread this thesis with care and made critical comments.

Thanks to Nick Walker who provided the images for Chapter 5; Daniel Zicha for our discussion on the behaviour of cancer cells; Jan Wysocki who helped me with capturing images on DAP.

The support of the School of Computer Science at the University of Westminster is greatly acknowledged. I am especially grateful to the Head of the School, Peter Morse, and the Head of the AI Division, Vassilis Konstantinou, for awarding me a half term sabbatical in order to work on my thesis.

Special thanks to Vassilis Konstantinou for his long term help, support and friendship. Without him carrying my heavy administrative load this term at the University of Westminster, it would be impossible to complete this thesis.

The final thanks are to my brother for his encouragement and to my parents who supported me over the years and made it possible for me to be here.

This work was conducted under the sponsorship of a SERC grant. Their support is greatly acknowledged.

Contents

Chapter 1

Introduction

1.1 Motion-based Recognition: The Perception of Visual Motion

From the evolutionary point of view the perception of physical motion is of decisive importance. The concept of a motionless animal in a totally static environment has hardly any biological significance [63]. In many lower animals, efficient perception of moving objects seems to be the most essential visual function. A frog or a chameleon, for example, can perceive and catch its prey only if the prey is moving. A motionless fly, even within easy reach, goes quite unnoticed [64]. Evidence for a similar dependence on changes in the visual stimulus pattern can also be demonstrated in man. Motion perception helps us recognize different objects and their motion in a scene, infer their relative depth or their rigidity. We have the ability to recognize a person walking at a distance from his or her gait, a particular dance step and flying birds, even though they are all made up of a complex sequence of movements [18]. Even when objects are standing still, their images on the retina are not static since the eyes and the head are never entirely still. It is therefore not surprising that a growing body of evidence has accumulated to suggest that the visual system has a special sensitivity to moving images and that the eye has evolved to function essentially as

a motion-detecting system [15, 60, 63].

The perception of motion has been studied extensively in psychology. From as early as the beginning of this century, Wertheimer, among other Gestalt theorists, investigated the phenomenon of apparent motion [142]. In particular, he discovered that the alternative exposure of two stimuli at an optimum rate and distance is perceived as a continuous translation of the retinal image. In this case, what is seen is not a moving stimulus, but simply movement *per se*. This experience of pure movement is called the *"phi"* phenomenon and arises as the result of temporal and spatial relationships between stimuli [42]. Wertheimer and his colleagues and followers also studied the effect of motion in spatial organization and demonstrated several principles by which moving stimuli organize themselves. Examples are the perception of grouping produced by the common or relative motion of stimuli.

More recently, in the 1970's, motion perception was studied by Johansson and his colleagues [62, 63, 64]. They studied the perception of complex light patterns produced by the movement of articulated bodies as shown in Figure 1.1. Johansson referred to the movement of human or other animals as *biological motion*. He was particularly interested in studying the visual interpretation of motion patterns of human bodies when the pictorial form aspect for these patterns was removed. Johansson carried out his studies by first recording the motion patterns produced by illuminated points placed on the main joints of actors [1] and then filming their movements in the dark. As only the illuminated points on the actors were visible, the perceived motion patterns did not carry any information about the figure of the human bodies and gave the impression of *Moving Light Displays* (MLDs). Later, subjects that did not know about the nature of the experiments were told to interpret the context of these films. When the actors were stationary, subjects found that the patterns of points

[1] The illuminated points were placed on the hips, knees, ankles, shoulders, elbows, and wrists of the actors [63].

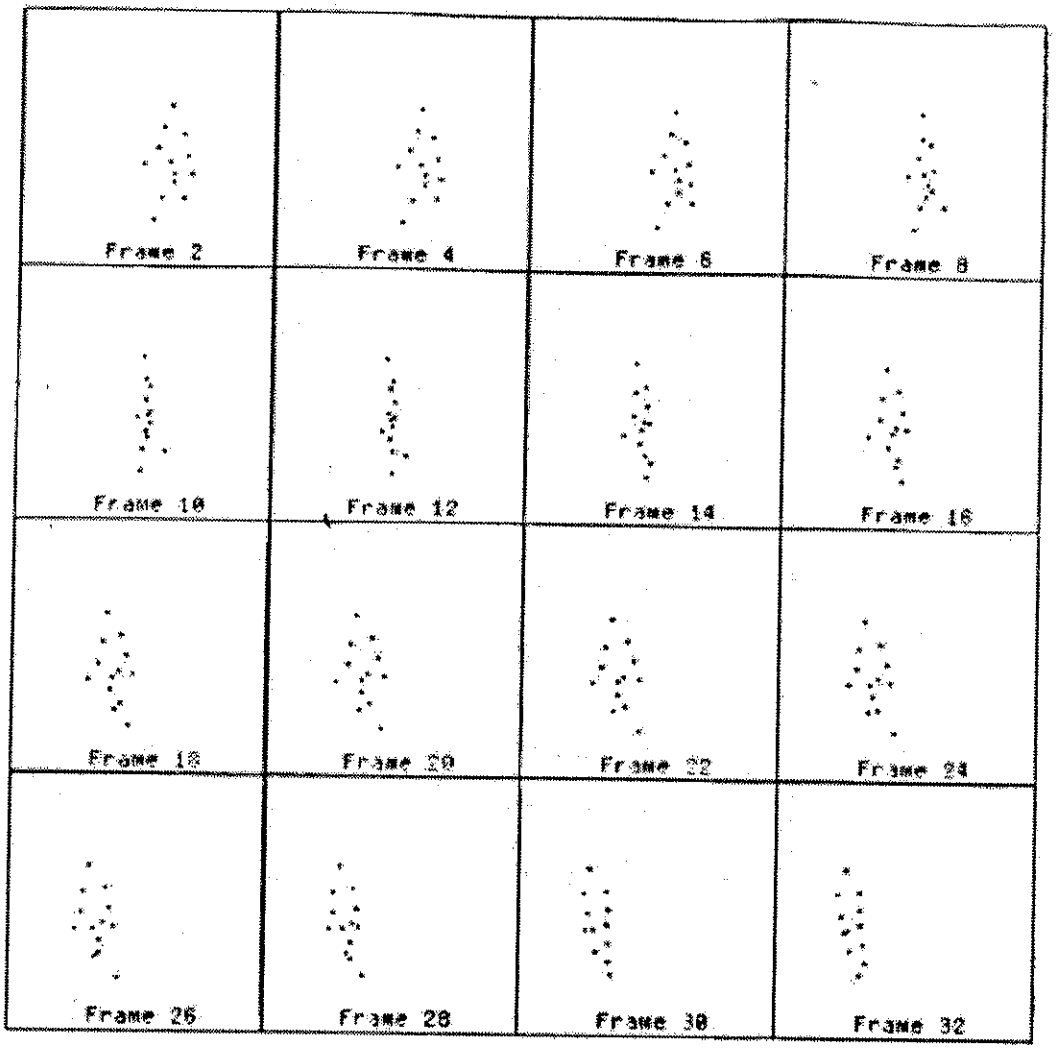


Figure 1.1: *Complex light displays produced by the movement of a human body [117].*

were meaningless, but as soon as the actors moved, their human shapes could be perceived, and the nature of their movement, such as walking or running, accurately described. More rigorous experiments conducted by Cutting and Kozlowski [22, 78] showed that the subjects could recognize the gender of a person and even the gait of a friend based solely on the motion of the illuminated points. The phenomenon of recognising behavioural patterns and activities based purely on 2D visual motion is known as *Motion-based Recognition* [134].

Performance of motion-based recognition by machine vision systems would be useful in many applications. For example, such systems would be essential in realising effective and efficient automated visual surveillance [40], face recognition [41, 115] and biomedical image sequence understanding [113]. However, machine motion-based recognition has received relatively little attention in the literature. Influenced by David Marr's [85] theory of computational vision, most research on motion understanding has in fact been focused on issues concerned either with structure-from-motion, where a sequence of images is used for the reconstruction of the 3D geometric structure of objects and scenes [96] or 3D model-based recognition, where an image sequence is used to verify the existence of objects based on known 3D shape models [52, 75, 84, 148, 149]. Few studies have considered specific aspects of motion-based recognition computationally [23, 39, 41, 70, 88, 105, 110, 111, 113, 115, 151].

In the following two sections of this chapter we will see how analysis of psychological experiments can provide an insight into the nature of the mechanism involved in the perception and the interpretation of 2D visual motion and suggest possible computational models.

1.1.1 Psychological Explanations

A number of psychological studies on motion perception have been concerned with the following issues [22, 32, 33, 35, 78, 124, 128, 133, 138, 142]:

1. Whether the perception of motion depends on establishing correspondence between different retinal snapshots or on extracting continuous spatio-temporal correlations.
2. Whether the perception of motion depends on the recovery of the underlying 3D geometric structure of moving objects, or if it can be inferred from 2D apparent visual motion information.
3. Whether the perception of motion is independent of cognitive control.

Regarding the perception of motion as that of establishing correspondence between different snapshots can be partly justified by the experiments of the Gestalt theorists, where with the "phi" phenomenon they showed that it is possible to perceive movement in the absence of continuous translation [142]. However, J.J. Gibson in his theory of *direct perception* categorically denied that visual perception involves computation or inference from static retinal snapshots [32, 33]. Based mainly on the analysis of the perception of textured outdoor scenes, he showed that any movement on the part of the observer or the objects in the world produce changes in the entire retinal image that follow patterns of *optical flow.* Gibson claimed that from the spatio-temporal patterns of optical flow, we can directly specify the 3D structures in the world, and detect the patterns of an observer's movement.

Physiological studies on the human vision system favour the assumption that the starting point of vision is a spatio-temporal pattern, rather than the acquisition of static snapshots [60]. However, there is no evidence of exact neurophysiological and cognitive processes that are applied in the recognition of objects and their movement. Because the human vision system has to perform a very wide range of different tasks it is possible that it uses multiple sources of information which may be processed in parallel by different but interacting modules [35]. Cutting and Kozlowski provided evidence that the recognition of objects and their movement may not require the recovery of their structure [22, 78]. Analysis of their experiments showed that the subjects who correctly recognised the gait of their friends used criteria such as "bounciness" and "arm swing". Conversely, subjects that used the height of their friends as a discriminating factors of the MLDs did badly in recognising their friends. This suggests that (1) our recognition of MLDs may be based on motion information rather than the structure of the moving objects and (2) we make use of previous knowledge, therefore suggesting a degree of *cognitive control* in the perception of visual motion.

Experiments conducted on motion perception of MLD's by Sumi [128] and Goddard [35] suggest similar conclusions. Sumi showed that MLDs presented upside-down are usually not recognised but informal experiments conducted by Goddard found that people seem to be much better at interpreting inverted MLDs when the dots are connected. The connection of the dots produces a stick-figure that can be perceived as the structure a human body. Goddard argued that since subjects can interpret the inverted stick-figures and not the inverted MLDs, it cannot be the case that MLD interpretation is performed by an input-driven structure-from-motion process. This is because the structure of the MLD is simply the connectivity of the moving stick-figure underlying the MLD, which does not depend on whether the image is rotated in its own plane. Therefore, if the visual system is computing a bottom-up structure-from-motion transform, it is one that is not powerful enough to derive the full structure of a MLD [35]. This suggests that: (1) even if the structure was extracted additional information is required for the interpretation of MLDs, (2) this information could be stored models of movement.

1.1.2 Computational Models

Some researchers in machine vision have produced computational models based on the psychological explanations outlined. For example, Ullman proposed a model for establishing correspondences between static retina snapshots and recovering the structure of moving objects [133]. Johansson, on the other hand, argued for a motion perception model that depends on the extraction of hierarchical components of projective invariances from the optical flow field [62, 63]. However, although enlightening and original at the time, their models failed to account for sufficient examples of motion perception.

A successful model for motion perception needs to explain not only "what" information is computed and "how" it is computed, but "why" it is computed. More recent

work in active vision has taken the view that vision is highly selective and purposive and is defined by the tasks an organism needs to perform [7, 11]. With this approach, interpretation of a dynamic scene does not necessarily require the reconstruction of objects and the scene. It can be achieved from the understanding and interpretation of the moving patterns of the objects. This involves having expectations of the moving patterns of the objects.

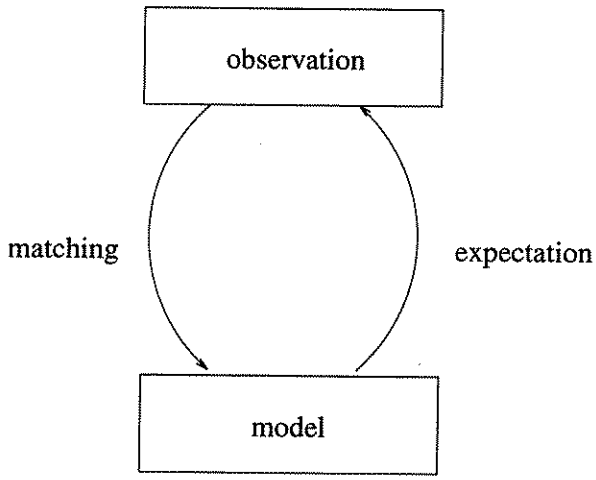


Figure 1.2: *A closed loop representation of visual motion perception.*

Motion perception in the framework of active vision can be seen as the closed loop process shown in Figure 1.2. It consists of (1) an observation of motion, (2) a *motion model*, (3) a process that matches an observation with the model and (4) a process that delivers an expectation of the motion or its interpretation. In this framework each process can have the following computational interpretations:

Observation An observation can be:

- The position of an object in the scene.
- The photometric temporal changes in the image.
- The optic flow induced by a movement in the scene.

Motion model The model of movement depends on the nature of the observation extracted from the scene. Accordingly a motion model can be:

- A sequence of the positions of a moving object defining a motion trajectory. Computationally it can be described as a sequence of coordinate values or curvature measurements.
- A sequence of the photometric temporal changes in the image.
- A sequence of the changes in the pattern of optic flow in the scene.

Expectation An expectation usually has the same representation as the observation or it can simply be the interpretation of the motion. A useful consequence of the expectation process is that it drives our attention and allows effective tracking of the position of the objects or other dynamic changes.

Matching This is a comparison between the current observation and the past expectation of the model.

The closed loop representation of visual motion perception provides an efficient and effective computational model for motion-based recognition, where the expectation process is used for tracking and recognising moving objects or their activities.

1.2 A Neural Network Approach

The motion perception models outlined in section 1.1.2 are computationally demanding. The typical computations involved, such as tracking and measuring optic flow, are costly and time consuming, whilst human vision can separate one person's "walking" or "running" from another with little difficulty and almost immediately. What is the discriminating factor? How can we build models of motion that describe such large classes of movements?

The poor performance of traditional rule-based models, and the apparent effectiveness of biological vision systems, has driven researchers to look to the brain as an inspiration for some successful methods for analysing images. Biological vision systems have been studied both in neurophysiology and psychology with computational models being proposed [51, 85]. Such models are often complex with several levels of processing and with different representations at different scales. Some of these models have been essentially neural in nature, and there is a very strong relationship between computational models of vision and neural networks [51]. The difficulty of discovering any explicit rules that govern such an intuitive and unconscious human capability inevitably draws research in the study of methods for learning implicit rules from examples.

It was the re-emergence of the neural computation paradigm in the late 1980's that motivated the work described in this thesis. In particular we were interested in exploiting the characteristics of artificial neural networks [2] for robust and fast computation in performing motion perception tasks such as the tracking and recognition of moving objects and their activities. For this, we were seeking a unified approach, where neural networks are used for (1) extracting observations, (2) building motion models through examples, (3) matching observations with motion models and (4) delivering expectations.

1.3 Outline of the Thesis

In this thesis we seek to explore the potential of neural networks in the perception of motion and answer the following questions:

[2] In the remainder of this thesis we will refer to "artificial neural networks" as *neural networks*, but they should not be confused with their biological counterparts to which they are an oversimplification.

- How effective are neural networks in the extraction of visual observations? We address particularly the issue of locating distinctive features and computing optic flow.

- How effective are neural networks in building motion models and providing expectations on the geometric position of feature observations?

The data used throughout this thesis is largely synthetic as the emphasis of this study was not on delivering a working system but rather, to investigate the limits and potential of modelling motion-based recognition by neural networks. To start with, synthetic images are developed to compute optic flow. Next a set of medical images of biological cells is used for the extraction of features. Motion models are then built by using simulated motion sequences for tracking the position of objects [3].

In Chapter 2, the implications of the dynamic nature of motion in the construction of motion models is discussed, followed by a presentation of the data representation and model matching techniques used in the interpretation of visual motion. The existing algorithms are critically reviewed and finally the need for investigating the use of neural networks in the modelling of the process of visual motion perception is justified.

After an introduction to the paradigm of neural computation, Chapter 3 examines the network architectures that are relevant to the process of motion-based recognition. As matching, and the derivation of expectation are intrinsic characteristics of neural networks, the networks that can be used for the computation of observations and the construction of motion models are presented. In particular, we look at the extraction of features from images and concentrate on the difficulties presented by cytological images. A discussion on an algorithm for the computation of optic flow and the

[3]More recently, we have built motion models for the dynamic recognition of faces by using sequences of face images [41, 115].

modeling of general time-varying information concludes this chapter.

In Chapters 4 and 5 we concentrate on the use of neural networks for the computation of observations from image sequences. Optic flow can be used for estimating the centroid of a moving object [110], or for creating motion models of the changes of the apparent 2D motion in the image sequence. Alternatively, the location of non-rigid objects can be estimated by the extraction of invariant features.

In Chapter 4 the mapping of an optic flow algorithm on a Hopfield network of linear processing elements is presented and the detection of motion discontinuities using continuous valued line processes modelled on a Hopfield model is discussed. Finally, their implementation on the AMT DAP (now manufactured by CPP), an SIMD parallel computer is presented.

In Chapter 5 the extraction of the centroid of cancer cells from a set of time-lapse cinemascopic cytological images is described. In particular, the potential of two feed-forward networks, which we refer to as "all connected" and "locally connected", for fulfilling this task is examined and their parallel implementation is presented.

The centroid of objects tracked in time can form motion trajectories. In Chapter 6 the requirements of a motion model based on motion trajectory information is presented and appropriate data representations and neural network architectures for its construction are proposed. In particular we show how a partially recurrent network can "learn" motion models that correspond to trajectories whose curvature varies along their length and have sharp curvature discontinuities. Finally, the effectiveness of the motion models is discussed by evaluating their performance under the conditions of spatio-temporal scale and positional variance.

Chapters 4, 5 and 6 reflect the fact that the approach taken in this work towards motion-based recognition is a unified one. In particular, Chapters 4 and 5 describe

how Hopfield models and feed-forward networks can be used for the extraction of information from a image sequence and Chapter 6 describes how partially recurrent neural networks can be used to create motion models based on such information. However, it should be emphasised that, although the work described in Chapters 4, 5 and 6 correspond to modules of a unified motion-based recognition system, the implementation in each chapter is independent.

Finally, in Chapter 7, the contributions of this thesis in the application of neural networks in motion-based recognition and their implementation on an SIMD machine are stated and we conclude with a discussion on future work on motion-based recognition using neural networks.

1.4 Notes on the Time-scale and the Aim of the Thesis

Most of the work described in this thesis was carried out in the period of 1988-1992. Two important events of this period were (1) the re-emergence of the paradigm of neural computation [123] and (2) the use of parallel architectures for the efficient implementation of computationally expensive processes [37, 118]. As the need for alternative computational paradigms for computer vision applications was evident, we looked at the benefits neural networks have to offer in this area coupled with their implementation on an SIMD machine. The work on Hopfield models was carried out in the period of 1988-1989, the work on the feature extraction using feed-forward neural networks was carried out in the period of 1990-1991 and most of the work on partially recurrent neural networks for the construction of motion models was carried out in the period of 1991-1992. From todays perspective, it is the work on motion models described in Chapter 6 that is really novel.

Chapter 2

Motion Models for Motion-based Recognition

Motion-based recognition is a general approach that favours the use of visual motion information for the purpose of recognition. However, the emphasis on visual motion as a means of quantitative reconstruction of world geometry [95] has tended to obscure the fact that motion can also be used for recognition. As a result, the analysis of a sequence of images for the recognition of continuous motion, like "walking", has in the past been driven by structural models of the moving object [3, 52, 102]. For example, in Hogg's work the output description of each frame of an image sequence depicting a walking person is an instance of a 3D model of a class of human walkers [52]. An ordered set of such instances creates a spatio-temporal description of the image sequence that facilitates an interpretation of the movement. However, this approach is computationally very expensive as it requires the delineation and labelling of objects and recognition of their subparts in each frame of the sequence. Furthermore, in many cases the search for the location of the model in each frame is driven only by information from the previous frame and by constraints on the structure of the object.

It is possible, however, to use motion information as a means of recognition directly. For example, the orientation and displacement of movement to derive motion models of the trajectories of moving objects has been suggested [38]. These models

have been used to drive the analysis of image sequences by delivering expectations of an object's positions. Such models can find numerous applications in automatic surveillance where motion recognition can be used to predict the path of moving vehicles [39], to remove ambiguities between possible or allowable types of motion from a non-desirable motion in a particular scene, or to distinguish between stationary and moving obstacles in planning the path of a moving piece of machinery [44]. Similar models can also be used to drive the analysis in a sequence of cytological images and interpret the dynamic behaviour of cells. For example, the dynamic characterisation of a class of cancer cells can be performed by assessing, from time-lapse cine-microfilms, the shape and speed of morphotype changes as well as the rate and directionality of cell migration [113, 137, 153]. Furthermore, models that use optic flow information have also been proposed for lip-reading [88] and the recognition of activities such as wind-blown trees and ripples of water [110].

Motion can also be recognised by modelling other time varying information, for example the photometric temporal changes in a sequence of images. Such techniques have been applied to the recognition of human activities [151], speech recognition [70, 105], recognition of gestures [23] and faces [41, 115]. In an alternative approach, Baumberg and Hogg [12, 13] have been looking into the automatic extraction and modelling of closed curve contours that correspond to moving objects in a scene. They have been successful in automatically grouping closed curves that correspond to the same instance of movement and associate each group with a direction of movement. This approach can be extended to the recognition of movements by relating a sequence of shape changes of the closed 2D curves to a sequence of motion patterns.

In order to use motion information successfully in the recognition of moving objects or their activities we need to identify the properties that motion models need to possess. The motion exhibited by a moving object can be described either by the path it follows or by the changes of properties along that path. In this respect the

description of motion is similar to that of the shape of a 2D boundary of an object where each position along the path is indexed by the frame number of the time sequence. However, there are some difficulties in the formalisation of a *motion class* that could be used for the model representation of similar motions. These difficulties are intrinsic to the nature of motions and can be attributed to the following reasons:

1. **Motion is dynamic and uncertain** The representation of motion is susceptible to "noise". This "noise" can arise from factors that are internal to a moving object or from unexpected circumstances where parts of an object's movement are (1) changed in order to adjust to changes that occur in the environment or (2) occluded due to changes occured in the environment.

2. **Motion can be of undetermined extent** Since motion is dynamic, its temporal scale cannot be predicted. Therefore, the recognition of motion cannot be constrained to a temporal window and it may be necessary to partially recognise it by using other already accumulated information.

3. **Motion is spatio-temporal** Not only the spatial location of the movement may change, but also its occurrence in time.

The difficulty of formalising motion suggests that for an effective motion-based recognition, an appropriate motion representation should have the following properties:

1. Allow the creation of models through the presentation of examples.

2. Account for spatial and temporal scale invariance.

3. Account for spatial and temporal position invariance.

In the remainder of this chapter some of the approaches used in the recognition of moving objects or their activities from motion or other time varying information are surveyed by identifying:

- The appropriate representation of motion information necessary in the organisation of motion models;
- The matching techniques employed in the comparison of unknown information with stored motion models.

The motion representation and matching schemes are then evaluated in the light of the properties that they need to possess for an effective motion-based recognition.

2.1 Motion Representation

There are three commonly adopted methods for representing motion information:

1. Trajectory representation
2. Optic flow
3. Photometric temporal changes in a sequence of images.

A diagram depicting different motion representations is shown in Figure 2.1.

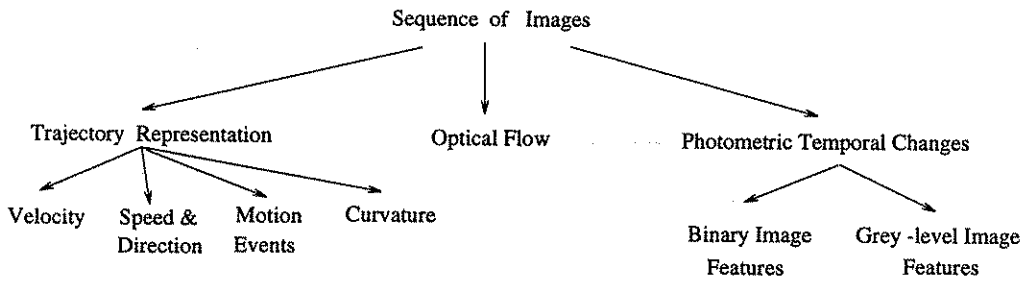


Figure 2.1: *Representation of motion information [18].*

2.1.1 Trajectory Representation

Trajectory representation requires the extraction of the position of significant features, known as tokens, in an image that can be tracked in time. This results in a *motion trajectory* in the image domain, i.e. a sequence of locations (x_i, y_i) for $i = 1 \ldots n$, where n is the number of frames in a sequence. The tokens need to be distinctive enough for easy detection and stable enough in time so that they can be tracked. Trajectories can be parameterised in several ways. For example, Gong and Buxton used discrete relative orientation and speed information along the trajectory of a moving vehicle to model its movement at an airport landing dock [39]. The speed and relative orientation representations of a trajectory are spatially invariant as they are independent of the position or orientation of the motion.

In another example, Gould and Shah parameterised trajectories using velocity and acceleration curves [44]. The velocity and acceleration curves along the trajectories were convolved with a set of second derivative of Gaussian functions with different standard deviations. This transformation smoothes the representation of the trajectories and detects discontinuities in motion at different scales. The position of any motion discontinuity in different scales is described by a curve known as the *Trajectory Primal Sketch* (TPS) curve. Each TPS curve corresponds to an important event in a motion trajectory and a sequence of such curves can be used to define primitive trajectories. Gould and Shah showed that by studying the velocity, acceleration and TPS curves it is possible to discriminate motion characteristics of four types of primitive trajectories: translation, rotation, projectile and cycloid. For example, they showed that the TPS acceleration curves of the rotation and cycloid primitives exhibit a sine or cosine relationship to each other. This property is not shared by a trajectory corresponding to translation. The aim is to express an arbitrary trajectory as a composition of these primitives. This representation of information with respect to scale is also known as *scale-space*. The scale is of temporal nature and the

representation possesses temporal scale invariance, as well as spatial and rotational invariance.

In the trajectory parameterisation described above, absolute values of speed, direction, velocity and acceleration were used. However, absolute values might sometimes be inadequate. In the case of human body motion the absolute velocity of a body part has less significance than the relative velocity between moving parts, and relative joint angles with respect to time carry important information. For example, relative angles can be computed between pairs of points relative to a given axis, or as the joint angle between three points, in each frame. The difference in angle is then determined between successive frames, which can be considered as angular velocities if we assume that the time between each frame is constant. In Goddard, the motion events in the interpretation of Moving Light Displays were described by the angle and the new angular velocity at the joints [35, 36]. The representation of relative motion is invariant to rotation, translation and spatial scale, but it is very difficult to compute as the correspondence between different parts of objects needs to be extracted.

trajectory parameterisation	on-line computation	invariances
sequences of relative orientation and speed	yes	translational, rotational
velocity and acceleration scale-space curves	no	temporal, translational, rotational
angular velocities	yes	rotational, translational, spatial scale

Table 2.1: *Trajectory representation schemes and their properties for motion-based recognition.*

Table 2.1 summarises the invariances possessed by the trajectory parameterisation schemes reviewed above. In this table the property of *on-line computation* is

also included. This indicates whether the data representation can be used for on-line tracking and recognition of moving objects or their activities. Not all applications in motion-based recognition require on-line computation. For example, motion textures such as the ones derived from wind-blown trees are best recognised from past accumulated information. However, in many applications, for example automated visual surveillance [40] and face recognition, [41, 115] on-line tracking and recognition is required.

The scale-space representation of trajectories possesses useful invariances but does not allow for on-line motion-based recognition. On the other hand, the representation of trajectories as sequences of relative orientation and speed possesses only spatial invariances. Combination, however, of the latter representation with an appropriate matching technique can result in temporally invariant and on-line motion-based recognition.

Motion-based recognition of articulated bodies is not considered in detail in this thesis. However, recognition of human bodies or activities using trajectory representation on movements of joints is particularly difficult as it requires the correct pairing of joints. Approaches that use a holistic representation of human bodies are more promising [12, 13, 151].

2.1.2 Optic Flow Representation

The computation of optic flow is based on the spatio-temporal intensity changes in a sequence of image frames. In motion-based recognition, optic flow can be used for estimating the centroid of a moving object or for creating motion models of the changes of the apparent 2D motion in the image sequence.

Polana and Nelson use optic flow for the recognition of motions that are characterised by spatial and temporal uniformity [110]. In particular, they showed that

certain statistical spatial and temporal features derived from approximations to the optic flow have invariant properties and can be used to classify regional activities such as wind blown trees. In their system, they considered features based on the magnitude and direction of the component of the optic flow normal to the intensity gradient. The four types of features chosen were (1) the mean flow magnitude divided by its standard deviation, (2) the positive and negative curl and divergence estimates, (3) the non-uniformity of flow direction and (4) the directional difference statistics in four directions. The feature values were arranged into a vector that was used for classification. This method is computationally expensive but it may provide the stability and invariance needed for a good classification.

Mase and Pentland used optic flow to perform recognition of spoken words from images of lip movements [88]. They observed that the most important features that affect mouth shape relate to the elongation of the mouth, and to the mouth opening. The average optic flow was computed in four windows around the mouth of the speaker, in successive pairs of frames. A principal component analysis was performed and two functions $O(t)$, expressing mouth opening in time and, $E(t)$, reflecting the elongation of the mouth as a function of time, were then created:

$$O(t) = v_b + v_l + v_r$$
$$E(t) = \frac{1}{2}v_a - u_l + u_r$$

where v_a, v_b, v_l, v_r are the vertical components of the flow vector of the upper, lower, left and right windows and u_r, u_l are the horizontal component of the flow in the right and left windows, respectively [18].

$O(t)$ and $E(t)$ were computed in each frame, then smoothed and normalised to a fixed variance. A sampling of these two functions was used to create a vector for input and model comparisons, and a match was established with the model that gave the smallest weighted squared difference.

One problem with optic flow in general, is that it is susceptible to the aperture problem, which only allows the precise computation of the normal flow. Problems also arise with boundary over-smoothing and multiple moving objects where segmentation can be difficult to achieve [18]. Despite all this, optic flow is a very rich source of information and with an appropriate representation can be used in motion-based recognition. However, its on-line use requires an efficient parallel implementation.

2.1.3 Photometric Representation

Another method for recognising moving objects or their motion path is to model their photometric temporal change. However, this kind of representation is sensitive to translation, rotation and scale and requires an accurate segmentation of objects in each frame of the image sequence. If large areas in the images that are used contain significant background rather than object information, then the extracted distribution would "say" more about the statistical properties of the background than that of the objects. Yamato *et al* [151] and Polana and Nelson [111] have used photometric representation to build motion models for representing and recognising activities. On the other hand, Darrell and Pentland [23] and Gong and Psarrou *et al* [41, 115] have used photometric temporal changes to recognise objects from a sequence of images taken from different viewpoints.

Yamato *et al* classified observed low-level image feature sequences into human action categories [151] . In his approach, time sequential binary images expressing human actions are transformed to a sequence of image feature vectors by extracting a feature vector from each image. A mesh feature [1] is used as the image feature. The feature vectors of the sample data set for training are vector quantised and each vector is assigned to a symbol which corresponds to a codeword in the code

[1] A mesh feature as used by Yamato is the number of black pixels/ number of pixels in a neighbourhood [151].

book created by vector quantisation. Consequently the time sequential images are converted to a sequence of symbols. Yamato showed that this representation was able to recognise normalised sequences of tennis strokes, but he did not refer to the limitations of this approach and ways of overcoming them. For example, such a representation depends on the accurate figure-ground segmentation of images and the precise thresholding of the intensity values. Furthermore, the representation does not account for translational, scale and rotational variance.

A different approach for modelling activities using photometric representation was proposed by Polana and Nelson [111]. They used the representation of spatio-temporal cubes to detect activities that are periodic. In their approach, the first part consists of providing the approximated location of the centroid of an object in time. The frames are then aligned with respect to the centroid of the object, such that it remains stationary in time. However, if the object presented any periodic motion, for example a person walking, the motion of the legs and arms remains. This motion would create certain periodic grey level signals over the image, especially around the centroid. The periodic motion can be extracted from the grey level signals using a Fourier transform. The periodic detection is invariant to the magnitude of motion, speed of the activity and is fairly robust to small changes in viewing angles. Their approach has not yet been demonstrated in the recognition of activities. However, this would require the fast computation of the Fourier transform in every pixel in the image, or at least in the area that includes the human body in a sequence. This is computationally very expensive. Another limitation of this approach is that it cannot be used on-line as the Fourier transform requires a complete presentation of the activities.

The two approaches are completely different in the way they use photometric representations for the recognition of activities. Yamato's technique is holistic, whereas that of Polana and Nelson depends on the values of single pixels. However, both tech-

niques have fundamental limitations to their application in the realistic recognition of activities.

Darrell and Pentland [23] presented a new method for learning, tracking and recognising human gestures from a sequence of images. The same approach is also used for the recognition of objects given a set of ordered viewpoints. They have implemented a system which automatically builds a view-based object model and its contextual dependencies while tracking the object through space and time. The model views of an object are built using normalised correlation. The first view is chosen by the user as one of the images from a sequence. The object in the subsequent input images is tracked, and when the correlation score drops below a predetermined threshold, a new model view is created with the current input image. Once all views of an object have been gathered, gesture models are created. A gesture is a set of views over time and is correlated with each stored view of the object and the correlation score plotted for each view with respect to time. Several examples of the same gesture are used, and the mean and variance of the correlation scores with respect to model view m are used to represent the particular gesture g. The gesture models need first to be adjusted to the same sequence length, and this is done through dynamic time warping. To compare a new gesture captured by an image sequence, each frame of the new sequence is correlated with a model view and its score determined. Finally the results with the input gesture are compared with all the gesture models.

Gong and Psarrou *et al* [41, 115] proposed motion models based on eigenfaces [69, 126, 131] to recognise the "temporal signature" of faces. With this approach the variation in a collection of face images is captured by computing the eigenvectors of the covariance matrix of a set of face images. Each face is then represented by a linear summation of the eigenvectors, called the pattern vector. However, as the approach is both scale and viewpoint dependent, current eigenface based face recognition models are limited to registering only single face images [131]. In the work

of Gong and Psarrou the environmental layout and the physical freedom in human head movement, that limits the possible changes of a moving face, are exploited as contextual constraints. They are used to encode any view invariant face features and recognise face images using information from sequences of viewpoint differences. In this system, a moving head from an on-line camera input is first detected and tracked before segmenting and normalising the face images. Then the time sequential images are converted to a sequence of pattern vectors.

Darrell-Pentland and Gong-Psarrou's approaches are similar in that they use a sequence of viewpoints to create models of objects. In the case of Darrell and Pentland, the viewpoints may span large viewpoint differences, which they interpolate, whereas the motion model of Gong and Psarrou, depends on continuous change in the viewpoints. However, in the case of Gong and Psarrou the data representation is smaller as it only requires to store pattern vectors, and therefore is more efficient in matching. The eigenspace approach has also been used by Kirby *et al* [70] to represent a sequences of lip images for the recognition of words by creating "signature" graphs. However, no precise method for recognition was described using such graphs.

2.2 Motion Matching

Once the motion information is extracted and organised, matching between a model and an input needs to be performed in order to recognise the moving object or its activity. Most algorithms use clustering techniques since their models and inputs are encoded by feature vectors, but other methods have also been proposed, including a probabilistic method of Hidden Markov Models (HMMs) [116] and a connectionist method [10, 28]. Another potential approach is that of Bayesian networks described in [16].

With feature vector representation of motion, matching by vector clustering has often been employed. The approach is based on the assumption that similar motion sequences will generate similar feature vectors. Motion models are then built by using any kind of classifier based on vectors and the matching process involves the computation of the distance between an input and a model feature vector. The model that gives the smallest distance is chosen as the most likely representation of the input.

Yamato *et al* took a probabilistic approach to the classification of different motions [151]. They used a sequence of symbols, one per frame, derived from a mesh feature to train HMMs and create motion models for each activity, each HMM corresponding to one model. Matching of an unknown sequence with a model is done through the calculation of the probability that an HMM could generate the particular unknown sequence. The HMM with the highest probability is the one that has most likely generated that sequence. The advantages of an HMM are that it can deal with time-sequential data and can provide time-scale invariability as well as learning motion models from examples. The limitation is that although the *a priori* probability distribution estimation required for modelling information in an HMM can be learned through examples, the analytic determination of the hidden states required is a computationally expensive process. A pair of HMM models was also used by Gong and Buxton [39] in the derivation of motion models using the information on the orientation and displacement of motion trajectories.

Another approach is that of the connectionist network proposed by Goddard [35]. His representation consists of an ordered sequence of events which are coordinated by temporal and motion events. Each event is represented as a node of a digraph, which represents order by the direction of edges and temporal distance by associating a time with each edge. With such a connectionist network, each graph node becomes a processing unit, and each directed edge a link with an associated delay, thus representing time implicitly. In general, the kinematics of a complex object may

be described by a sequence of events that each part of the object undergoes, together with information coordinating these sequences. The sequence of events for a single body part is represented by a set of graph nodes and links, and the coordination between sequences by more links. Thus a hierarchy is used: A low-level feature triggers an event which is sent to the layer above in the hierarchy; a combination of events at that level trigger other events at yet higher levels, and so on until the coordinated sequence of events of a body in motion can trigger one motion model at the highest level. This is the output level which represents the global motion of walking, running or skipping. However, it is not at all clear how such an architecture can be used for learning motion sequences from examples, and how it can adjust to different time scales.

2.3 Neural Computation for Motion Models

The construction of models that fulfill the requirements of motion representation, described on page 15, and allow for on-line computation depends on the effective combination of data representation schemes and the matching techniques employed. Data representations that account for temporal invariances can not be used for on-line motion recognition. From the approaches reviewed, is evident that on-line and temporally invariant motion-based recognition is best addressed by the combination of HMMs and the parameterisation of motion trajectories or feature vector representation of images.

The main disadvantage of the latter approach is the computational cost in determining the hidden states of HMMs. An alternative approach is to use neural network architectures. In this case, however, we propose that neural networks are not seen as a framework for connectionist representation [2] but rather as a frame-

[2]Goddard used a connectionist representation for the interpretation of MLDs [35].

work for optimisation, where the best possible motion model is computed through the presentation of examples. In particular, neural network would be suitable for addressing the problems encountered in motion-based recognition because they possess the following properties:

- They can learn from examples and therefore we do not have to specify explicitly the analytical definitions of the motions of objects.
- They can deal with noisy data, both in creating motion models and in the comparison of unknown data with a model.
- Recurrent neural networks can achieve temporal scale invariance.
- The recognition can be performed on-line and can deduce results from partial information.
- It is possible to build an integrated architecture of cascaded neural networks with modules responsible for the computation of an observation and its prediction (see Figure 2.2).

2.4 Summary

In this chapter we introduced the notion of motion models in motion-based recognition and reviewed data representation schemes and matching techniques for their construction. The motion data representations were based on: (1) absolute or parameterised trajectories, (2) optic flow and (3) photometric temporal changes in a sequence of images, and the most popular matching techniques employed were HMMs and centroid classifiers.

The dynamic nature of motion dictates that an appropriate motion representation should have the following properties:

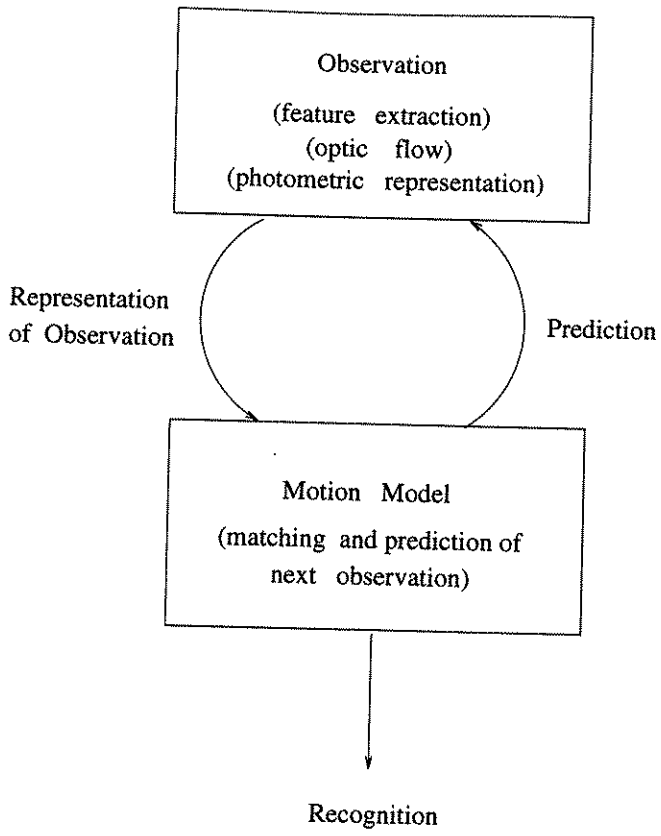


Figure 2.2: *A cascaded network consisting of the observation and prediction modules.*

- Allow the creation of models through the presentation of examples.
- Account for spatial and temporal scale invariance.
- Account for spatial and temporal position invariance.

Furthermore, many applications in motion-based recognition require on-line computation. This excludes the use of temporally invariant motion representation schemes in the construction of motion models. Alternatively, temporal invariance in motion-based recognition can be addressed by the matching technique employed. A combination of such a technique with a motion representation scheme that accounts for translational, rotational and spatial scale invariance can provide a temporally invariant, on-line motion model.

We concluded this chapter by indicating that the requirements of motion models can be addressed by using neural network architectures and suggested how a cascaded form of neural networks can be used for building an integrated system for motion-based recognition.

In the next section we introduce the paradigm of neural computation and discuss the neural network architectures that can be applied in motion-based recognition.

Chapter 3

Neural Computation for Motion-based Recognition

We approach the problem of motion-based recognition by looking at the paradigm of neural computation. Under this approach, motion-based recognition requires the implementation of two distinctive processes:

1. The computation of observations from an image sequence.
2. The modelling of a sequence of observations in such a way as to construct motion models.

In this chapter we discuss the computation of two types of observation:

1. 2D feature extraction
2. 2D apparent motion

In the case of motion models, we address the issue of modelling a sequence of observations for the purpose of tracking moving objects and concentrate on the modelling of motion trajectories.

The paradigm of neural computation is realised with networks of highly interconnected simple processing elements (neurons) called *Neural Networks* (NNs) [47, 50]. Instead of performing a program of instructions as in a von Neumann architecture, NNs explore many competing hypotheses simultaneously using parallel but local computation at their processing elements. NNs are also known as "connectionist models" [10, 28] or "parallel distributed models"[123]. They denote a computational paradigm that is inspired by the functionality and the architecture of biological nervous systems but have very little physical resemblance to their biological counterparts. Throughout this thesis, the processing elements that comprise the NNs will also be referred as *elements* and the links between them as *weights.*

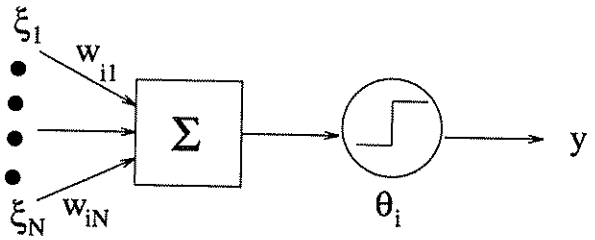


Figure 3.1: *A schematic diagram of a McCulloch-Pitts processing element.*

3.1 Characteristics of Neural Networks

Existing NN models are similar in the following respects:

- *Information processing occurs for many elements simultaneously by the application of an activation function to the weighted sum of their inputs:* A typical processing element is that of McCulloch and Pitts [89] shown in Figure 3.1. This is a binary threshold element, i, whose output y depends on the weighted

sum of its inputs $\xi_{1...N}$:

$$y_i = \Theta(\sum_{j=1}^{N} w_{i,j}\xi_j - \theta_i) \tag{3.1}$$

where N is the total number of inputs, $w_{i,j}$ is the weight between elements j and i, θ_i is a threshold value for element i and $\Theta(x)$ is a step function, acting as the activation function of the element, given by:

$$\Theta(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

In most NN models, the step function is replaced by a more general sigmoid nonlinear function $\varphi(x)$ that is differentiable. $\varphi(x)$ is usually given by the *logistic function*:

$$\varphi(x) = f_\beta(x) = \frac{1}{1 + \exp(-2\beta x)}, \qquad -\infty < x < \infty$$

where β is the steepness parameter. The logistic function is non-symmetric and bounded between $(0, 1)$ or the *hyperbolic tangent function*:

$$\varphi(x) = tanh(\beta x) = \frac{1 - \exp(-\beta x)}{1 + \exp(-\beta x)}, \qquad -\infty < x < \infty$$

which is asymmetric and bounded between $(-1, 1)$. In general, some networks learn faster when the sigmoidal activation function built into the processing element of the networks is asymmetric than when is non-symmetric [47].

- *Signals are passed between any two elements through their connections, which are associated with real values called weights:* Negative weights denote an inhibitory interaction between two elements, whereas positive weights denote an excitatory interaction.

- *Elements are usually grouped in layers:* The flow of the signals in NNs is either (i) *feed-forward* from the input (first) layer towards the output (last) layer of a network (Figure 3.2), or (ii) *recurrent* with all or some feedback signals in the network flow (Figures 3.3 and 3.4). The output value of an element of a

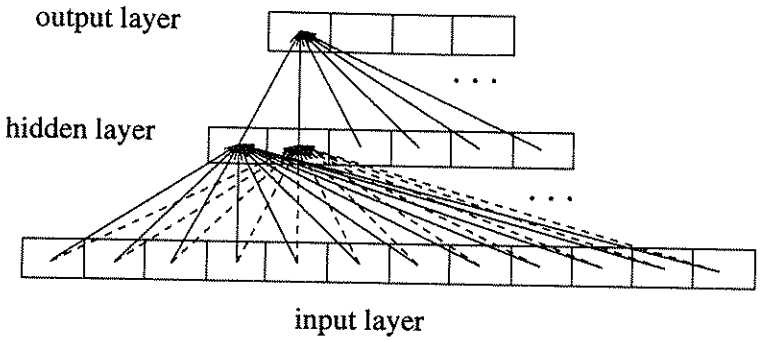


Figure 3.2: *An illustration of a multi-layer feed-forward network.*

recurrent network at time t is given by an update equation of the following forms:

$$y_i(t) = \varphi(\sum_{j=0}^{N} w_{i,j} y_j(t-1)) + \theta_i \tag{3.3}$$

$$y_i(t) = \varphi(\sum_{j=0}^{L} w_{i,j} y_j(t-1) + \sum_{j=0}^{N} w_{i,j} \xi_j(t)) \tag{3.4}$$

where L is the total number of output elements.

Networks with symmetric connections whose output values are updated using Equation (3.3) converge to a stable state. An example is the Hopfield model [54, 55] shown in Figure 3.3 which is typically used for modeling associative memories or solving constraint satisfaction problems. Networks with similar properties but different architectures have also been proposed [6, 106, 121].

The behaviour of most of the networks that learn how to recognise or reproduce temporal sequences is described by equations similar to that of (3.4). Examples of such networks can be found in [27, 65, 93, 104, 119, 127, 145, 146]. The main disadvantage of such networks is that they do not converge to a stable state [47, 50]. An example of a single layered recurrent neural network [27] is shown in Figure 3.4.

- *Information in NNs is stored in the weights and is distributed over the whole network:* Most of the networks acquire information through either *supervised* or *unsupervised* learning procedures. With a supervised procedure, learning is done on the basis of direct comparison of the output of the network with known correct outputs [122, 123]. In an unsupervised procedure, learning is not defined by specific input-output pairs. Instead, the only available information to be learnt are the correlations in the input data. A network is expected to extract meaningful information from these correlations [17, 49, 73, 82, 100].

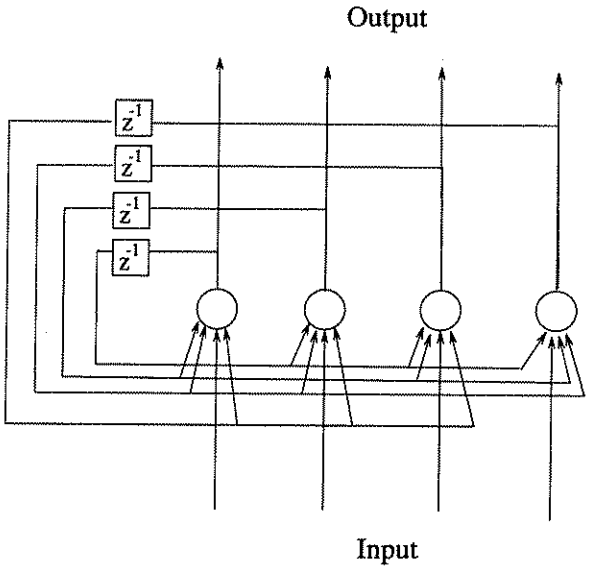


Figure 3.3: *The Hopfield model is a fully connected recurrent network.*

The main advantages of the NN models compared to that of von Neumann are:

- *The NN model is highly parallel:* Most importantly, once the weights between the elements of a network have been assigned, a learned task can be performed very fast.

- *The information is distributed in the weights of the network:* This gives a greater degree of robustness or fault tolerance [123].

- *Most networks collect their information through the adaptation of the weights between the processing elements:* With NNs, instead of having to specify every detail of a calculation, one simply needs to compile a training set of representative examples.

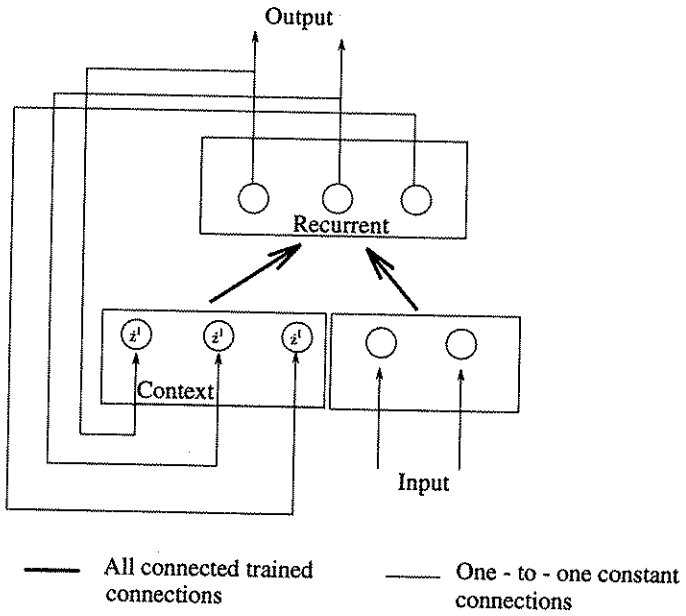


Figure 3.4: *A typical single layered recurrent network.*

In the remainder of this chapter, we describe in detail the architecture and functionality of NNs that can be used for motion-based recognition. As the scope of this thesis does not allow for the detailed description of the neural computation paradigm, interested readers should look at the work of McCulloch and Pitts [89] on the modelling of the first processing element; Hebb [48] on the physiological learning rule for synaptic modification between two neurons; Rosenblatt [120], Widrow and Hoff [143, 144] on the first NN models: Perceptron and Adaline; von Neumann [98, 99] on

the redundancy in the representation of information in NNs; Winograd and Cowan [147] on the use of distributed redundant representation on NNs; Minsky [90] on the introduction of the idea of feedback connections in neural networks for the representation of sequences and Minsky and Papert [91] on the computational limitations of Perceptrons. Other interesting work and reviews on NNs can be found in various articles [8, 20, 21, 46, 83] and the books by Hertz *et al* [50] and Haykin [47].

3.2 Feature Extraction

Automatic extraction of closed contours from grey-level images irrespective of their position, orientation, scale and deformation has long been one of the main concerns of researchers. Some of the most notable applications are the extraction of features such as eyes, nose or mouth from face images [79], recognition of characters [30, 81] and cells from biomedical images [113]. Effective extraction of closed-curve features requires three distinctive processes:

1. The construction of a parameterised feature model.
2. The delineation of the feature's boundary in the image and computation of its properties such as edge orientation, contour curvature, binding energy of the contour.
3. The matching of the information extracted from the image to that of the model.

In some cases the last two stages of computation are integrated into a single process, as for example in the representation of the shape of a feature on the *Generalised Hough Transform* (GHT) [9, 25, 61] or the *Active Contour Model* [14, 66, 67]. Both techniques are widely used for the modelling and detection of 2D shapes with irregular contours. With GHT, a parametric description of the contour is constructed based on

sample features. This information is then used to locate similar features irrespective of their position, scale or orientation. In active contour models, the contour of a feature is defined as an energy-minimising spline, also known as an "active snake" as it exhibits dynamic behaviour. However, both these methods have serious limitations:

1. GHT requires a complete specification of the exact shape of the contour.
2. A solution using the active contour models requires an interaction that must specify an approximate shape and a starting position for the snake somewhere near the desired contour. This *a priori* information is then used to push the snake towards an optimal match.
3. They are computationally very expensive. GHT requires a lot of storage and extensive computation whilst active contour models seek an optimisation of a set of linear equations by iteration.
4. The solutions suffer from numerical instability. Active contour models depend on the designation of numerous parameters and the initialisation process whereas GHT depends on the "peak" detection in a parameter space.

3.2.1 Cytological Image Interpretation

For various applications, automatic extraction of features from cytological images is particularly important but also challenging. It is important because the manual interpretation of cytological images is time consuming and requires the use of highly skilled cytotechnicians which is prone to human misinterpretation. There have been instances where cell characteristics that are not easily identified by the human eye have been discovered by automated systems [77]. It is challenging for two main reasons:

1. Microscopic and cinema-scopic techniques provide bad imaging quality.

2. The shape of the cells is dynamic as it changes according to the phase status of the cell and its interactions with the environment.

Existing cytological image interpretation systems that use image processing techniques for image segmentation and feature enhancement were developed for chromosome analysis and cervical smear screening [24, 45]. These systems involve a primitive image analysis concept, poor knowledge representation, and depend heavily on user interaction in order to function. An example of such a system is given by Graham where the image is segmented using thresholding based on global or local grey level values which depend on the task [45]. The aim of the system is to count chromosomes and when the enhanced image quality allows, to classify well formed and separated chromosomes. During the classification, problems encountered with "difficult" cells are overcome by recourse to operator interaction as in the case of composite chromosomes or displaced centromeres.

More recent specialised medical and biomedical applications use artificial intelligence techniques [103, 107], representation of *a priori* knowledge [150], and mathematical formulation [101]. In particular, Wu used *a priori* knowledge in a model based contour analysis technique in order to recognise and resolve composite chromosomes [150]. However, the models are not generic and do not include uncertainty measurements. Consequently, there has to be a model for each different chromosome configuration in order for the algorithm to cope with the cases that have not been explicitly specified. In another attempt, Oliver used a mathematical formulation in order to correlate the position of the centre of the cell with deformation parameters of the cell boundary [101]. However, it was recognised that the "snake" method used to mark the cell boundaries does not work in areas of poor resolution, minimal gradient, or subjective contour. Furthermore, the process of mapping the boundaries using an

energy model was proved to be computationally expensive.

3.2.2 Neural Networks for Feature Extraction

It is therefore not surprising that many researchers have found the application of NNs to the analysis of cytological images particularly attractive. In the work reported by Dytch and Wied, a neural network classifier for binary cells has achieved 20% improvement compared with the results from a statistical classifier [26]. In another example, Turner *et al* used a series of cascaded neural networks to extract features and classify chromosomes [132].

A popular network architecture for feature extraction is the *feed-forward multi-layer perceptron* such as the one shown in Figure 3.2. It consists of a set of input elements, one or more hidden elements and output elements. The input elements are linear, the hidden elements are non-linear and the output elements are either linear or non-linear depending on the application. In most applications that use neural networks for image extraction, the networks are trained by associating a set of images with a feature class. A common way to train such networks is to use the *error-back-propagation* learning algorithm [122]. This is an iterative algorithm whose objective is to adjust the weights of the network so as to minimise an overall cost measure $\mathcal{E}$, given by the sum of squared error between the expected and the computed outputs of the network. After each *epoch* [1] n, $\mathcal{E}$ is given by:

$$\mathcal{E} = \frac{1}{2}\sum_{\mu=1}^{P}\sum_{i\in\mathcal{C}}(\zeta_i^\mu - o_i^\mu(n))^2 \tag{3.5}$$

where P denotes the total number of example features, the set $\mathcal{C}$ includes all the elements in the output layer of the network, ζ_i^μ is the expected output of element i for pattern μ and $o_i^\mu(n)$ is the output computed by the network for pattern μ at epoch n.

[1] The presentation of all training examples to the network constitutes an epoch [47].

An efficient way to minimise $\mathcal{E}$ is to use an approximation of the steepest descent algorithm that is based on the instantaneous estimation of a cost function $\mathcal{E}(\mu)$, where μ is an example feature:

$$\mathcal{E}(\mu) = \frac{1}{2}\sum_{i\in\mathcal{C}}(\zeta_i^\mu - o_i^\mu)^2 \tag{3.6}$$

In this case the weights of the network are updated on a feature-by-feature basis and the adjustments are made in accordance with the respective errors computed for each example feature presented to the network using the *Delta rule*:

$$\Delta w_{ji}(\mu) = -\eta\frac{\partial\mathcal{E}(\mu)}{\partial w_{ji}(\mu)} \tag{3.7}$$

where η is the learning rate and $w_{ji}(\mu)$ is the weight between elements i and j. This incremental learning is called *on-line learning*. An alternative approach only updates the weights after all examples have been presented. This is known as *batch learning*. Although the effectiveness of these two different approaches depends on the problem, the on-line learning seems superior in most cases [47].

A momentum term is usually added to smooth the weight changes over time. This is given by the *Generalised Delta rule*:

$$\Delta w_{ji}(\mu) = \alpha\Delta w_{ji}(\mu - 1) - \eta\frac{\partial\mathcal{E}(\mu)}{\partial w_{ji}(\mu)} \tag{3.8}$$

where α is the momentum constant and should be restricted to $0 \leq |\alpha| < 1$. Details on the performance of this algorithm and the determination of the parameter values can be found in [47, 50].

An alternative way to classify features is to use unsupervised networks, where the network creates classes according to "common features" of the presented examples. Examples of such algorithms are Kohonen's self organising map [74], and Carpenter and Grossberg's ART [17].

The feed-forward layered architectures discussed so far do not encode any *a priori* knowledge about the geometrical or topological properties of the input data. An

alternative architecture has been introduced by Le Cun [80, 81] referred to here as *"locally connected"*, shown in Figure 3.5.

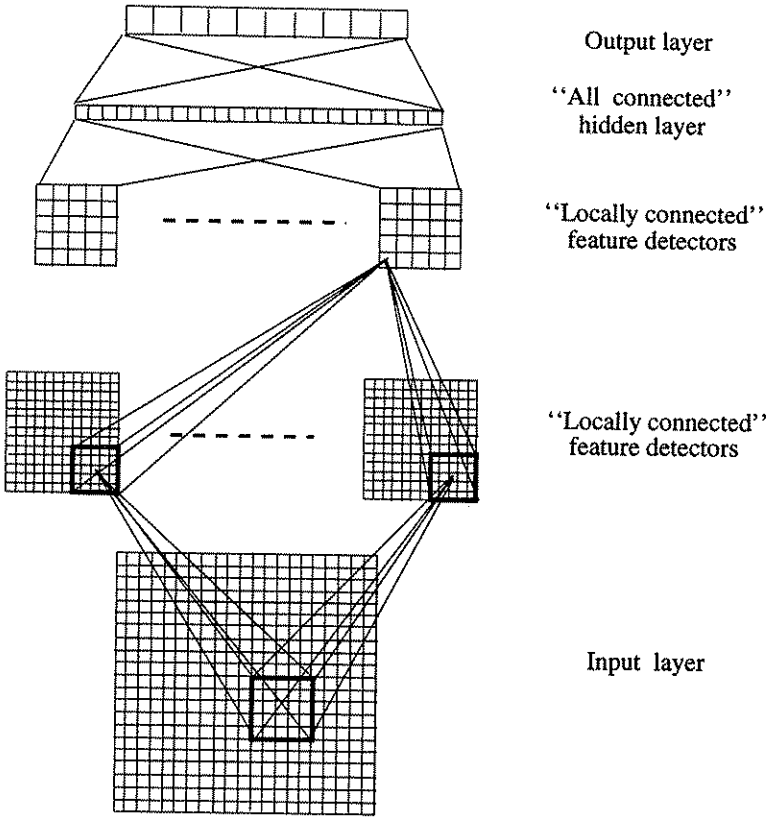


Figure 3.5: *A "locally-connected" network was designed to recognise handwritten ZIP codes from the U.S. mail [81]. From bottom to top we have the input layer, first layer of "locally connected" feature detectors, second layer of "locally connected" feature detectors, a layer of "all connected" elements and the output layer.*

Each layer in this network is divided into overlapping tiles and the elements of each tile are connected to one (or more) element(s) in the next layer. This approach is particularly applicable to image pattern classification since in the construction of the network architecture it uses knowledge about the spatial arrangements of the image intensities. Furthermore, this network decreases the degrees of freedom in the network, thus reducing the training time and improving its generalisation ability.

In Chapter 5 we illustrate the extraction of the centroids of cancer cells from cytological images using similar feed-forward architectures.

3.3 Computing Optic Flow

Optic flow is the spatio-temporal disparity in the retina that is largely caused by the relative motion between textured objects and the observer. Therefore, it is also known as two-dimensional retinal apparent motion and it is one of the main sources of information for motion-based recognition. The computation of optic flow is based on the detection of spatio-temporal changes in the intensity from a sequence of images [37]. In particular, it relies on the measurement of spatial and temporal image gradients to estimate the apparent speed and direction in which each image pixel moves. Based on the Constant Illumination Assumption, i.e. apparent motion is only caused by physical motion in 3D without any change in lighting conditions, Horn and Schunck [59] proposed a scheme that constrains the computation of optic flow as follows:

$$\nabla I \cdot \mathbf{u} + \frac{\partial I}{\partial t} = 0 \tag{3.9}$$

where I is the image intensity, ∇I is the spatial gradient of the intensity and $\mathbf{u}$ is the optic flow displacement vector. Equation (3.9) is known as the *motion constraint equation.*

Equation (3.9) alone is not sufficient to determine both components of vector $\mathbf{u}$. It only constrains their values to lie on a straight line in the motion space. This is known as the "aperture problem" which states that only the normal component of optic flow along the direction of the intensity gradient can be measured locally. The normal component of the optic flow is given by:

$$\vec{n} \cdot \mathbf{u} = -\frac{I_t}{|\nabla I|}$$

where $\vec{n}$ is the unit gradient vector and $I_t = \partial I/\partial t$.

In order to determine both components (u, v) of $\mathbf{u}$, the following additional constraints have been proposed in the literature:

- Local smoothness exists in flow field variation [59].
- Optic flow is constant over certain segments of an image [68].
- A multi-constraint method which relies on the motion constraint Equation (3.9) and the use of several other functions of intensity. Candidate functions include directional derivatives or various spatial operators like contrast, entropy, average and power content [1, 92].

In Horn and Schunck's algorithm, a full optic flow field is computed by using global optimisation techniques to minimise an error function based upon the motion constraint equation and the assumption of local smoothness in flow field over the entire image. As we will show in the remainder of this section such a formulation lends itself naturally to an implementation on a linear resistive network [71].

3.3.1 Horn and Schunck's Algorithm

The error function in Horn and Schunck's algorithm is given by:

$$E(u,v) = \iint (uI_x + vI_y + I_t)^2 + \lambda[(\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2]\ dxdy \quad (3.10)$$

where (u, v) is the optic flow vector and I_x, I_y and I_t are the partial derivatives of the intensity I with respect to x, y and t. The minimisation of the first term in (3.10) constrains the final solution to be as close as possible to the measured data, whereas the minimisation of the second term imposes the local smoothness constraint on the solution. The degree of smoothness is governed by the parameter λ.

The error function $E(u, v)$ can be approximated by a finite-difference scheme [58]. Using the values of the optic flow at grid point (i, j) and its neighbours, one can approximate the smoothness constraint as:

$$s_{i,j} = \frac{1}{4}[(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 + (v_{i+1,j} - v_{i,j})^2 + (v_{i,j+1} - v_{i,j})^2]$$

and the difference between the true and the estimated (u, v) by:

$$c_{i,j} = (I_x u_{i,j} + I_y u_{i,j} + I_t)^2$$

where I_x, I_y, I_t are estimates of the rates of change of intensity with respect to x, y and t at point (i, j). Then, the true velocity $(u_{i,j}, v_{i,j})$ can be derived by minimising

$$e = \sum_i \sum_j (c_{i,j} + \lambda s_{i,j}) \tag{3.11}$$

Minimising functional e (3.11) gives a set of two linear equations:

$$(I_x u_{i,j} + I_y v_{i,j} + I_t) I_x - \lambda(u_{i,j} - \bar{u}_{i,j}) = 0 \tag{3.12}$$

$$(I_x u_{i,j} + I_y v_{i,j} + I_t) I_y - \lambda(v_{i,j} - \bar{v}_{i,j}) = 0 \tag{3.13}$$

where $\bar{u}_{i,j}$ and $\bar{v}_{i,j}$ are local averages of $u_{i,j}$ and $v_{i,j}$. The set of values $(u_{i,j}, v_{i,j})$ that minimise e is given by the following two iterative equations:

$$u_{i,j}^{n+1} = \bar{u}_{i,j}^n - \frac{I_x \bar{u}_{i,j}^n + I_y \bar{v}_{i,j}^n + I_t}{\lambda + I_x^2 + I_y^2} I_x \tag{3.14}$$

$$v_{i,j}^{n+1} = \bar{v}_{i,j}^n - \frac{I_x \bar{u}_{i,j}^n + I_y \bar{v}_{i,j}^n + I_t}{\lambda + I_x^2 + I_y^2} I_y \tag{3.15}$$

3.3.2 Locating Motion Discontinuities in Optic Flow Field

The disadvantage of Horn and Schunck's algorithm is that it smoothes the field over the boundaries of objects that move differently, and therefore is unable to identify motion discontinuities and thus distinguish motion boundaries [59]. One solution to

this problem is to introduce additional terms in the error function, $E(u, v)$ that mark the object boundaries in images and therefore break the global smoothness in the optic flow field. These terms are modelled as binary valued vertical and horizontal line processes because of the "all-or-nothing" character of the discontinuities [31, 50]. Therefore, a motion discontinuity between two neighbouring pixels $(i, i+1)$ in the image is indicated by $l_i = 1$, whereas $l_i = 0$ indicates that there is no discontinuity present (Figure 3.6).

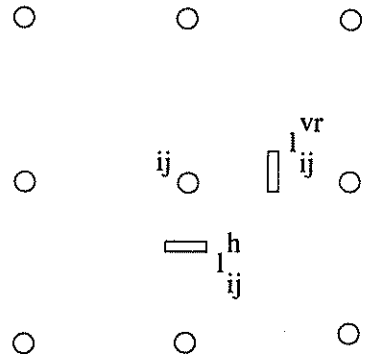


Figure 3.6: *The location of the horizontal* $(l^h_{i,j})$ *and vertical* $(l^{vr}_{i,j})$ *motion discontinuities relative to rectangular motion-field grid.*

An energy function using horizontal and vertical line processes to detect the object boundaries in the image while computing the optic flow is introduced by Koch *et al* [71]. Given that l^h, l^{vr} denote the horizontal and vertical line processes respectively, the energy function that needs to be minimised takes the following form:

$$
\begin{aligned}
E(u,v,l^h,l^{vr}) \;=\; & \sum_{i,j}(I_x u_{i,j} + I_y v_{i,j} + I_t)^2 \\
& + \lambda \sum_{i,j}(1 - l^h_{i,j})[(u_{i+1,j} - u_{i,j})^2 + (v_{i+1,j} - v_{i,j})^2] && (3.16)\\
& + \lambda \sum_{i,j}(1 - l^{vr}_{i,j})[(u_{i,j+1} - u_{i,j})^2 + (v_{i,j+1} - v_{i,j})^2] && (3.17)\\
& + c_c \sum_{i,j} l^h_{i,j} + c_c \sum_{i,j} l^{vr}_{i,j} && (3.18)\\
& + c_p \sum_{i,j} l^h_{i,j}(l^h_{i+1,j} + l^h_{i-1,j}) && (3.19)\\
& + c_p \sum_{i,j} l^{vr}_{i,j}(l^{vr}_{i,j+1} + l^{vr}_{i,j-1}) && (3.20)\\
& + c_i \sum_{i,j} l^h_{i,j}[(1 - l^h_{i,j+1} - l^{vr}_{i,j} - l^{vr}_{i+1,j})^2 \\
& \qquad + (1 - l^h_{i,j-1} - l^{vr}_{i,j-1} - l^{vr}_{i+1,j-1})^2] && (3.21)\\
& + c_i \sum_{i,j} l^{vr}_{i,j}[(1 - l^{vr}_{i+1,j} - l^h_{i,j} - l^h_{i,j+1})^2 \\
& \qquad + (1 - l^{vr}_{i-1,j} - l^h_{i-1,j} - l^h_{i-1,j+1})^2] && (3.22)
\end{aligned}
$$

where (3.16) and (3.17) determine whether there is a discontinuity in the optic flow field; (3.18) is a penalty for the introduction of line variables in the energy function; (3.19) and (3.20) are penalties for the formulation of parallel lines; (3.21) and (3.22) are penalties for line intersections. c_c, c_p, c_i are parameters that determine the relative contribution among various terms.

For the computation of the discontinuities two sets of parameters are important: First, the ratio between weights (c_p, c_c, c_i) which describe the interaction between the line processes and the weight λ, which determines the smoothness in the optic flow. Decreasing the importance of the line interaction terms versus the smoothness term encourages the formation of lines at smaller and smaller optic flow gradients. Second, the relative weight of the individual components of the line interaction terms.

From term (3.21), the cost of the various line intersections for a horizontal line process $l^h_{i,j}$ is illustrated by Figure 3.7.

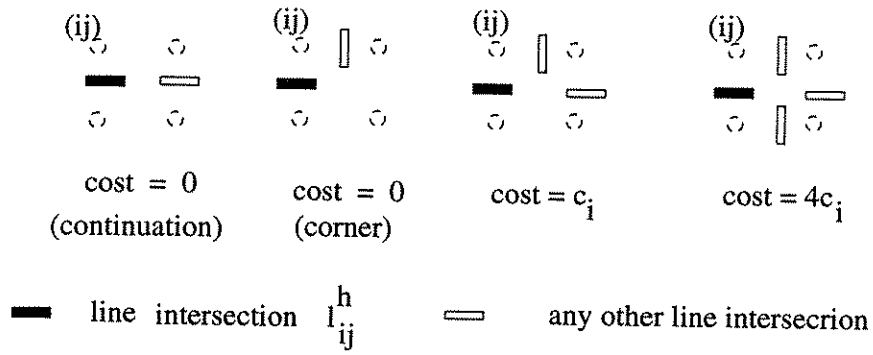


Figure 3.7: *Cost of the intersections of the line processes given the presence of a horizontal line process ($l^h = 1$).*

Given that $l^{vr}_{i,j} = 1$, the cost of the line intersections is similar.

The solution can be found by minimising $E(\mathbf{u}, l^h, l^{vr})$ for a given arrangement of the line processes (l^h, l^{vr}) by varying $\mathbf{u}$, since $\mathbf{u}, l^h, l^{vr}$ are independent variables.

3.3.3 Computing Optic Flow on a Resistive Neural Network

Hardware realisations of networks that exhibit behaviour similar to that of neural networks are very popular because:

- The computation in such networks is much faster than simulations on digital computers.
- They are more realistic implementations of the biological neuron system.

Koch *et al* [71] has shown that optic flow can be computed using a pair of simple linear resistive networks [71, 72]. Figure 3.8(a) shows a discrete two-dimensional grid,

on which Equations (3.12) and (3.13) can be modelled as:

$$I_x^2 u_{i,j} + I_x I_y v_{i,j} - \lambda(u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1}) + I_x I_t = 0 \quad (3.23)$$

$$I_y^2 v_{i,j} + I_x I_y u_{i,j} - \lambda(v_{i+1,j} + v_{i,j+1} - 4v_{i,j} + v_{i-1,j} + v_{i,j-1}) + I_y I_t = 0 \quad (3.24)$$

Figure 3.8 (b) shows a linear resistive network for computing one of the components of the optic flow. The combination of the capacitance C, the conductance $g_{i,j}^u$ and the battery $E_{i,j}$ can be thought of as a processing element, whose output is given by the voltage at node (i,j). In this case the voltage at each node (i,j) denotes one of the components of optic flow (u,v). Using Kirchoff's current law to the center node of the resistive network we have the following update equation:

$$C\frac{du_{i,j}}{dt} = T(u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1}) + g_{i,j}^u(E_{i,j} - u_{i,j}) \quad (3.25)$$

The optic flow is computed by using two such superimposed networks, where corresponding nodes are connected via a variable conductance $T_{c-i,j}$, as in Figure 3.9. We then have two equations similar to Equation (3.25) with a coupling term:

$$\begin{aligned} C\frac{du_{i,j}}{dt} &= T(u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1}) \\ &\quad + g_{i,j}^u(E_{i,j} - u_{i,j}) + T_{c-i,j}(v_{i,j} - u_{i,j}) \quad (3.26) \\ C\frac{dv_{i,j}}{dt} &= T(v_{i+1,j} + v_{i,j+1} - 4v_{i,j} + v_{i-1,j} + v_{i,j-1}) \\ &\quad + g_{i,j}^u(E_{i,j} - v_{i,j}) + T_{c-i,j}(u_{i,j} - v_{i,j}) \quad (3.27) \end{aligned}$$

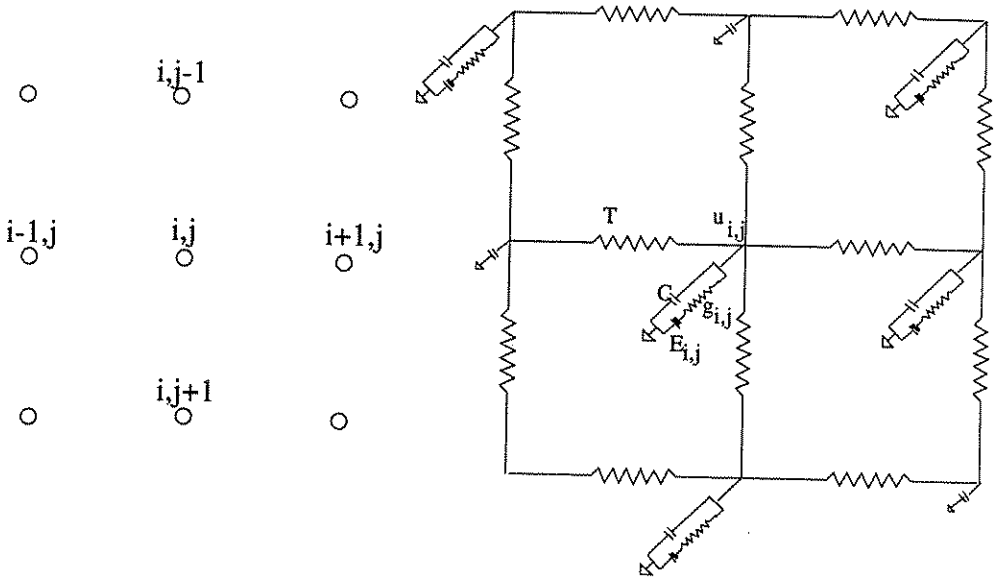


Figure 3.8: *Left: The sum of squares of the first partial derivatives of u and v at position (i, j) can be estimated using the differences of the optical flow components at neighbouring points. Right: Part of the resistive network minimising the discrete version of the error function given by Equation (3.10). The conductance T connecting neighbouring nodes is constant. Each node connects to a variable battery $E_{i,j}$ via a conductance $g^u_{i,j}$. The final network consists of two such resistive networks superimposed via a variable conductance $T_{c-i,j}$, as is shown in Figure 3.9. Once the batteries $E_{i,j}$ and conductances $g^u_{i,j}$ and $g^v_{i,j}$ have been set, the network will converge, following Kirchhoff's laws, to the state of least power dissipation that corresponds to the solution of the error Equation (3.10) [71].*

Both Equations (3.26) and (3.27) are identical to Equations (3.23) and (3.24) if we associate:

$$
\begin{aligned}
T &\rightarrow \lambda \\
T_{c-i,j} &\rightarrow -I_x I_y \\
g^u_{i,j} &\rightarrow I_x(I_x + I_y) \\
g^v_{i,j} &\rightarrow I_y(I_x + I_y) \\
E_{i,j} &\rightarrow -\frac{I_t}{(I_x + I_y)}
\end{aligned}
$$

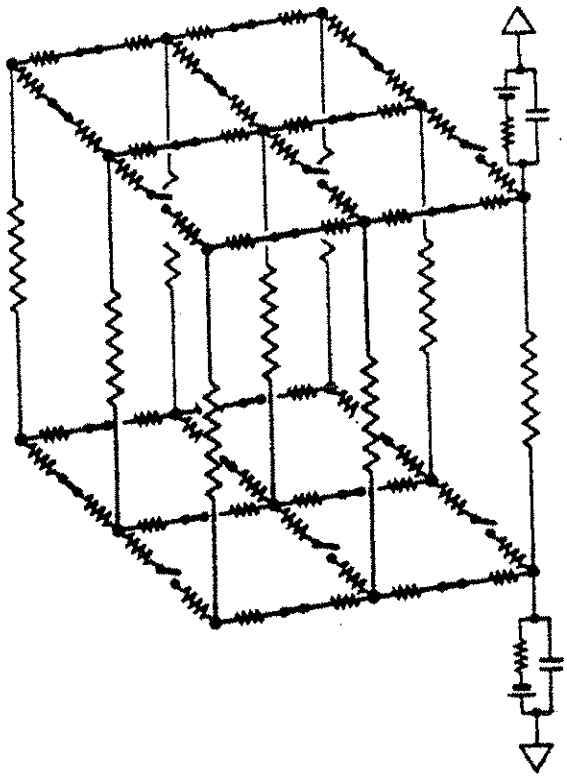


Figure 3.9: *The hybrid resistive network, computing the optical flow field in the presence of discontinuities [71].*

Once the batteries $E_{i,j}$ and conductances $g^u_{i,j}$ and $g^v_{i,j}$ have been set with appropriate values related to the precomputed intensity gradients I_x, I_y, I_t of the images, the network will converge, following Kirchoff's laws, to the state of the least power dissipation that corresponds to the solution of Horn and Schunck's error function given by Equation (3.10).

To find the motion discontinuities in a sequence of images, Koch *et al* used a purely deterministic algorithm based on solving Kirchoff's equations for a mixed analog and digital network [71, 72, 86]. The algorithm exploits the fact that for a fixed distribution of line processes, the energy function defined above is quadratic. Accordingly, the resistive network is first initialised with no line processes on, and the

network converges to the smoothest solution of the optic flow field. Subsequently, the line processes are updated by deciding at each site whether the overall energy can be lowered by setting or breaking the line process.

The algorithm always accepts the state of the line processes that correspond to the lowest energy configuration. For example l^h will be turned on if

$$E(u, v, l^h = 1, l^{vr}) < E(u, v, l^h = 0, l^{vr})$$

Otherwise $l^h = 0$. After the completion of one such analog-digital cycle, they reiterate and compute the smoothest state of the analog network for the newly updated distribution of line processes.

Although there is no guarantee that the system will converge to a global minimum, the system will find near-to-optimal solution in about 10 to 15 analog-digital cycles [71]. Furthermore, the algorithm must converge, because at each step the energy E is always reduced and E is bound from below.

3.4 Modelling Time-varying Information

As was described in Chapter 1, an activity or an object may be recognised directly from motion without structure recovery and this phenomenon was referred to as motion-based recognition. An ordered set of motion measurements from an image sequence constitutes a motion pattern which is associated with a particular movement of an object. Knowledge about the motion patterns allows us to construct motion models that can be used in the tracking and recognition of moving objects and their activities.

We also discussed in Chapter 2 that depending on the structure of an object and its movement, a motion model can be represented as:

1. The trajectory of a moving object [18, 39].
2. Temporal events that are seen as particular occurrences happening in the motion [35, 36, 44, 76], such as a change of direction or a stop at a particular time instance in the image sequence.
3. Optical flow measurement [87, 110].
4. Time-varying information on the photometric temporal change of a moving object. Such a representation may be based on the principal component analysis on the intensity values of the image [70] or on feature vectors [105, 151].

As motion constitutes information that is dynamic by nature and therefore time dependent, any representation of a motion model can be perceived as a temporal pattern. Assuming that a well defined relationship exists between the past and future values of the temporal pattern of a motion, then a motion model can be created through the presentation of a set of motion examples.

The most common solution to represent temporal information has been to give it a spatial representation. However, as we shall show in the next section, this representation has many disadvantages. A better approach would be to represent time implicitly, that is, to represent time by the effect it has on processing and not as an additional dimension of the input. Such a representation can be modelled by using temporal information from examples on Hidden Markov Models [39, 151]. However, this method requires the determination of the *a priori* probability distribution of the elements of the temporal patterns and that of their relationship. This can be difficult. An alternative approach is to use neural networks [47, 50]. Successful applications have been reported in the reproduction of grammars [34, 43] and prediction of financial

markets [94, 140]. In the remainder of this chapter we introduce some representations of temporal patterns on neural networks and present a network architecture that is suitable for motion-based recognition.

3.4.1 Representing Time in Feed-forward Neural Networks

A major limitation of feed-forward networks of the type trained using the back-propagation algorithm is that they are only suitable for learning input-output mappings that are static. This means that the input $\boldsymbol{\xi}$ and the output $\mathbf{O}$ represent patterns that are independent of time. The most straightforward way for a feed-forward network to perform recognition of a temporal sequence is to turn the sequence into a spatial pattern which is to be taken as the input of the network. In this case, the input $\boldsymbol{\xi}$ to a network is defined from the past samples $\xi(n-1), \xi(n-2), \ldots, \xi(n-p)$ as follows:

$$\boldsymbol{\xi} = [\xi(n-1), \xi(n-2), \ldots, \xi(n-p)]$$

The output responding to input $\xi(n-1)$ is an one-step ahead prediction $\hat{\xi}(n)$ whereas the actual input $\xi(n)$ represents the desired response.

In practice, temporal patterns could be fed into a delay line which is tapped at various intervals. The resulting architectures are sometimes called *time-delay neural networks*. An example is shown in Figure 3.10. The spatial representation of temporal patterns by parallelising time involves several drawbacks [27]:

1. It only allows a fixed window size on the temporal pattern representation.
2. It requires large amount of memory.
3. The network cannot easily distinguish the relative temporal position of an element in a sequence from its absolute temporal position.

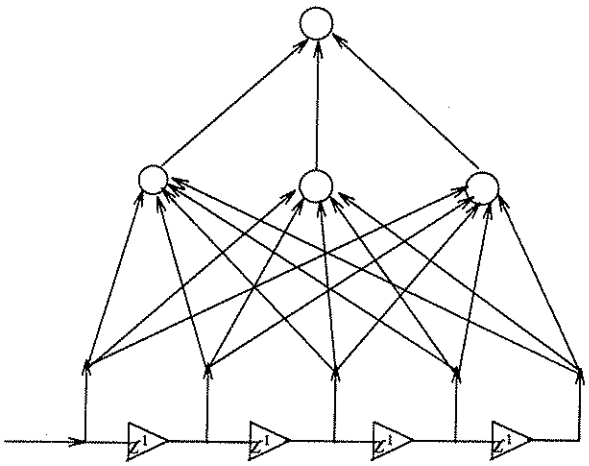


Figure 3.10: *A time-delay network.*

For a neural network to learn how to predict and classify temporal patterns produced by motion models, we need to add dynamic properties to networks that make them responsive to time-varying signals [47].

One way in which this can be achieved is to introduce time delays in the connections of a network [140] and to adjust their values during the learning phase. A modified McCulloch-Pitts element that includes temporal properties is shown in Figure 3.11. The temporal model of this element is used to construct a feed-forward network that can be trained with a gradient descent algorithm called *temporal back-propagation* [139] . This is a supervised learning algorithm in which a desired response is provided at each instance of time. The main drawback of this approach for motion-based recognition is that the network requires the input of several elements of the temporal pattern before it can start to predict the next element.

Another way in which a network can assume dynamic behaviour is to make it recurrent.

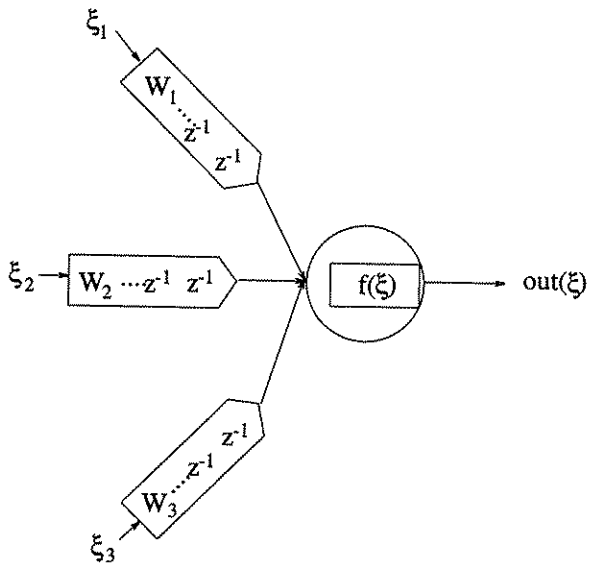


Figure 3.11: *A modified McCulloch-Pitts element to include temporal properties.*

3.4.2 Representing Time in Recurrent Neural Networks

It was stated in the introduction that objects move purposefully in an environment and effective prediction of their trajectories can be achieved by modelling the spatio-temporal regularities associated with such motions [16, 39, 114]. Temporal prediction and recognition require:

- A short-term memory that retains aspects of the input sequence relevant to prediction and recognition.
- The specification of a function that combines the current memory and the current input in order to form a new temporal context [94].
- The identification and learning of regularities from temporal sequence.

These tasks require a network architecture that:

- Accepts discrete time-varying input signals.
- Can form internal states with transition loops among them that can be traversed with the application of time-varying signals.

To accomplish this, we need to utilise a system that is capable of storing internal states and implementing complex dynamics. Neither a feed-forward network nor a network with symmetric connections will do, because they necessarily converge to a stationary state [50]. However, in a recurrent network with asymmetric connections, the state of the system can be encoded in the activity pattern of the elements and a wide variety of dynamical behaviours can be programmed by the weights.

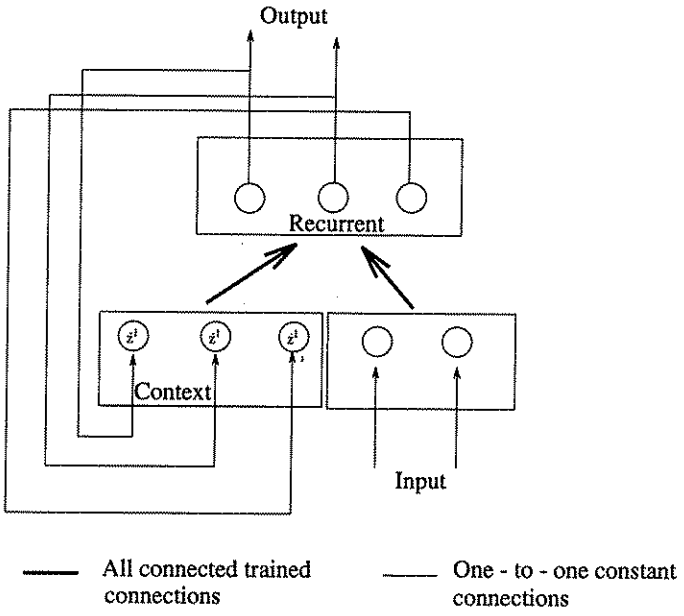


Figure 3.12: *A typical single layered recurrent network.*

An example of a Single Layered Recurrent Neural Network (SLRNN) was shown in Figure 3.12. It consists of a set of L recurrent elements, some serving as output

elements and the rest as hidden elements, and a set of N non-recurrent input elements which receive data from outside the network. Let $\boldsymbol{\xi}(t)$ denote the $N \times 1$ external input vector applied to the network at discrete time t, and let $\boldsymbol{s}(t+1)$ denote the corresponding $L \times 1$ output of the recurrent elements produced at $t+1$. The input vector $\boldsymbol{\xi}(t)$ and the one-step delayed vector $\boldsymbol{s}(t)$ are concatenated to form the $(N+L) \times 1$ vector $\boldsymbol{u}(t)$, whose ith element is denoted by $u_i(t)$. Let $\mathcal{A}$ denote the set of indices i for which $u_i(t)$ is an external input, and let $\mathcal{B}$ denote the set of indices i for which $u_i(t)$ is the output of a recurrent element. The dynamics of the system are then described by the following recurrence equations:

$$\begin{aligned} s_i(t+1) &= \varphi(h_i(t)) \\ h_i(t) &= \sum_{j}^{N+L} w_{i,j} u_j(t) + \theta_i \end{aligned}$$

where:

$$u_i(t) = \begin{cases} \xi_i(t) & \text{if } \in \mathcal{A} \\ s_i(t) & \text{if } \in \mathcal{B} \end{cases}$$

and $\boldsymbol{\theta}$ is a vector of biases, whilst φ is a nonlinear activation function.

In general an error function similar to that of the feed-forward networks is defined in recurrent networks and its gradient with respect to the weights is derived. The main differences from the feed-forward networks are that (1) input and output data are not static vectors, but temporal patterns and (2) a change in a weight can affect the future behaviour of the entire network. Two algorithms that do not involve the use of approximations in the computation of gradients which may be used to train a recurrent network are:

- *Back-propagation through time [123]*: The idea behind this approach is that for every recurrent network it is possible to construct a feed-forward network with identical behaviour over a particular time interval. For temporal patterns spanning time steps $t = 1, 2, \ldots, T$, we need to duplicate all T elements, so that

a separate element s_i^t holds the state $s_i(t)$ of the equivalent recurrent network at time t (see Figure 3.13). In this algorithm we define the following error function:

$$\mathcal{E}_{total}(n_0, n_1) = \frac{1}{2} \sum_{n=n_0}^{n_1} \sum_{j \in \mathcal{A}} e_j^2(n)$$

where n_0 and n_1 denote the start and end time of an epoch, $\mathcal{A}$ is the set of indices j pertaining to those elements in the network for which desired responses are specified and $e_j(n)$ is the error at the outputs of such elements measured with respect to the desired response.

The main limitation of this algorithm is the need for large computer resources as we need to duplicate all the elements of the network. For long temporal patterns and for temporal patterns of unknown length, the approach becomes impractical [50].

- *Real-time recurrent learning (RTRL) algorithm [145, 146]:* By using this learning algorithm, recurrent neural networks are trained without duplicating the units. If $\mathcal{C}$ denotes the set of output elements of a network and $d_j(t)$ denotes the desired response of output element j at time t, an error function $\mathcal{E}_{total}$ is given by summing $\mathcal{E}(t)$ over time t:

$$\begin{aligned} \mathcal{E}_{total} &= \sum_t \mathcal{E}(t) \text{ and} \\ \mathcal{E}(t) &= \frac{1}{2} \sum_{j \in \mathcal{C}} e_j^2(t) \end{aligned}$$

where:

$$e_j(t) = \begin{cases} d_j(t) - s_j(t) & \text{if } j \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$$

Similar to the error-back-propagation algorithm, described in section 3.2.2, $\mathcal{E}_{total}$ is minimised by computing the gradient $\nabla_{\mathbf{w}} \mathcal{E}_{total}$ as follows:

$$\begin{aligned} \nabla_{\mathbf{w}} \mathcal{E}_{total} &= \frac{\partial \mathcal{E}_{total}}{\partial \mathbf{w}} \\ &= \sum_t \nabla_{\mathbf{w}} \mathcal{E}_t \end{aligned}$$

where $\nabla_{\mathbf{w}}\mathcal{E}_t$ is the gradient of $\mathcal{E}(t)$ with respect to the weight matrix $\mathbf{w}$. The storage requirements of this algorithm are also very large and proportional to the length $\mathbf{T}$ of the temporal pattern. They can be reduced if we approximate the solution of the error function $\mathcal{E}_{total}$ by updating the weights of the network after each time step. Nevertheless, in the case of a fully interconnected network with a total of L recurrent elements and N external inputs, we need, at each time step, to store a total of $L(L^2 + NL)$ values [47].

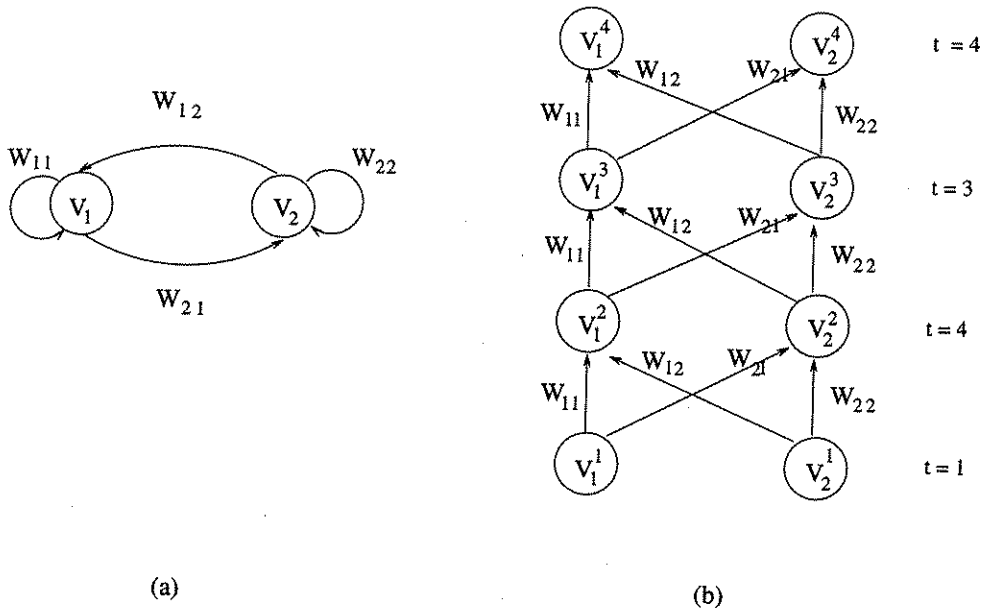


Figure 3.13: *(a) A fully recurrent network with two processing elements. (b) The network unfolded in time.*

- *Partially Recurrent Networks*: Some learning schemes of recurrent networks take into account the network dynamics for only one step back in time. Examples are the partially recurrent networks of Elman [27], Jordan [65] and others [94, 127]. These architectures operate only in discrete-time and the weight of the feedback connections is fixed. A set of context elements holds a copy of the activations of the hidden or the output units. The trainable weights are all on feed-forward

connections, and can therefore, be trained by the conventional back-propagation method. In the case of temporal patterns this is done at each time step. Two examples are shown in Figure 3.14.

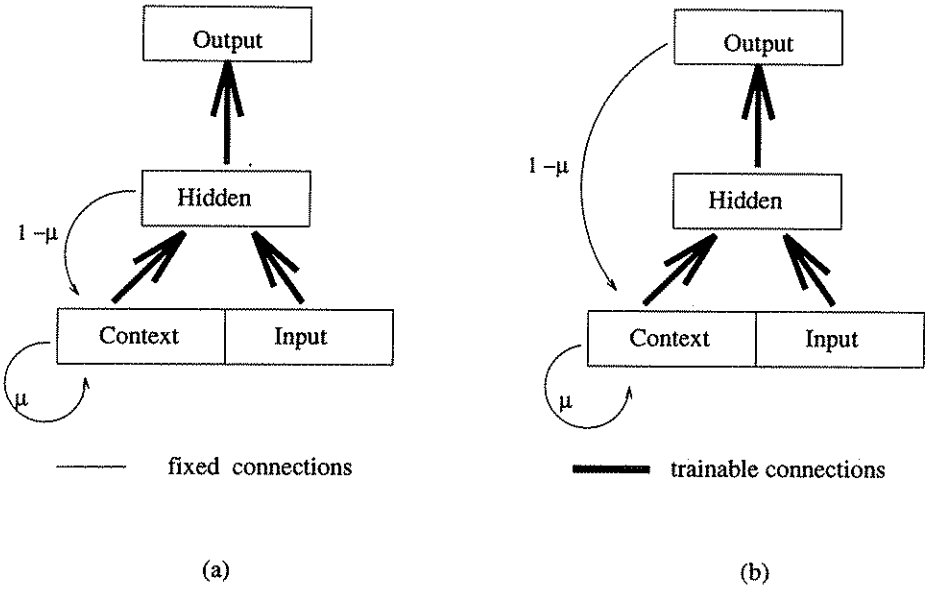


Figure 3.14: *Examples of partially recurrent networks: (a) Elman network (b) Jordan network.*

In these architectures, the network consists of N input elements, L context elements, K output elements and S hidden elements. The context elements remember some aspects of the recent past, and so the state of the whole network at time t depends on the aggregate of previous states as well as on the current state. The pattern of activation of the hidden elements represent an "encoding" of the features of the input patterns that are relevant to the task and therefore, hidden elements can now encode information about the relevant features of successive input data. Thus, in Elman networks the pattern of activation of the context elements represent an "encoding" of the relevant features of the past input elements [27]. Furthermore, the short term memory of the network can be augmented by convolving the context elements with an appropriate function

[94]. The main purpose for this is to give the context elements individual memory or inertia. An example is when the values of the context elements become an exponential encoding of the activation values of the hidden elements. The context elements maintain moving averages of past hidden value activations according to the equation:

$$y(t) = (1 - \mu_i)x(t) + \mu_i y(t-1) \tag{3.28}$$

where μ_i lies in the interval $[-1, 1]$ and represents averages spanning various intervals of time, $x(t)$ represents the vector of the hidden elements activation values at time t and, $y(t)$ represents the context vector i at time t. This is equivalent of convolving the context elements with the function $c_i(t) = (1 - \mu_i)\mu_i^t$.

The main advantages of recurrent networks over feed-forward networks are:

- The temporal pattern can now be processed sequentially without the need for a buffer.
- There is no requirement for absolute temporal position of an element.
- The transition between the states of a network can be described by a finite state machine [19, 43]. For a quintuple $(I, O, S, \delta, \lambda)$ where I is a set of inputs, O is a set of outputs and S is a set of states, the evolution of the network can be described by the state transition function δ and the output function λ

$$\delta : I \times S \rightarrow S \quad \text{is the state transition function}$$
$$\lambda : I \times S \rightarrow O \quad \text{is the output function}$$

In addition, the advantages of partially recurrent networks for motion-based recognition are:

- One step prediction can be used to track movement of objects.
- Evaluation of the differences between the predicted and computed values can be used to recognise movement of objects.

In Chapter 6 we show how partially recurrent neural networks based on Elman's architecture can be used to construct motion models for motion-based recognition.

3.5 Summary

As shown in the previous chapter, neural networks are suitable for addressing the problems encountered in motion-based recognition.

In this chapter we looked at the characteristics of neural networks and identified the network architectures that can be used for the computation of observations and the construction of motion models. In particular we showed (1) feed-forward neural networks that can be used for feature extraction, (2) resistive neural networks that can be used for the computation of optic flow and (3) fully and partially recurrent neural networks that can be used in modelling time-varying information.

In the following chapters of this thesis we present our work in the parallel implementation of neural networks, and their application in (1) the computation of the centroid of cancer cells, (2) the hybrid computation of optic flow and (3) the construction of motion models from motion trajectories.

Chapter 4

Hybrid Parallel Computation of Optic Flow Field

The measurement of optic flow from a sequence of images is an expensive process that requires an efficient parallel implementation, if a real-time computation is needed [37]. Furthermore, the detection of motion discontinuities near object boundaries requires the introduction of additional constraints that further add to the computational cost. As shown in Chapter 3, one way of introducing such constraints is the use of line processes in the energy function that describes the optic flow vector. Koch *et al* [71] mapped Horn and Schunck's algorithm [59] onto a linear resistive network (see Figure 3.9) and computed the motion discontinuities by using a deterministic algorithm on a mixed analog and digital network. In this case, the line processes correspond to a regular grid of serial programmable processors that communicate locally with the linear resistive network. The digital processors can break the resistive connection between two neighbouring analog processors thus denoting a motion discontinuity. Finally, to accurately compute the locations of the motion discontinuities they were required to coincide with the locations of intensity changes.

In this chapter we present a software implementation of Horn and Schunck's algorithm on an SIMD parallel machine (the AMT DAP) and show the similarity of Horn and Schunck error function with that of the Liapunov function that describes

the behaviour of the Hopfield model. Furthermore, we exploit the properties of the DAP and the Hopfield model to deliver an efficient parallel computation of motion discontinuities using binary and continuous valued line processes.

4.1 The Hopfield Model

Hopfield has shown that the dynamic behaviour of a discrete recurrent network with symmetric weights ($w_{i,j} = w_{j,i}$), no self-connections ($w_{i,i} = 0$), and processing elements with step activation functions can be described by a *Liapunov function*

$$H = -\frac{1}{2}\sum_{i,j} w_{i,j} y_i y_j - \sum_i I_i y_i + \sum_i \theta_i y_i$$

where y_i, y_j are the outputs of the elements i, j respectively, I_i is the bias and θ_i is the threshold value of a processing element i [54, 55].

The output values of the elements change iteratively by using the following rule:

$$\begin{aligned} y_i &\rightarrow 0 \text{ if } \textstyle\sum_{j \neq i} w_{i,j} y_j + I_i < \theta_i \\ &\rightarrow 1 \text{ if } \textstyle\sum_{j \neq i} w_{i,j} y_j + I_i > \theta_i \end{aligned}$$

This would reduce the value of H until it reaches a stable state. The discrete Hopfield model can solve constraint satisfaction problems by appropriately representing the constraints through certain combination of the initial network state, externally supplied inputs and both excitatory and inhibitory connections between the elements.

A similar architecture has processing elements with continuous outputs which are monotonically increasing functions of the inputs and are limited to $0 \leq y_i \leq 1$ [55]. This is known as the continuous Hopfield model [56, 57]. The update rules for the elements are given by the following set of equations:

$$\tau_i\left(\frac{dy_i}{dt}\right) = -y_i + \varphi\left(\sum_j w_{i,j} y_j\right) \tag{4.1}$$

where τ_i is a suitable time constant and $\varphi(x)$ is a non-linear function.

In an equilibrium, $y_i(t)$ ceases to change and $dy_i/dt = 0$ for all i. In this case the output of all units in the network is given by $y_i = \varphi(\sum_j w_{i,j} y_j)$.

The energy function that describes the behaviour of the continuous Hopfield model is given by:

$$H = -\frac{1}{2}\sum_{i,j} w_{i,j} y_i y_j + \frac{1}{\lambda}\sum_i \int_0^{y_i} g^{-1}(y)dy \tag{4.2}$$

where λ is a finite positive number, g^{-1} is a monotone increasing function and y_i, y_j are the output of elements i, j respectively.

Networks with this basic organisation can be used to compute a solution to an optimisation problem if it can be formulated as an energy function that corresponds to the Liapunov function of the Hopfield model.

The behaviour of these networks can either be simulated by solving the set of equations given in (4.1) or realised by analog computational networks [56, 57]. In the latter case the processing elements are modelled as amplifiers that have a sigmoid input/output curve and the connections between the elements are modelled using resistors. The time constant τ is modelled using a capacity C. This analog implementation is similar to that of the linear resistive network of Koch *et al* [71].

4.2 Horn and Schunck's Algorithm on a Hopfield Model with Linear Elements

Based on Equation (3.11) the discrete form of the error function $E(u, v)$ for measuring optic flow is given by:

$$\begin{aligned} e(u,v) &= \sum_{i,j}(I_x u_{i,j} + I_y v_{i,j} + I_t)^2 \\ &\quad + \frac{\lambda}{4}\sum_{i,j}((u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 \\ &\quad + (v_{i+1,j} - v_{i,j})^2 + (v_{i,j+1} - v_{i,j})^2) \end{aligned} \tag{4.3}$$

Using the gradient descent rule we can determine the values of (u, v) that minimise Equation (4.3) by:

$$\tau\frac{du_{i,j}}{dt} = -\frac{\partial e}{\partial u_{i,j}} \tag{4.4}$$

$$\tau\frac{dv_{i,j}}{dt} = -\frac{\partial e}{\partial v_{i,j}} \tag{4.5}$$

From Equations (4.4), (4.5) and (4.3) we have:

$$\begin{aligned} \frac{du_{i,j}}{dt} &= -[\frac{\lambda}{2}(4u_{i,j} - u_{i+1,j} - u_{i,j+1} - u_{i-1,j} - u_{i,j-1}) \\ &\quad + 2(I_x^2 u_{i,j} + I_x I_t + I_x I_y v_{i,j})] \end{aligned} \tag{4.6}$$

$$\begin{aligned} \frac{dv_{i,j}}{dt} &= -[\frac{\lambda}{2}(4v_{i,j} - v_{i+1,j} - v_{i,j+1} - v_{i-1,j} - v_{i,j-1}) \\ &\quad + 2(I_y^2 v_{i,j} + I_y I_t + I_x I_y u_{i,j})] \end{aligned} \tag{4.7}$$

Equations (4.6) and (4.7) are similar to Equation (4.1) but with *linear* elements $\varphi(u) = u$. From the first terms in Equations (4.6) and (4.7) we see that:

$$w_{(i,j),(i+1,j)} = w_{(i,j),(i-1,j)} = \frac{\lambda}{2}$$

$$w_{(i,j),(i,j+1)} = w_{(i,j),(i,j-1)} = \frac{\lambda}{2}$$

In addition, $I_x I_t + I_x I_y v_{i,j}$ and $I_y I_t + I_x I_y u_{i,j}$ are extra constant terms. Equations (4.6) and (4.7) are similar to Equations (3.26) and (3.27) which define the optic flow in the resistive network implementation with their solution given by Equations (3.14) and (3.15). Therefore, Equation (4.3) corresponds to the Liapunov function of a

continuous Hopfield model and the values $(u_{i,j}, v_{i,j})$ that satisfy $du_{i,j}/dt = 0$ and $dv_{i,j}/dt = 0$ would minimize Equation (4.3). Figure 4.1 shows an example of a synthetic sequence that has been used in the parallel computation of optic flow using Equations (4.6) and (4.7).

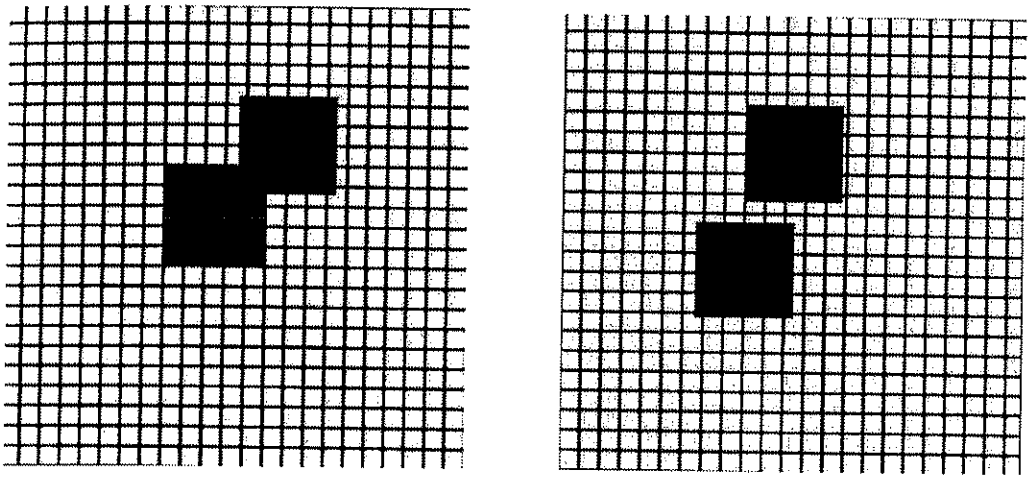


Figure 4.1: *A synthetic sequence used for the computation of optic flow. Two overlapping squares that are moving away from each other.*

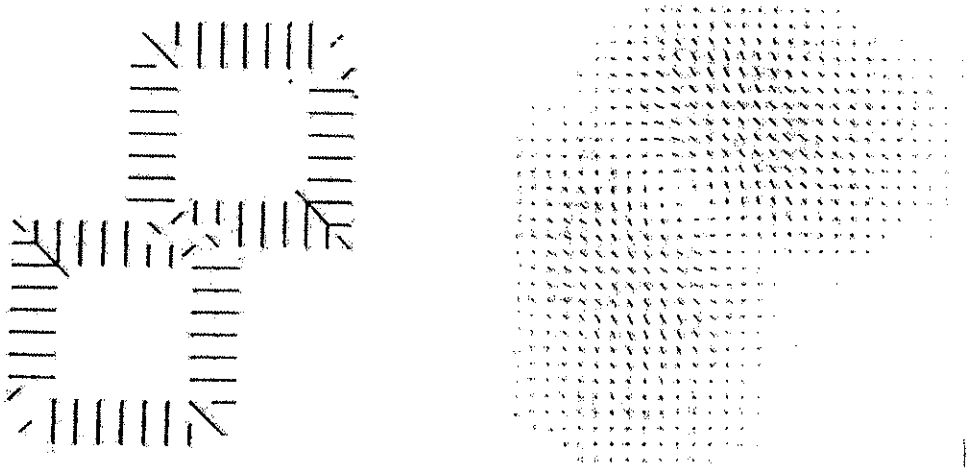


Figure 4.2: *The computed optic flow from the two overlapping squares after 1 iteration (left) and 240 iterations (right).*

Figure 4.2 shows the computed flow field using Equations (4.6) and (4.7) after 1 and 240 iterations respectively. The result is similar to that of the computation of the optic flow using Equations (3.14) and (3.15), but the equations converge to a higher energy minima.

4.3 Continuous Line Variables for Motion Discontinuities on a Hopfield Model

Koch *et al* [72] defined an energy function that included continuous line variables, to interpolate depth values in the image. The method is based on the work of Hopfield [55, 56] for solving combinatorial optimization problems by allowing the binary variable to vary continuously between 0 and 1 and to introduce terms in the energy function that force the final solution to one of the corners of the hypercube [0,1] . Following [55], the binary line processes l^h and l^{vr} are mapped into continuous variables bound by 0 and 1. The associated energy function is defined as:

$$
\begin{aligned}
E(u,v,l^h,l^{vr}) \;=\; & \sum_{i,j}(I_x u_{i,j} + I_y v_{i,j} + I_t)^2 \\
& + \frac{\lambda}{4}\sum_{i,j}(1-l^h_{i,j})[(u_{i+1,j}-u_{i,j})^2 + (v_{i+1,j}-v_{i,j})^2] \\
& + \frac{\lambda}{4}\sum_{i,j}(1-l^{vr}_{i,j})[(u_{i,j+1}-u_{i,j})^2 + (v_{i,j+1}-v_{i,j})^2] \\
& + c_c\sum_{i,j} l^h_{i,j} + c_c\sum_{i,j} l^{vr}_{i,j} \\
& + c_p\sum_{i,j} l^h_{i,j}(l^h_{i+1,j} + l^h_{i-1,j}) \\
& + c_p\sum_{i,j} l^{vr}_{i,j}(l^{vr}_{i,j+1} + l^{vr}_{i,j-1}) \\
& + c_v\sum_{i,j} l^h_{i,j}(1-l^h_{i,j}) \qquad (4.8) \\
& + c_v\sum_{i,j} l^{vr}_{i,j}(1-l^{vr}_{i,j}) \qquad (4.9) \\
& + c_i\sum_{i,j} l^h_{i,j}[(1-l^h_{i,j+1}-l^{vr}_{i,j}-l^{vr}_{i+1,j})^2 \\
& \qquad + (1-l^h_{i,j-1}-l^{vr}_{i,j-1}-l^{vr}_{i+1,j-1})^2] \\
& + c_i\sum_{i,j} l^{vr}_{i,j}[(1-l^{vr}_{i+1,j}-l^h_{i,j}-l^h_{i,j+1})^2 \\
& \qquad + (1-l^{vr}_{i-1,j}-l^h_{i-1,j}-l^h_{i-1,j+1})^2] \\
& + c_g\sum_{i,j}\int_0^{l^h_{i,j}} g^{-1}_{i,j}(l^h_{i,j})\,dl^h_{i,j} \qquad (4.10) \\
& + c_g\sum_{i,j}\int_0^{l^{vr}_{i,j}} g^{-1}_{i,j}(l^{vr}_{i,j})\,dl^{vr}_{i,j} \qquad (4.11)
\end{aligned}
$$

where the terms (4.8) and (4.9) force the line variables to the corners of the hypercube [0,1]. We introduce the terms (4.10) and (4.11) to define the activation function of the processing elements of the Hopfield model. The values of (u, v) that minimise $E(u, v, l^h, l^{vr})$ are given by the following update rules:

$$\begin{aligned}\frac{\partial E}{\partial u_{i,j}} &= 2(I_x^2 u_{i,j} + I_x I_t + I_x I_y v_{i,j}) \\ &\quad + \frac{\lambda}{2}(1 - l^h_{i,j})(2u_{i,j} - u_{i+1,j} - u_{i-1,j}) \\ &\quad + \frac{\lambda}{2}(1 - l^{vr}_{i,j})(2u_{i,j} - u_{i,j+1} - u_{i,j-1}) \qquad (4.12)\\ \frac{\partial E}{\partial v_{i,j}} &= 2(I_y^2 v_{i,j} + I_y I_t + I_x I_y u_{i,j}) \\ &\quad + \frac{\lambda}{2}(1 - l^h_{i,j})(2v_{i,j} - v_{i+1,j} - v_{i-1,j}) \\ &\quad + \frac{\lambda}{2}(1 - l^{vr}_{i,j})(2v_{i,j} - v_{i,j+1} - v_{i,j-1}) \qquad (4.13)\end{aligned}$$

To compute the value of the line variables we follow Koch *et al* [72] by using the update equations:

$$\frac{dm_{i,j}}{dt} = -\frac{\partial E}{\partial l^h_{i,j}} \text{ and } \frac{dn_{i,j}}{dt} = -\frac{\partial E}{\partial l^{vr}_{i,j}}$$

where $m_{i,j}$ and $n_{i,j}$ are the internal state variables for the processing elements that correspond to the horizontal and vertical line processes respectively. Furthermore, $l^h_{i,j} = g(m_{i,j})$ and $l^{vr}_{i,j} = g(n_{i,j})$. Therefore, $\frac{\partial E}{\partial l^h_{i,j}}$ is given by:

$$\begin{aligned}\frac{\partial E}{\partial l^h_{i,j}} &= -\frac{\lambda}{4}[(u_{i+1,j} - u_{i,j})^2 + (v_{i+1,j} - v_{i,j})^2] \\ &\quad + c_c \\ &\quad + 2c_p(l^h_{i+1,j} + l^h_{i-1,j}) \\ &\quad + c_v(1 - 2l^h_{i,j}) \\ &\quad + c_i[(1 - l^h_{i,j+1} - l^{vr}_{i,j} - l^{vr}_{i+1,j})^2 + (1 - l^h_{i,j-1} - l^{vr}_{i,j-1} - l^{vr}_{i+1,j-1})^2] \\ &\quad + 2c_i l^h_{i,j-1}(l^h_{i,j} - 1 - l^{vr}_{i,j} - l^{vr}_{i+1,j}) \\ &\quad + 2c_i l^h_{i,j+1}(l^h_{i,j} - 1 - l^{vr}_{i,j-1} - l^{vr}_{i+1,j-1}) \\ &\quad + 2c_i l^{vr}_{i,j}(l^h_{i,j} - 1 - l^{vr}_{i+1,j} - l^h_{i,j+1}) \\ &\quad + c_g g^{-1}_{i,j}(l^h_{i,j}) \qquad (4.14)\end{aligned}$$

Similarly for $\partial E / \partial l^{vr}_{i,j}$.

The solution of the equation $E(u, v, l^h, l^{vr})$ should approximate as closely as possible the state of lowest energy. Since u, v, l^{vr}, l^h are independent variables, the solution can be found by minimising $E(u, v, l^{vr}, l^h)$ for a given arrangement of the line processes by varying u, v. In a similar manner as above, the smoothest velocity field is computed while the line variables are set to zero. Then the line variables are updated by keeping the velocity values constant. Figure 4.3 shows a synthetic sequence that has been used for measuring optic flow and the detection of discontinuities based on minimising $E(u, v, l^h, l^{vr})$. It shows a square moving diagonally downwards and Figure 4.4 shows the computed optic flow and detected motion discontinuities.

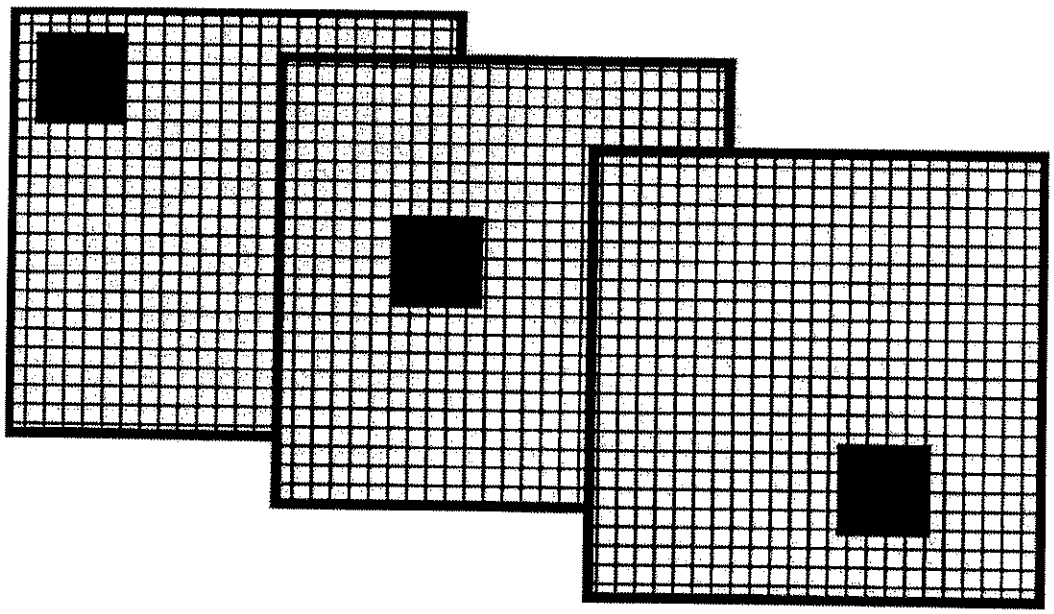


Figure 4.3: *One square moving diagonally downwards.*

From Equation (4.14) we can see that the update of the line variables must be asynchronous. Since each line variable depends only on four of its neighbours, the update of the network can be performed in two cycles, each cycle updates half of the elements in the network. The method does not guarantee that the network will

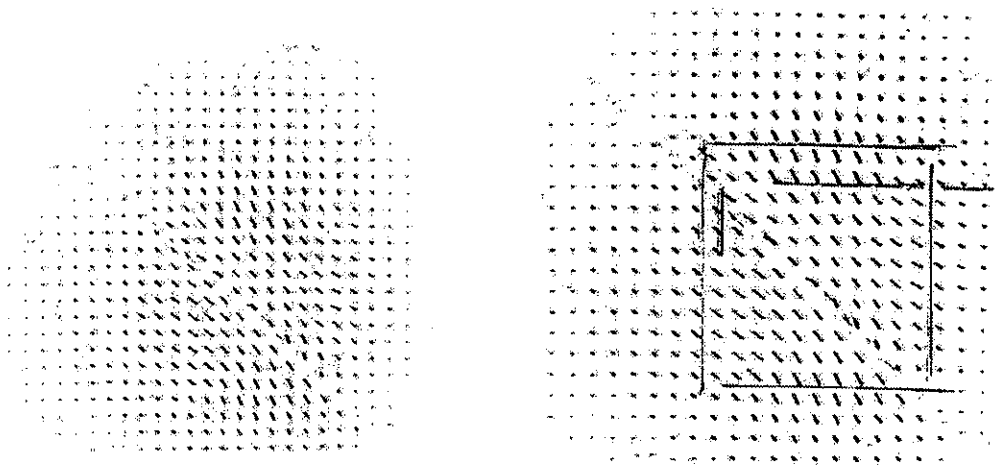


Figure 4.4: *Computed optic flow (left) from a square moving diagonally downwards and the detected motion discontinuities (right).*

converge to the state of lowest energy. Therefore, the fact that the relative costs between the various intersections are defined closely may affect the convergence of the network to the state of the lowest energy. Furthermore, a disadvantage of the multiple cost terms is the symmetric or relative cost that is assigned to the line processes. This makes the search for suitable constants harder.

4.4 Parallel Implementations on DAP

The AMT DAP (Distributed Array of Processors) used for the parallel computation of optic flow and the continuous Hopfield models is an SIMD machine consisting of 1024 processors connected as a 32×32 2D array. A sketch of the DAP architecture is given in Figure 4.5. Each processor has its own 16K of RAM and is connected to each four nearest neighbours.

The DAP is programmed using the parallel Fortran*, known as DAP-Fortran that supports the parallel processing of both one dimensional and two dimensional

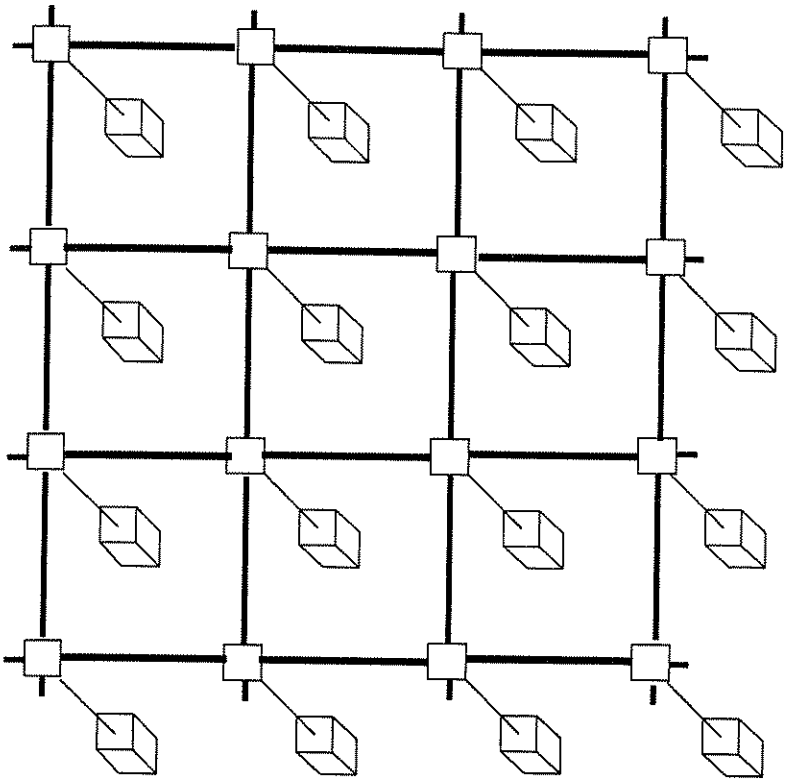


Figure 4.5: *A sketch of the DAP architecture. Each processor has interconnections to its four nearest neighbours and has access to its own 16k RAM [97].*

structures, called "vectors" and "matrices" respectively, thus making it ideal for the implementation of early vision algorithms and artificial neural networks. Also, among the benefits of DAP-Fortran is the inclusion of vectors and matrices of unconstrained size [1] and the introduction of new functions for indexing vectors and matrices that assist in the assignment and reading of new structures. Consequently, from the programming point of view it would appear that the layers of a neural network and their inter-connections can be mapped into the vectors and matrices supported on a DAP and their weights can be processed in parallel regardless of their size. A number of neural network applications have been implemented on the DAP. Forest has used the DAP for the restoration of binary images [29]. Psarrou and Buxton have used

[1] In earlier versions of DAP-Fortran, vectors and matrices had to be multiples of 32 elements.

the DAP extensively to implement Hopfield models [112], feed-forward and recurrent neural networks [113].

In the next sections we discuss in detail our experiments on the parallel computation of optic flow and the detection of motion discontinuities. In particular, in section 4.4.1 we show how Horn and Schunck's algorithm can be mapped onto a DAP and in section 4.4.2 we show the mapping of Koch's resistive network onto a DAP. In section 4.4.3 we discuss the computation of binary line variables on a DAP using the chessboard approach. This approach is later used in section 4.4.4 to compute the line variables modelled on a continuous Hopfield model. The computations on sections 4.4.2 and 4.4.4 consist the two components of the hybrid parallel implementation of optic flow.

4.4.1 Implementation of Horn and Schunck's Algorithm

Optic flow is extracted from different positions that objects occupy in an image between time t and t + dt. Accordingly, the input in Horn and Schunck's algorithm, takes two images that differ by one time step. In our implementation, 32×32 images, are mapped directly onto a DAP architecture.

The algorithm consists of computing the spatial and temporal gradients in the two images and iteratively updating the velocity components (u, v) of the optic flow field:

```
CALL DERVS(Image_1, Image_2, Ix, Iy, It)
CALL ITER(U_new, V_new, Ix, Iy, It, U_init, V_init)
DO 10 I = 1, COUNT
  CALL  MEANVEL(U_new, U_mean)
  CALL  MEANVEL(V_new, V_mean)
  CALL  ITER(U_new, V_new, Ix, Iy, It, U_mean, V_mean)
10  CONTINUE
```

where subroutine DERVS computes the intensity changes in the image; subroutine

ITER computes the velocity components (u, v) of the optic flow field; subroutine MEANVEL gives the average velocity estimation at each point in the image taking into account the four nearest neighbours.

Since only local information is involved in these computations, they are implemented in parallel using the special programming features provided by DAP-Fortran. Each velocity component (u, v) is mapped to a 32×32 DAP matrix. Initially the velocities are set to zero, and then updated iteratively until they converge to a stable state.

4.4.2 Koch's Resistive Network

The type of networks proposed by Koch *et al* [71, 72, 108] are usually referred to as neural circuits. The main advantages of these networks are their fast computation, and their ability to simulate human neuronal functionalities. Figure 4.6 shows an implementation of Koch's linear resistive network on the DAP based on Equations (3.26) and (3.27). The top layer computes the u component of the optic flow, whereas the bottom layer computes the v component. The coupled term $T_{c-i,j}$ of Equations (3.26) and (3.27) is implemented by a one-to-one connection between the processing elements $u_{i,j}$ and $v_{i,j}$.

4.4.3 Computing the Boolean Line Variables

As discussed in Chapter 3, Koch *et al* augmented the energy function for the computation of optical flow field to include terms for the detection of object boundaries. These additional terms consist of boolean line variables, that are set using a deterministic model described by the algorithm:

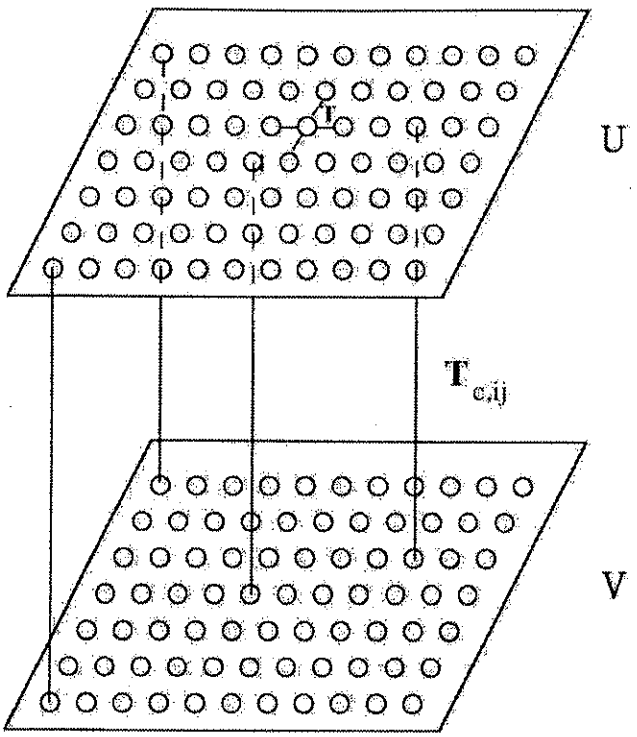


Figure 4.6: *Mapping optic flow vectors (u,v) onto DAP layers of processors.*

```
DO I = 1, MAXITER
  CALL FINDVELOCITIES(U_new,V_new,U_old,V_old,H_lines,V_lines,COUNT)
  CALL LINEPROCESS(New_hlines,New_vlines,U,V,Old_hlines,Old_vlines)
CONTINUE
```

where subroutine FINDVELOCITIES computes the smoothest velocity field given a certain distribution of lines. This is a parallel simulation of the resistive network. Subroutine LINEPROCESS computes the new estimated values for the binary line processes. This serves as the digital cycle of the network. MAXITER holds the number of times the two cycles are performed and COUNT holds the number of iterations performed in each cycle in order to reach a smooth velocity field.

Here we compute the line variables by using a parallel approach on the DAP. Following Murray, Kashko and Buxton [97], we used a chessboard pattern approach to

exploit the local information that is needed for the computation of the line variables. Consider the horizontal and vertical line variables mapped to two DAP 2D arrays and each array arranged alternately into white and black squares as in a chessboard pattern, shown in Figure 4.7. The center element of the chessboard corresponds to line variable $l_{i,j}$.

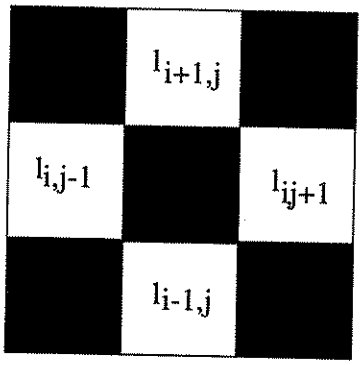


Figure 4.7: *The horizontal and vertical line variables mapped to two DAP 2D arrays and each array is arranged alternately into white and black squares as in a chessboard pattern.*

The deterministic method used for the computation of the line variables searches for the lowest energy configuration by changing the value of one of the line processes while keeping the rest constant. In this case, synchronous updating of the line variables will be disastrous. By updating, the black squares separately from the white ones, however, an asynchronous update of the line variables can be achieved and, therefore, line processors that change simultaneously are not neighbours. On an SIMD machine one can change every other row and column synchronously, and so all the line processors can be visited in just two synchronous updates. On the DAP it is easy to prevent broadcast commands reaching certain processors using a logical mask similar to that of the chessboard pattern in Figure 4.7. For computing the line variables we can first only update the black squares for the horizontal line processes:

```
Horizontal_oldlines = H_lines;
Vertical_oldlines = V_lines;

/* the values of the horizontal lines associated
   with black squares are inversed  */

Board(Black, H_lines) = NOT(H_lines);

/* evaluation of the new energy value */

Out = ENERGY(H_lines,V_lines,Const);

/* evaluation of the old energy value */

Outold = ENERGY(Horizontal_oldlines,V_lines,Const);

Not_accepted = FALSE;

/* find out which line setting  lower  the overall
   value of the energy function  */

Net = Out - Out_old;

Not_accepted(Net.ge.0) = TRUE;

/* the lines that don't lower the overall value of
   the energy function  and are associated with black
   squares are given their old value  */

H_lines(Not_accepted and Black) = Horizontal_oldlines;
```

Then we follow the same procedure for the vertical lines, before repeating these for the white squares of the chessboard.

4.4.4 Computing the Continuous Line Variables

In section 4.3 an energy function was described that includes continuous line variables, for the detection of motion discontinuities in the image. The best values of the line variables are computed using Equation (4.14) for the horizontal and vertical line variables. As shown in section 4.3 Equation (4.14) corresponds to the update equation of the continuous Hopfield model. Therefore, treating (u, v) as constants the Hopfield model will converge to the values of l^h, l^{vr} that minimise the function $E(u, v, l^h, l^{vr})$.

In a way similar to the computation of the boolean line variables, the smoothest velocity field can be computed by initially setting the line variables to 0.5. Then the values of the continuous line variables are evaluated so that the overall energy is lowered. The line variables are mapped onto 32 × 32 2D DAP arrays, and Equation (4.14) sets the elements (line processes) to the value that minimise the overall energy in the Hopfield model. The minimisation of $E(u, v, l^h, l^{vr})$ using this approach is described as:

```
DO I = 1, MAXITER
  CALL OPTIC_FLOW(Net_Constants, H_Lines, V_Lines, U, V, COUNT)
  CALL HORIZONTAL(U, V, Constants, H_Lines, V_Lines)
  CALL VERTICAL(U, V, Constants, H_Lines, V_Lines)
CONTINUE
```

where subroutine OPTIC_FLOW computes the smoothest velocity field given a certain distribution of lines; subroutine HORIZONTAL sets the horizontal line variables to the values that minimise the overall energy value; subroutine VERTICAL sets the vertical line variables to the values that minimise the overall energy value; MAXITER holds the number of iterations the two cycles are performed; COUNT holds the number of iterations performed in each cycle to smooth the velocity field.

The network that computes the optic flow field is iterated (COUNT = 10) for every single update of the line process network. Functionally, this is equivalent to assuming that the line process network is stationary or substantially slower than the optic flow network.

4.5 Summary

As stated in Chapter 2 optic flow is a rich source of information which can be used for estimating the centroid of moving objects and consequently creating models of their motion trajectories. This requires an efficient computation of optic flow and motion

discontinuities.

In the previous chapter we showed how optic flow can be computed on resistive neural networks. In this chapter we presented a hybrid parallel implementation on the DAP that consists of Horn and Schunck's optic flow algorithm and the detection of discontinuities on a continuous Hopfield model.

In particular we showed that the process of determining the optic flow vector (u, v) by minimising the error function in Horn and Schunck's algorithm is similar to the behaviour of a continuous Hopfield model with linear processing elements. Furthermore, in section 4.4.2 we presented an implementation of Koch *et al* linear resistive network on the DAP and showed that the components u and v of the optic flow can be mapped on layers of DAP processors.

Next we showed how the error function for the computation of optic flow can be augmented to include continuous line variables that detect motion discontinuities. A solution to the value of such continuous line variables is given by mapping them to the processing elements of a continuous Hopfield network.

Finally, we presented how the value of both binary and continuous line variables can be computed on the DAP by using a chessboard approach. As the determination of the lowest energy configuration requires the asynchronous update of neighbouring line variables we exploit the masking capabilities of DAP to alternatively update the white and black squares of a chessboard pattern.

The results from measuring optic flow and detecting motion discontinuities with synthetic data on the DAP were shown.

Chapter 5

Feature Extraction for Cell Recognition in Cytological Images

As stated in Chapter 2 one source of information in motion-based recognition is the motion trajectories of objects. These can be computed by tracking distinctive features of objects, and therefore the efficient and effective extraction of features is particularly important.

In this chapter we examine the extraction of features from cancer cells using feed-forward neural networks. In particular, we are looking at a class of cancer cells that has a high probability to metastasize. The behaviour of such cancer cells in vitro is usually described in terms of a number of phenomena. Among them, *in vitro* migration is considered the most important. The migrational activity of these cells is assessed by a measure of direction defined as the ratio of the net displacement (ND) of the cell to the total distance travelled (TDT) by the cell, i.e. *directionality = ND/TDT*. ND and TDT can be computed by determining the position of the cells and capturing their spatio-temporal motion patterns [135, 136, 153]. As the migrational activity of the cells with a high potential to metastasize is distinctive different to that of the cells with a low potential to metastasize [152], motion-based recognition can be used to discriminate between these two classes of cancer cells [1]. The motion models

[1] This requires a large amount of cytological images that were not available at the time this thesis

of their behaviour can be also be augmented by taking account of changes in their shape and texture[135, 136, 153].

The automatic extraction of features from cytological images is important but also challenging. It is important because the manual interpretation of cytological images is time consuming and requires the use of highly skilled cytotechnicians which is prone to human misinterpretation. It is challenging for the following reasons:

1. Microscopic and cinema-scopic techniques provide bad imaging quality.
2. The shape, size and texture of the cells is dynamic as it changes according to the phase status of the cell and its interactions with the environment.
3. The effective detection of the shape of the cells is reduced by the limitation of the images processing techniques, such as those described in Chapter 3.

To identify the position of cells and model their spatio-temporal patterns we need to extract their centroid [113]. This process differs from the feature extraction described in section 3.2.2 because instead of associating a set of images to a class we need to compute the centroid of the features presented to a feed-forward network.

In the following sections, we present our experiments in computing the centroid of cancer cells using the feed-forward network architectures described in section 3.2.2 and trained with the back-propagation learning algorithm.

5.1 Characteristics of the Training Data

Training feed-forward networks to compute the centroid of cancer cells can be achieved by associating each cell with an image of its centroid area. The input images were

was carried out.

obtained from time-lapse sequences and show cancer cells in various phase and contact states. They are preprocessed and segmented into 50 × 50 pixel images each containing one cell only. The output images depicted a grey-scale area around the centroid of the cancer cells that we refer to here as *centroid area.* The centroid areas were precomputed using Walker's algorithm [137] and resulted in a set of 50 × 50 output images. Walker used a Canny edge detector to delineate the boundaries of the cells and subsequently compute their centroid. The actual centroid of the cells in the output images is referred to as *centroid coordinate* and is computed by calculating the center of mass [125]. An example of the cell images used for the computation of the cell centroid and its associated centroid computed using the algorithm from Walker is shown in Figure 5.1.

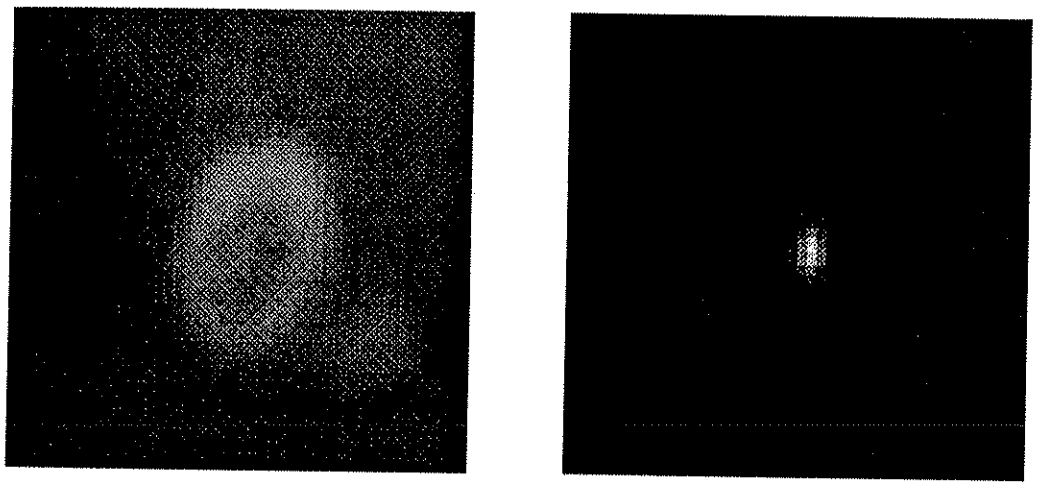


Figure 5.1: *A typical cell (left) and the computed centroid (right) using Walker's algorithm.*

Since the number of training data available was small, we scaled both input and output images down into 10 × 10 pixel images in order to reduce the degrees of freedom of the network.

The training set consisted of 100 images and the test set consisted of 20 images. To train the network we used the batch back-propagation learning algorithm described

in Chapter 3. In this case the weights of the networks were updated after the presentation of all 100 images, using Equation (3.8). The value of the momentum parameter was kept constant at $\alpha = 0.9$, and the learning rate η was varied manually depending on the state of the training process.

5.2 "All Connected" Feed-forward Neural Networks

For training an "all connected" network with 10×10 pixel images a two layer architecture was used consisting of a 10×10 element input layer, a 10 element hidden layer and a 10×10 element output layer. This resulted in a network that consisted of 210 (100 + 10 + 100) elements and 2000 (1000 + 1000) weights.

To measure the performance of the network, an average displacement of the centroid coordinates of the cells is used. The *centroid displacement* is defined as the distance between the expected (measured from the output data) and the computed (calculated by the neural network) centroid coordinates of the cells. The computation of the training and test error depends on the one-to-one difference between the grey-scale pixel value of the computed and output centroid area images. The average centroid displacement is the sum of the centroid displacements in each image divided by the total number of images. As the computation of a centroid takes into account the distribution of the grey-scale in the images, measuring its displacement is a more accurate reflection on the learning performance of a network. We have three categories to describe the centroid displacement of cells:

on the pixel which indicates that the position of the expected and the computed cell centroids coincide in the image and that the centroid displacement is equal to 0.0.

one pixel away which indicates that the position of the expected and computed cell centroids differ by one pixel. The centroid displacement is equal to $\sqrt{2}$ if their positions are on the same diagonal, otherwise it is equal to 1.

2 or more pixels away which indicates that the position of the expected and computed cell centroids differ by two or more pixels. In this case the centroid displacement is greater than $\sqrt{2}$.

Since the centroid displacement does not provide sub pixel information, the computed centroid coordinates that are *at most one pixel away* from the actual centroid coordinates are regarded as positive results. In Figure 5.2 we plot the success percentage from the results of the "all connected" network against the number of iterations.

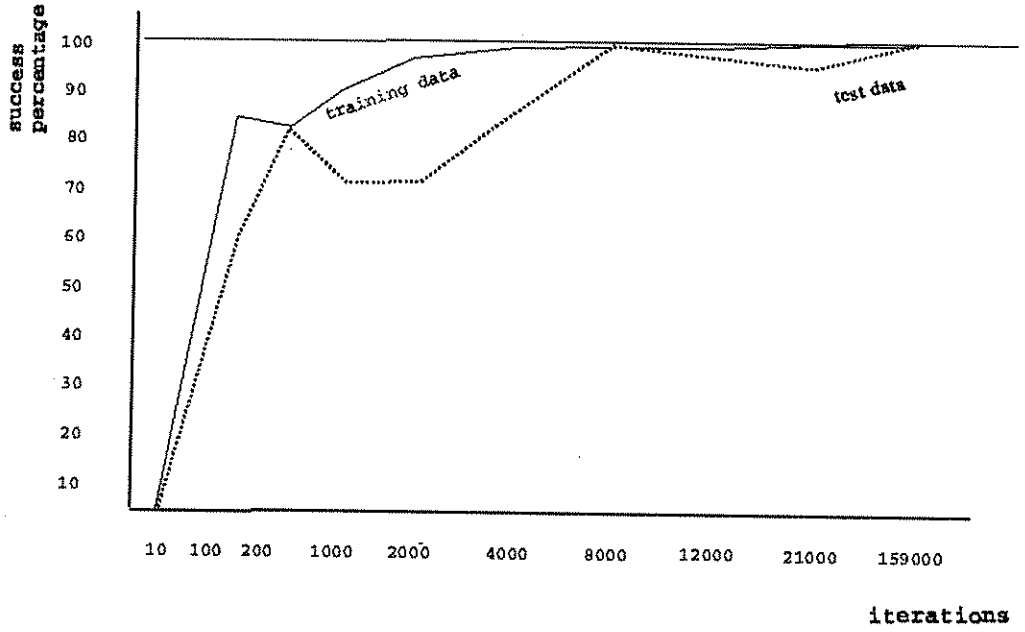


Figure 5.2: *The percentage of the success for both the training and test cell images when applied to an "all connected" network. A success includes both "on the pixel" and "one pixel away" results.*

This graph shows that the success rate of the network on the training data increases almost monotonically and reaches a 100% success rate after 8,000 iterations.

step	iterations	training error	av. displacement in training images	test error	av. displacement in test images
0.01	10	66.19	3.85	14.92	3.39
0.01	100	14.15	1.00	5.00	1.67
0.009	1000	12.23	0.86	4.77	1.56
0.009	5000	5.71	0.45	6.02	0.92
0.009	**8000**	**4.06**	**0.41**	**7.59**	**0.84**
0.009	10000	3.42	0.42	8.46	1.03
0.008	15000	2.28	0.30	9.66	1.00
0.008	21000	1.48	0.25	10.59	1.14
0.004	**159000**	**0.30**	**0.13**	**14.07**	**0.93**

Table 5.1: *Results after training an "all connected" network with 10 × 10 pixel images.*

The graph of the success rate of the test data shows, however, that there is a different evolution. It starts increasing monotonically after 4,000 iterations until it reaches 100% success rate after 8,000 iterations. Then, the success rate fails to 90% until it is restored again to 100% after 159,000 iterations. An explanation of this behaviour can be given by comparing the change in the training and test error with the change in the average displacement of the cell centroids after each iteration.

Table 5.1 shows that with each iteration, the training error decreases with the decrease in the average displacement of the cell centroids. Although, the relation between the training error and the average displacement is not linear, this tendency continues until the network converges. However, in the case of the test images, the error initially decreases but after 8,000 iterations it starts to increase. This is consistent with the training character of feed-forward networks, where a large number of iterations during training overfits the network to the training data. However, in the case of the test data, changes in the average displacement of the cell centroids do not always correspond to similar changes in the test error of the network. Furthermore, the results on the test images after 10 and 159,000 iterations shows an error of 14.92

and 14.07 but with corresponding average displacements of the centroids of 3.39 and 0.93 respectively.

	8,000 iterations		*159,000 iterations*	
	training images	test images	training images	test images
on the pixel	61%	30%	87%	15%
one pixel away	38%	70%	13%	85%
2 or more pixels away	1%			

Table 5.2: *Percentage of the average centroid displacements obtained using an "all connected" network after 8,000 and 159,000 iterations.*

The difference between the test error and the average displacement of the test data is due to the different characteristics of these two measurements: one depends on the pixel by pixel difference in intensity values, while the other is related to the distribution of the intensity values. Table 5.2 shows how the network has learned the intensity distribution after 8,000 and 159,000 iterations respectively.

	8,000 iterations		*159,000 iterations*	
	training images	test images	training images	test images
average displacement	0.41	0.84	0.13	0.93
best displacement	0	0	0	0
worst displacement	2	1.4	1	1.4

Table 5.3: *Average, best and worst centroid displacement of the training and test images using an "all connected" network after 8,000 and 159,000 iterations.*

In the case of the training data, after 159,000 iterations, the network computes the exact position of the cell centroid for 87% of the input images, compared to 61% after 8,000 iterations. However, in the case of test data, after 159,000 the network computes the exact position of the cell centroid only for 15% of the images compared to 30%

after 8,000 iterations. However as is shown in Table 5.3 the average displacement of the cell centroids in the test images has not increased considerably.

The results shown on Tables 5.2 and 5.3 are consistent with the problem of over-fitting the network. On the other hand, the success in computing the cell centroids can be explained by examining the distribution of both the centroid coordinates in the images and the computed ones from the network.

The 10 × 10 grid shown in Table 5.4 represents the size of a cell image

	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4				*1.4* 1			*1.4* 1			
5				*1* 3	*0.3* 5	*0.7* 2	*1.4* 1			
6					*1* 2					
7				*1.4* 1	*0.6* 3					
8				*1.4* 1						
9										
10										

Table 5.4: *Centroid distribution in the 10 × 10 test images and the average displacement between the actual and the computed centroid coordinates after 8,000 iterations using an "all connected" network.*

The **Lower numbers** in the elements of the grid indicate the number of cell centroids that are found at this location in the image. Therefore giving a measure of the cell *centroid distribution* in the images.

The **Upper italics numbers** in the elements of the grid indicate the *average centroid displacement* between the expected and the computed positions of the cell

centroids at this position after 8,000 iterations.

From Table 5.4 we can see that the average centroid displacement of the test cancer cells that are found near the centre of the image is of sub pixel value, whereas there are five cases of centroid coordinates away from the centre of the image where the centroid displacement is 1.4.

Based on the same convention, Table 5.5 shows the centroid distribution and the average centroid displacement of the test images after 159,000 iterations. In this case, the average centroid displacement has increased for cancer cells that are near the centre of the image. However, there is only one cancer cell that is located away from the centre of the image and its centroid displacement is 1.4.

	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4				*1.4* 1			*1* 1			
5				*1* 3	*0.7* 5	*1.2* 2	*1* 1			
6					*1* 2					
7				*1* 1	*1.1* 3					
8				*0* 1						
9										
10										

Table 5.5: *Centroid distribution of the 10 × 10 test image and the average centroid displacement between the actual and the computed centroid coordinates after 159,000 iterations using an "all connected" network.*

Comparing the results in these two tables, it is evident that initially the network computes more accurately the centroid position of the cells that are located near the centre of the image. However, after 159,000 iterations the network is specialised to sparse cells and therefore has a better ability in determining the centroid positions since it is able now to respond to sparse cells present in the test images.

5.3 "Locally Connected" Feed-forward Neural Networks

The "all connected" feed-forward architecture described in section 5.2 does not encode any *a priori* knowledge about the geometrical or topological properties of the cell images. As discussed in Chapter 3 an alternative network architecture that uses knowledge about the spatial arrangements of image intensities is that of "locally connected" feed-forward networks.

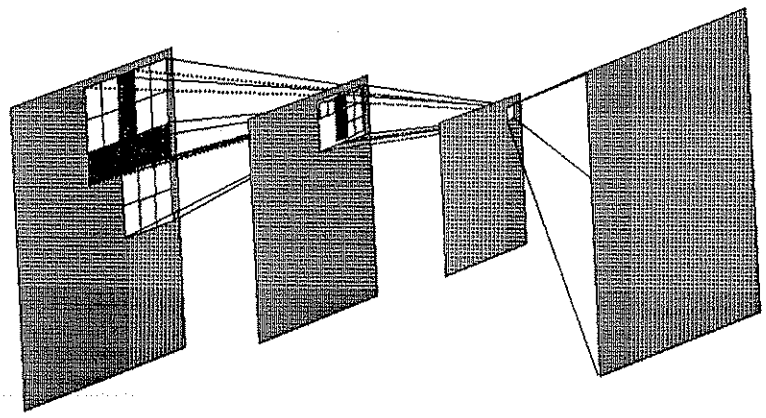


Figure 5.3: *A three layer "locally connected" network: From left to right, we have the input layer, first hidden layer, second hidden layer and the output layer. In this case, a 4 × 4 tile in the input layer is connected to a single unit in the first hidden layer using an overlapping tile structure. The first hidden layer is again divided into overlapping tiles, each one of them connected to a single unit of the second hidden layer. Finally, all the units of the second hidden layer are connected to the units in the output layer.*

For training a "locally connected" network with 10 × 10 pixel images, the three layer network shown in Figure 5.3 was used. It consisted of a 10 × 10 element input layer, a 4×4 element first hidden layer, a 3 × 3 element second hidden layer and a 10 × 10 element output layer. The input layer is divided into 16 4 × 4 overlapping tiles, the first hidden layer is divided into 9 2 × 2 overlapping tiles. All the elements of a tile in the input layer or the first hidden layer are connected into a single element of the next layer. Finally, all the elements of the second hidden layer are connected to all the elements of the output layer. This architecture brings the total number of processing elements to 225 (100 + 16 + 9 + 100) compared to that of 210 elements in the "all connected" network. However, the number of weights has been reduced to 1192 (16*16 + 9*4 + 9*100) compared to that of 2000 in the "all connected" network.

Table 5.6 shows the results after training the "locally connected" network with 10 × 10 images. Similar to the results from the "all connected" network, the error computed with Equation (3.5) does not reflect the progress on the training of the network and instead the average centroid displacement is used. The bold lines shows that the network achieves its best generalisation ability after 12,000 iterations.

step	iterations	training error	av. displacement in training images	test error	av. displacement in test images
0.009	10	13.63	1.09	4.56	1.75
0.007	100	14.28	0.98	4.9	1.67
0.007	1000	12.32	0.83	4.75	1.64
0.007	5000	5.89	0.52	5.13	1.23
0.007	10000	2.84	0.38	6.59	1.03
0.007	**12000**	**2.41**	**0.34**	**7.19**	**0.99**
0.007	15000	1.93	0.32	8.10	1.20
0.007	21000	1.29	0.3	9.00	1.22

Table 5.6: *Results after training a "locally connected" network with 10 × 10 images.*

In Figure 5.4, the percentage of success of the network is plotted against the number of iterations. In this case the network obtained a high success rate for both the training and test images after only 10 iterations. However, even though the success rate for the training images is high, it reaches only 80% success rate in the test images after 12,000 iterations.

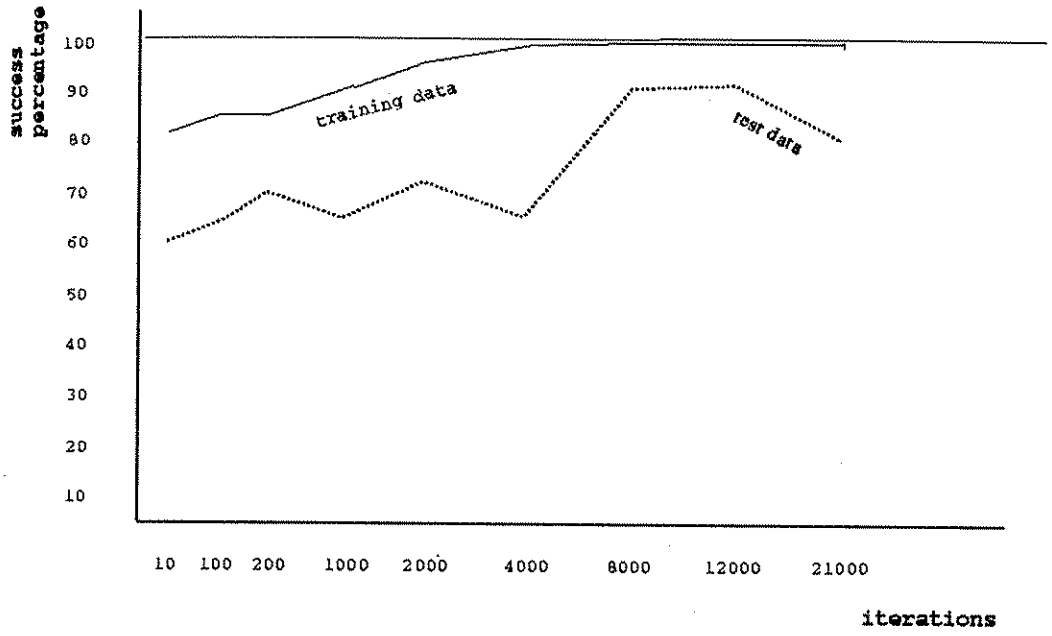


Figure 5.4: *This graph shows the success rate for the training and test images when applied to a "locally connected" network. Success includes "on the pixel" and "one pixel away" results.*

Table 5.7 shows that in 25% of the test cancer cells the network has computed the exact position of their centroid. However, for 10% of the test data the position of the centroid was computed more than 2 pixels away from the expected position. The results for the training data were similar to those given by the "all connected" network after 8,000 iterations.

	12,000 iterations	
	training images	test images
on the pixel	68%	25%
one pixel away	31%	65%
2 or more pixels away	1%	10%

Table 5.7: *Percentage of the average centroid displacements obtained using a "locally connected" network after 12,000 iterations.*

	12,000 iterations	
	training images	test images
average displacement	0.34	0.99
best displacement	0	0
worst displacement	2	2.8

Table 5.8: *Average, best and worst centroid displacements of the training and test images using a "locally connected" network after 12,000 iterations.*

Table 5.8 shows the average centroid displacements for both the training and test images. Comparing the results with that of Table 5.2, the average displacement computed by the "locally connected" network is larger, with a worst displacement of 2.8, for the test images, which indicates the erroneous computation of the location of a cell centroid more than two pixels away.

Table 5.9 shows the centroid distribution and the average centroid displacement of the test images after 12,000 iterations on a "locally connected" network. Comparing this to the results in Tables 5.4 and 5.5, it is evident that the ability of the "locally connected" network to extract the centroid of the cells is worse than that of the "all connected" network. This can be attributed to the fact that to accurately determine the centroid of the cells requires global information in the image.

	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4				*1* 1			*2* 1			
5				*1* 3	*0.5* 5	*1.8* 2	*1* 1			
6					*1.5* 2					
7				*1.1* 1	*1.1* 3					
8				*2.8* 1						
9										
10										

Table 5.9: *Centroid distribution of the 10 × 10 test images and the average displacement between the actual and the computed centroid coordinates after 12,000 iterations using a "locally connected" network.*

5.4 Parallel Implementation on DAP

For the parallel implementation of the "all connected" and "locally connected" networks we used the DAP SIMD machine described in section 4.4. In this section, we restrict our discussion only to the description of the implementations.

The two network architectures require different implementation strategies. The experiments carried out to compare the performance of "all connected" to that of "locally connected" network on the DAP have concluded that the DAP is well suited for "all connected" architectures but less so for the "locally connected" ones, due to the overlapping tiles discussed in Chapter 3. These overlapping tiles cannot be processed in parallel, and furthermore, since each tile is smaller than the array size of the DAP, the processing power of the DAP is not maximised.

The networks used for the extraction of the cell centroids have been implemented so that they take advantage of the DAP's capabilities and are as flexible as possible for the user to program. In order to achieve this, different data structures have been used in the implementation of the two networks.

5.4.1 "All Connected" Implementation

In the case of the "all connected" network, the units of the layers and weights are mapped into single structures of the form:

```
layer(*total_number_of_units)
```

Similarly, the weights of the network are held in the structure:

```
weights(*total_number_of_weights)
```

where, "*" signifies that these structures are vectors and not arrays of elements.

The implementation of the networks has been so designed that the user can specify freely the number of layers and the units on each layer without having to introduce new structures.

5.4.2 "Locally Connected" Implementation

The value of the processing elements and the weights in a "locally connected" network are held in a different structure as shown in Figure 5.5. In order to make efficient use of DAP's storage and its computational abilities, in this implementation the number of layers used are predefined in the sense that a different structure is defined for each layer of the network. Consequently, the data in each layer are held in a structure of the form:

`layer(*Height, *Width)`

where `Height` and `Width` are the height and width of that layer.

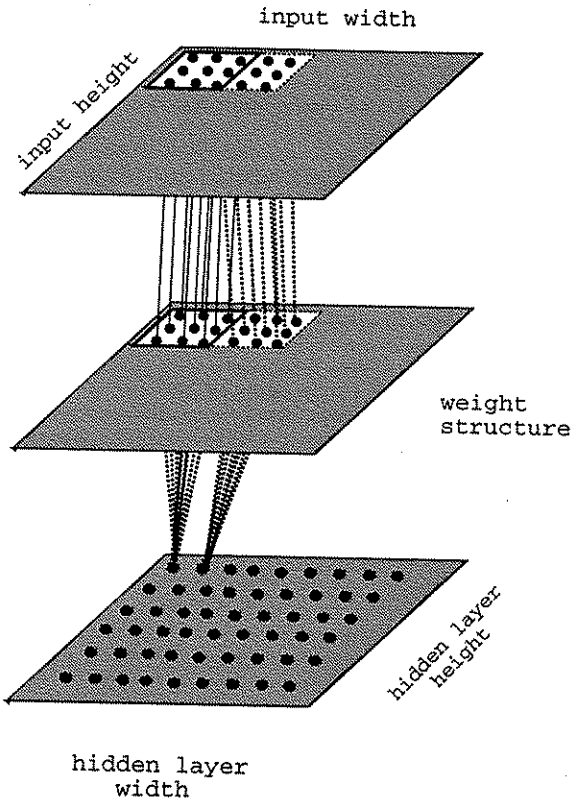


Figure 5.5: *The structures between the units of the input layer and the first hidden layer in the "locally connected" network. The input units are held in a height × width matrix and divided into 4 × 4 partially overlapping tiles. The 16 units of each of these tiles are connected to a single unit in the following hidden layer, thus the weight connection between each tile and hidden layer forms a matrix of the same size as the tile i.e. 4 × 4. All the weighted connections between the tiles of the input layer and the units of the hidden layer are held in an array of matrices that in this case has the form* `weight(tile_height, tile_width, no_of_tiles)`, *i.e.* `(4, 4, no_of_tiles)`.

The connections between the units of two layers are held in an array of matrices such as:

`weights(*Rows, *Cols, no_of_tiles)`

where `no_of_tiles` is the number of tiles in the preceding layer; `Rows` and `Cols` are the rows and columns of the weights associated with each tile. In the case where each

tile is connected to a single element of the following layer, `Rows` and `Cols` have the dimensions of the tile.

In a more general form this can be written as:

```
Rows = (tile_height x  no_of_units_the_tile_is_connected_to)
Cols = (tile_width x  no_of_units_the_tile_is_connected_to)
```

Even though this implementation limits the number of hidden layers that can be specified, it allows great flexibility in the tile size of each layer and speeds up the computation.

5.5 Summary

As stated in the previous chapter, motion trajectories of objects can be formed by determining their centroid from optic flow. However, the optic flow algorithm can not be applied to non-rigid objects such as the cancer cells.

In this chapter we showed how the centroid of cells from a set of cytological images can be extracted by using feed-forward networks. In order to exploit any structure in the cancer cells depicted in the cytological images we looked at two particular architectures: (a) an "all connected network" where all the elements of one layer of the network are linked to all the elements of the next layer and (b) a "locally connected" network where the elements in each layer are divided into tiles, and each tile of a layer is linked with a tile or element of the following layer.

The experimental results showed that the cytological images do not possess any local structure that can be exploited by the use of "locally connected" networks, and therefore they produced poorer results in determining the position of the cell centroids. However, the training pattern of the "locally connected" networks demonstrated that networks with fewer degrees of freedom were able to reach a good solution for the training data after only 10 iterations. The "all connected" networks were able

to reach a first good solution for the training data after 100 iterations, however, they showed a 100% success rate [2] in both test and training cell images.

Furthermore, we were able to investigate the behaviour of the "all connected" network and the significance of the least mean squared error computed by the back-propagation algorithm. The experiments showed that a much better indication of the generalisation ability of a network can be given by using error measurements relative to the task. In our case we used a measurement of the difference between the expected and computed position of the cell centroids, which we called displacement.

The results have shown that even though a measurement of the least mean squared error suggest an over-fitting of the network to the training data, the network was able to "learn" how to extract the centroid of the cancer cells that were located in positions away from the center of the images.

This result is attributed to the fact that the least mean squared error of a network depends on the one-to-one difference between the pixel intensity values of the computed and expected centroid area in the image. On the other hand, the displacement error depends on the difference between the intensity distribution of the image. The results have shown that the "all connected" network achieves its best generalisation ability after 159,000 iterations.

Finally, we discussed the parallel implementation of feed-forward networks on the DAP and concluded that the DAP is well suited for "all connected" architectures but less so for "locally connected" ones. This is because the overlapping tiles of the "locally connected" architecture cannot be processed in parallel. Although, a parallel implementation of the "locally connected" networks was possible, it greatly limits the flexibility in the run-time declaration of hidden layers.

[2] The computed centroid coordinates that are at most one pixel away from the actual centroid coordinates are regarded as positive results.

Chapter 6

Object Tracking Using Motion Models

There has been significant interest in computer vision, over the last 15 years in the analysis of motion sequences in order to recognize an object or its motion patterns [4, 5, 18, 23, 36, 44, 75, 88, 111, 130, 141, 148, 149, 151]. Most research has concentrated on the recovery of the three-dimensional structure of an object from motion information, and its subsequent use in the recognition of the motion of the object. Recently, however, researchers have been looking into the direct use of motion information for the construction of motion models to represent movements or activities performed by an object. The purpose of such motion models is two fold:

1. The recognition of activities performed by objects.

2. The tracking of objects in a scene.

In Chapter 2 we discussed the different data representations involved in the construction of motion models and the pattern matching techniques that can be used for the comparison of a motion pattern extracted from an image sequence with that of a stored model. Here we address data representation and pattern matching issues that are related to the construction of motion models for the tracking of moving ob-

jects in a scene. The motion models are based on trajectory data representation and neural network matching techniques. Such motion models can be applied in traffic surveillance [39, 114] and cell modelling [113] where the trajectories of objects correspond to the motion path of their mass centroid. For rigid objects, the position of the mass centroid can be determined from the computation of the optical flow field [111]. For non-rigid objects, the mass centroid can also be determined through the use of feed-forward multi-layer networks as shown in Chapter 5.

In order to track objects in the scene, one of the main functionalities of such motion models is to predict the next position of objects in the scene. Therefore, it is essential that the motion models possess the following properties:

1. To be able to associate a set of "similar" motion paths or activities of an object with a trajectory class.

2. To be able to predict the next position of an object given the current observation. In this way the motion model aids in locating the next position of the object by reducing the search space.

For an effective tracking of multiple objects in the scene, it is desirable that the motion models also fulfill the following requirements:

1. They are able to segment motion patterns in an image sequence.

2. They are able to track objects when part of a motion trajectory is occluded.

3. They are invariant to translation, rotation and spatio-temporal scale changes in the motion trajectories.

In our approach, we use neural networks for the representation of motion models because the learning properties of neural networks can be used to define classes of

trajectories through examples instead of explicitly specifying them, and the ability of neural networks to generalize can help to overcome noise due to occlusion [50, 123] .

Furthermore, as discussed in Chapter 3, partially recurrent networks can be used to model time variant information, for example, the trajectory of a moving object. Consequently, such network architectures can be used for sequence recognition, reproduction and prediction [50] and, therefore, for tracking an object by predicting its next position on the scene.

Finally, the requirements of the motion models for the prediction of motion trajectories have the following implications:

1. Each of the "similar" motion paths or activities that define a trajectory class may consist of an unequal number of elements, depending on the speed and movement of the objects. This implies that the networks that are used for representing motion models should be able to deal with input sequences of varying length.

2. The prediction of the next position of a moving object given an observation requires on-line operation by the network.

3. The motion model needs to perform only one step prediction in order to track moving objects.

A simple network that fulfils the requirements of the motion-model and also requires short training time is the Elman network discussed in section 3.4.2 [27] . Such a network is especially attractive because it can also be used for the modelling of finite state automata [19, 27, 34], which allows the temporal invariant modelling of trajectories and the extraction of points of interest [114].

There are many issues that remain to be addressed on the exploitation of fully and partially recurrent neural networks on time-series in sequence recognition and prediction. Most noticeably these include, data representations, network architectures, learning algorithms, system dynamics and the extent to which neural networks can learn and memorize temporal correlations and dependencies. There has been little work on studying neural networks for motion-based recognition with the exception of the work of Goddard [36, 35] on a connectionist representation of motion events. To our knowledge there has not been any work in the use of recurrent networks for the modelling of motion trajectories for visual interpretations. In the remainder of this chapter, we will concentrate on the modelling of motion trajectories on Elman networks and evaluate their use in tracking moving objects in a scene. We specifically address the issues of (1) representation of trajectories, and (2) memory stored in the context elements of Elman networks in order to establish the most appropriate data representation and network architectures for our task.

6.1 Shape of Motion Trajectories

Motion trajectories used in our studies are simulated and represent a set of 2D trajectories of an object tracked in time through sequences of n frames. A trajectory is defined as a sequence of locations (x_i, y_i), for $i = 1 \ldots n$, where n is the number of frames in the sequence. The motion models presented in this chapter represent motion trajectories with the shape of "figure **8**". They were chosen as a test bed because such trajectories have curvature that is not constant but varies randomly along their length. Predicting the evolution of such trajectories from a given point requires knowledge of the position on the trajectory, i.e. it depends on the spatial and temporal context of that point. Furthermore, such trajectories have sharp curvature discontinuities, which cannot be modelled by using analytic motion models. Examples of trajectories used in the training set are shown in Figures 6.1, 6.2, 6.3

and 6.4.

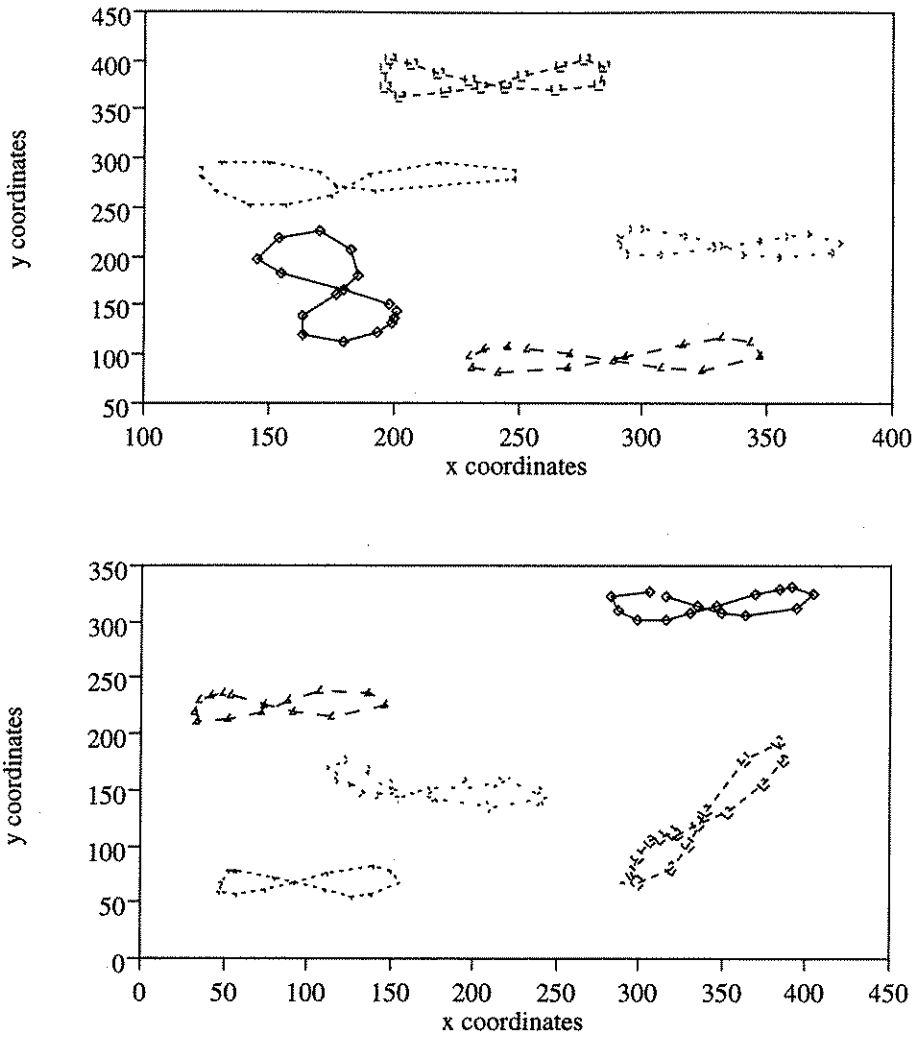


Figure 6.1: *Examples of trajectories of "figure* **8***" that are used in our training set.*

These trajectories were randomly drawn in order to introduce a degree of noise and shape variation so that the data are more realistic and statistically sound. Each trajectory was drawn inside a grid in a clockwise order starting from a similar relative position but could be centered around any position inside the grid and vary in orientation and size. The generation order of each point of the trajectories was recorded to create the set of time sequences required for the experiments. These trajectories were

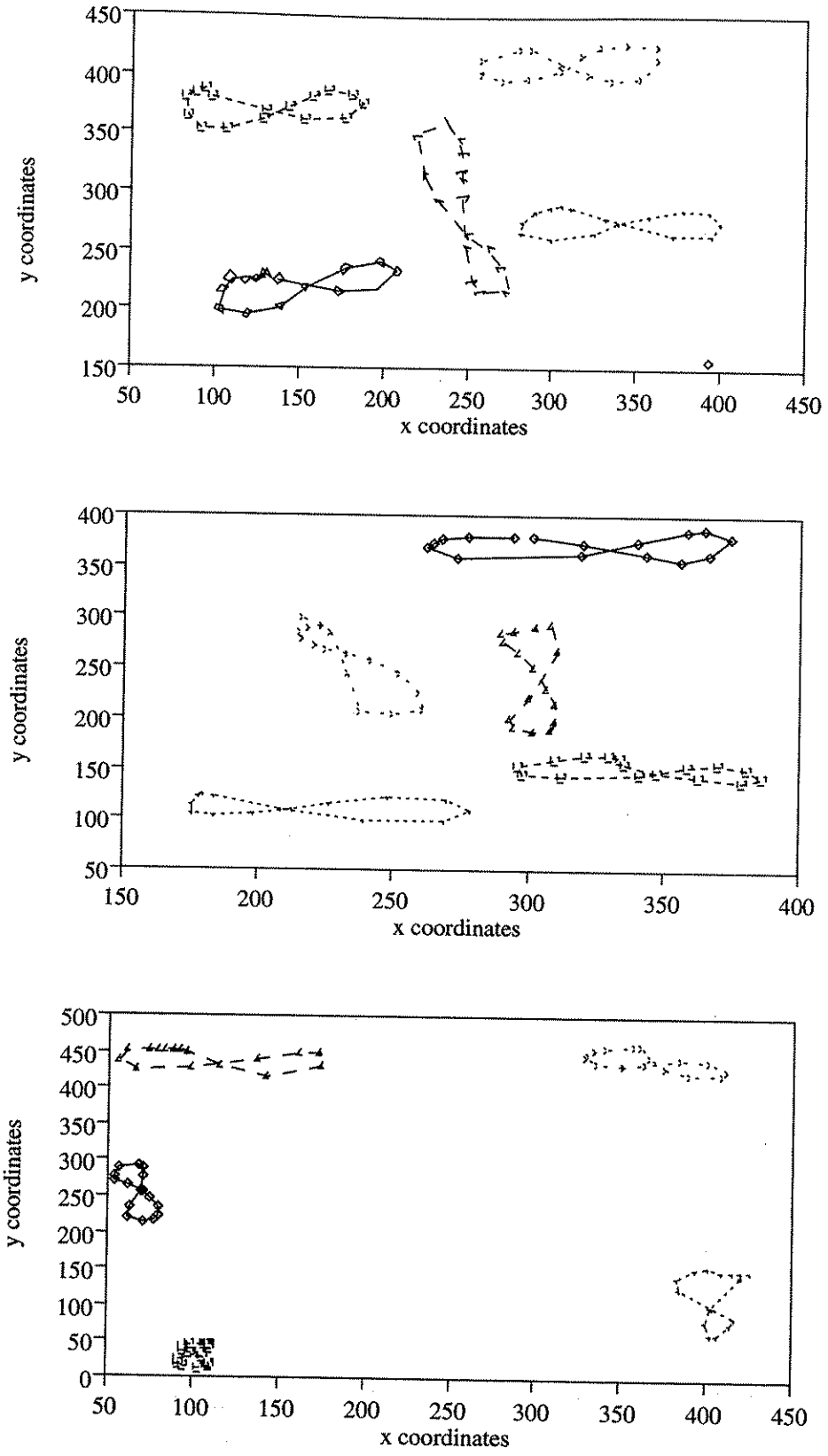


Figure 6.2: *Examples from training set (continuing).*

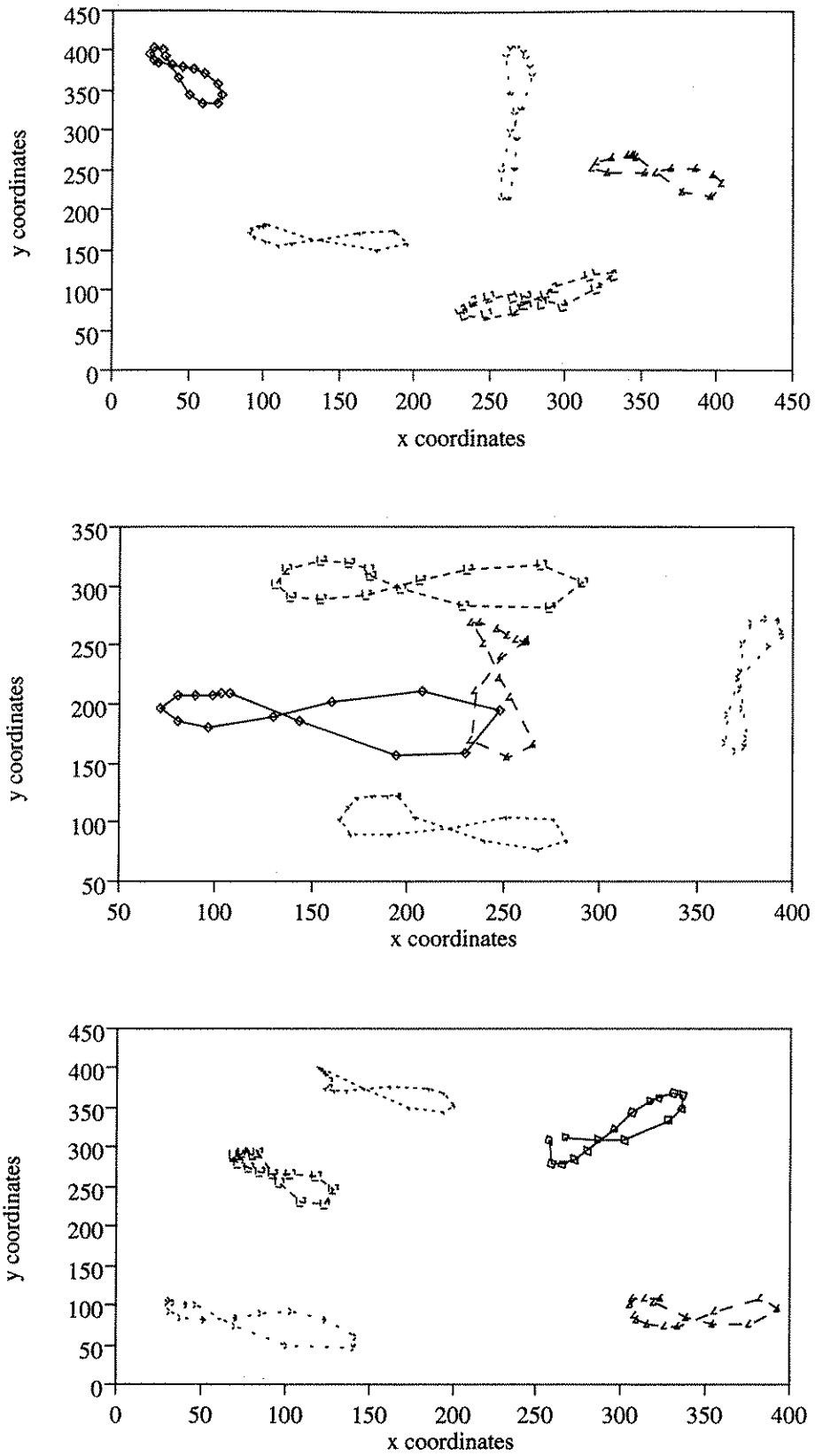


Figure 6.3: *Examples from training set (continuing).*

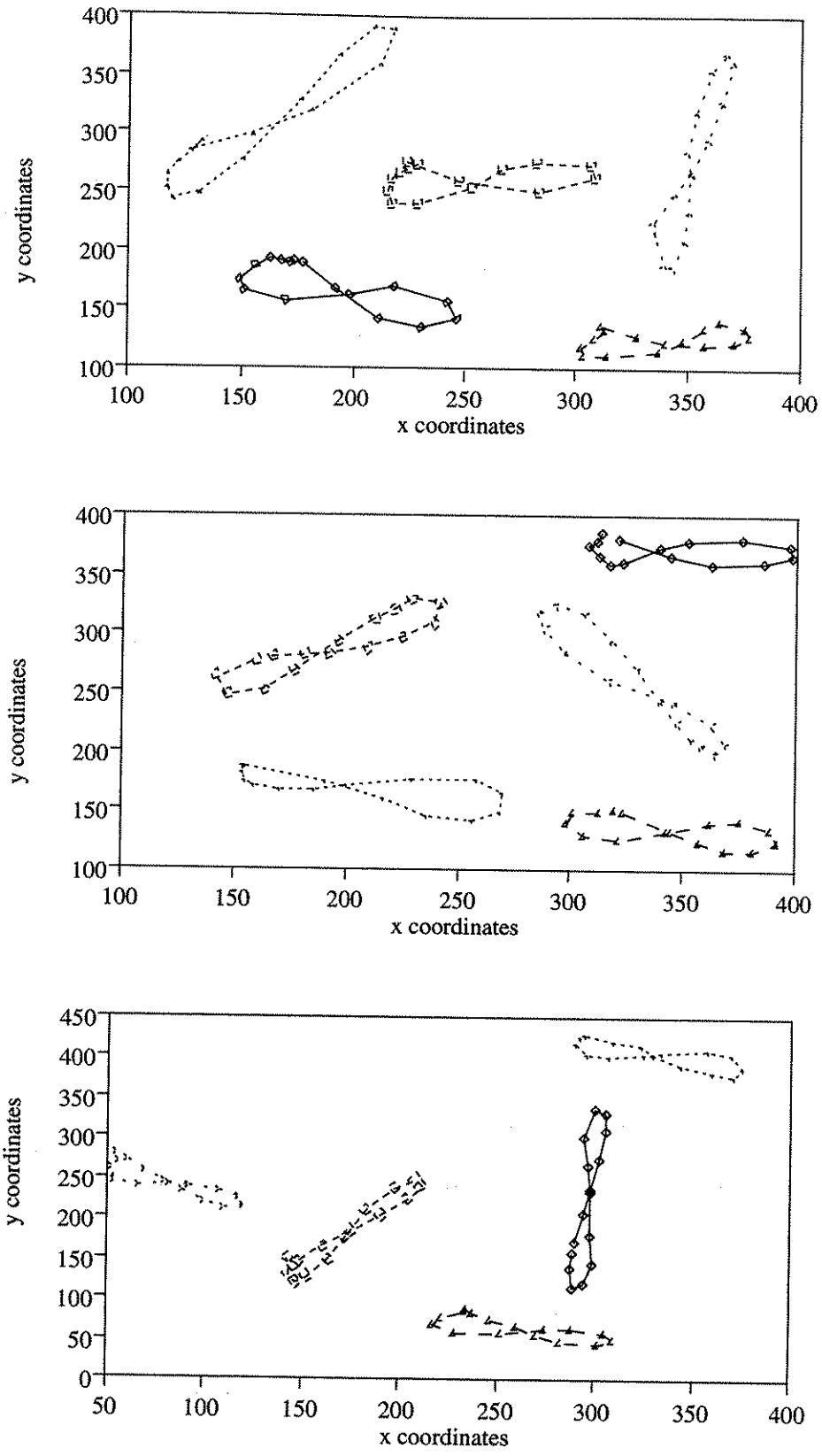


Figure 6.4: *Examples from training set (continuing).*

also distorted in both space and time [1]. All the trajectories have a sample length of 16. From a set of 100 trajectories, 70 were used to train the networks and the remaining 30 were used for testing the generalization ability of the networks.

6.2 Network Architectures

The network architectures for these experiments are simple Elman networks [27]. They consist of (1) the input, hidden, output and context elements, (2) a set of feed-forward weights, (3) a set of fixed feedback weights from the hidden to the context elements and (4) a set of fixed recurrent weights of the context elements. A network is shown in Figure 6.5.

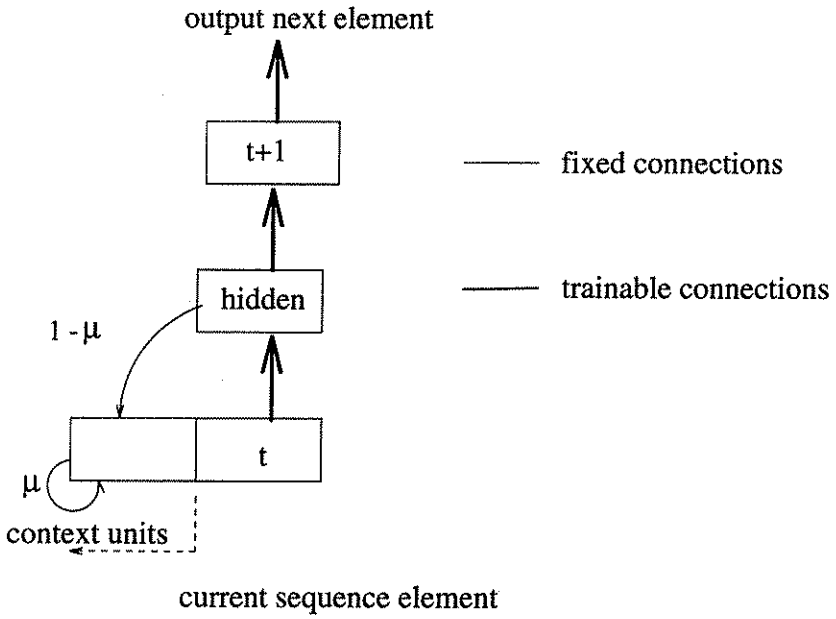


Figure 6.5: *A typical architecture of an Elman network used for modelling trajectories. The context layer is an exponential encoding of the hidden elements.*

[1] A trajectory is referred to as "space distorted" when it is expanded or reduced along one or more axis, whilst "time distortion" occurs when the evaluation of the trajectory is locally speeded-up or slowed-down.

The amount of past information held in the context elements depends on the value of the recurrent weights. In this way the context elements act as an exponential encoding of the hidden element activation values by maintaining moving averages of past hidden value activations according to:

$$y(t) = (1-\mu)\,x(t) + \mu\,y(t-1)$$

where μ lies in the interval $[-1, 1]$ and allows for the representation of average spanning various intervals of time; $x(t)$ represents the vector of the hidden element activation values at time t and $y(t)$ represents the context vector at time t (if $\mu = 0$, the context holds a single copy of the previous hidden element activations).

Back-propagation with momentum given by Equation (3.8) was used as the training algorithm for all networks. The weights were initialised between [-1.0, 1.0] and the context elements were initialised to 0.5. Typically, the momentum μ was set to 0.5 and the learning rate η varied between 0.2 to 0.0001.

A measure of the sum squared error (SSE) given by $\Sigma_{all\ pattern}|T-O|^2$ was used as the learning criteria during training, where T is the expected output of the network and O is the predicted output. With our trajectories of generalized "figure **8**" shape, the best network was selected by comparing their ability to restore the topological features of the shape of the trajectory using the chord length distribution [129].

6.3 Representation of Trajectories

In learning to predict a trajectory with a recurrent network, it is important to have the input representation reflecting the geometric and topological features of that trajectory. This partly determines the effectiveness of learning and the network's ability to generalize and predict. A few typical representation schemes are as follows:

1. Representation of trajectories with coordinates of absolute position is translational, rotational and scale variant.

2. Curvature information provides a representation that is both translationally and rotationally invariant but is still sensitive to the size of the trajectory unless fixed sampling schemes are used.

3. Curvature information can also be described by eight qualitative direction variables, as shown in Figure 6.6. Each variable corresponds to a range of angles that span 45 degrees. This results in a coarse representation of the curvature of the trajectory, but it is a more appropriate representation for a neural network as it reduces the degrees of freedom in the training data.

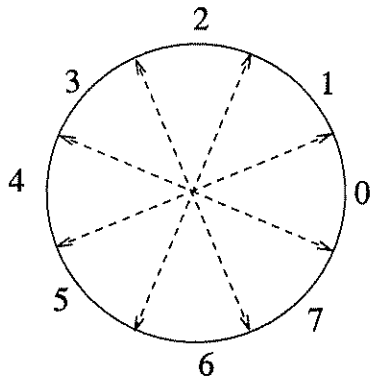


Figure 6.6: *The eight directional variables that band-limit the curvature representation of trajectories.*

During our experiments we explored a variety of representation schemes in order to determine the appropriate one for the trajectories. The representations that we used were (1) coordinate representation, (2) curvature and speed representation and (3) discrete curvature representation.

6.3.1 Coordinate Representation

During these experiments, sampled trajectories were represented with the normalised coordinate values in the range of [0.1, 0.9]. An Elman network (2 input, 8 context, 8 hidden, 2 output) was trained using the back-propagation learning algorithm given by Equation (3.8). The learning rate η was set to 0.2 and the momentum α to 0.5. With different experiments the exponential parameter μ varied between 0.0 to 0.5.

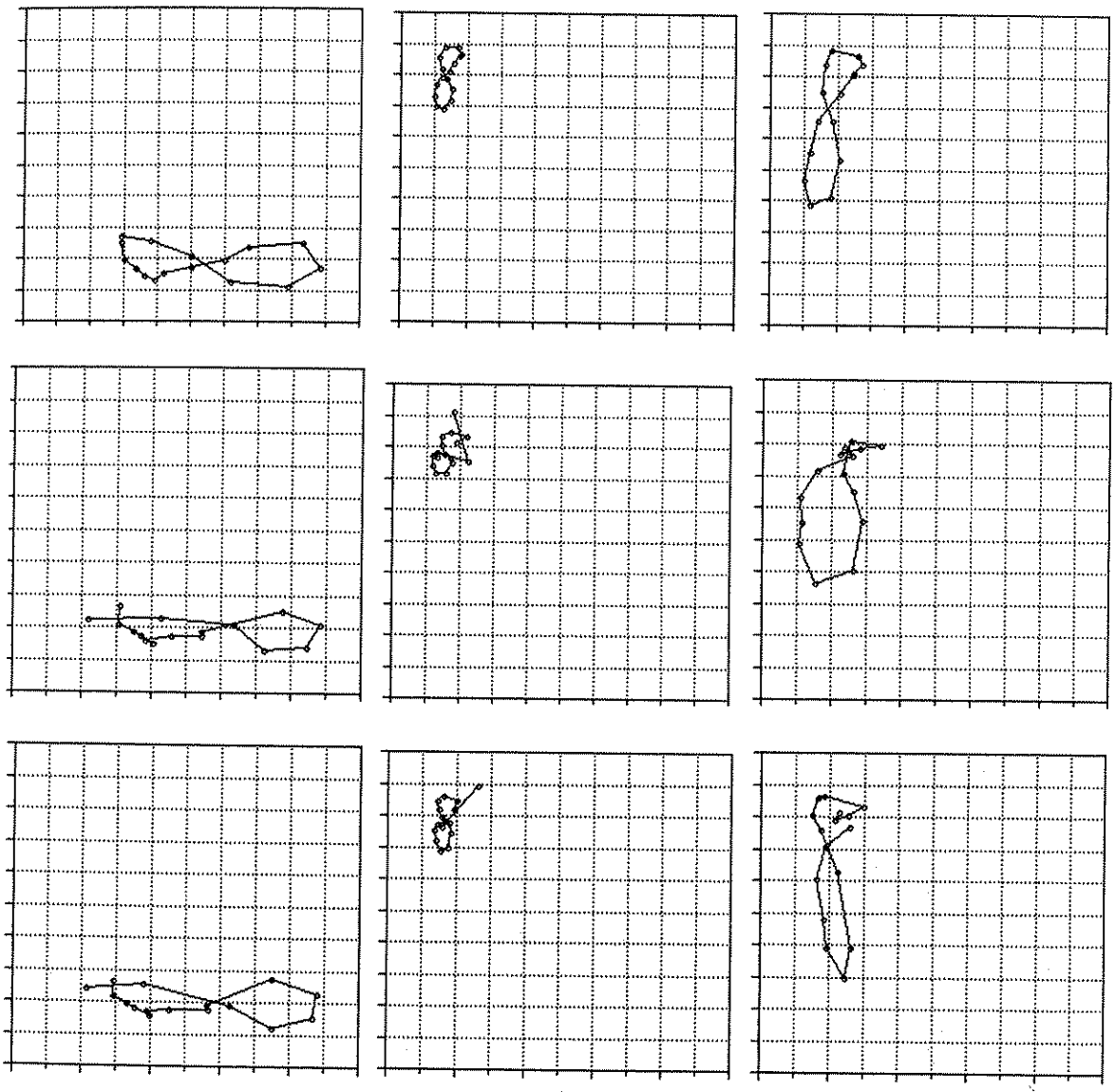


Figure 6.7: *Three different trajectories of "figure 8" (top row). Predictions of these trajectories by a network without (middle row) and with (bottom row) exponential memory.*

The network converged in 1500 epochs. However, by comparing the network's ability to restore the topological shape of the original trajectories, it was clear that the network achieved the best generalization after 600 - 1000 epochs. The output of the network on some of the testing data is shown in Figure 6.7. The first row shows the original trajectory shape of the testing data. The second row shows the predicted output of a network that has been trained without an exponential memory ($\mu = 0.0$). The third row shows the output of a network when the exponential parameter μ was set to 0.5. The results given by the second and third rows of Figure 6.7 show that the network was able to predict the evolution of the trajectories independently of their position and scale. However, when the network was trained with exponential parameter $\mu = 0.5$, it converged faster and improved its prediction at locations of high curvature along trajectories compared to those predicted by the network without the exponential memory. This can be explained as the exponential memory of the network allows for prediction of the next position by maintaining moving averages of past hidden value activations.

6.3.2 Curvature and Speed Representation

A more explicit representation of the shape of trajectories is provided by computing their curvature, which is defined as the rate of change of slope given by $\partial\theta \,/\, \partial s$, where $\partial\theta = (\theta 1 - \theta 2)$ is the angle difference between two position vectors on a trajectory and ∂s is the distance between these two consecutive positions along the curve. It was calculated in clock-wise order and $\partial\theta \,/\, \partial s$, was normalised in the range [0.1, 0.9]. Representation of a trajectories based on curvature can be further constrained by introducing speed. Speed is represented as the distance between two points given by $|\mathrm{s}| = \sqrt{(dx)^2 + (dy)^2}$. In these experiments, trajectories are represented with coupled curvature and speed measurements. An Elman network (2 input, 8 hidden, 8 context and 2 output) with exponential parameter $\mu = 0.5$ was trained with the learning

rate of 0.1 and momentum of 0.5 and it only converged momentarily. The outputs from a trained network on some of the test trajectories are shown in Figures 6.8, 6.9 and 6.10. The top picture shows the original trajectory, the middle picture shows the trajectory represented by curvature and speed and the bottom picture shows a reconstructed trajectory by network's prediction on curvature and speed.

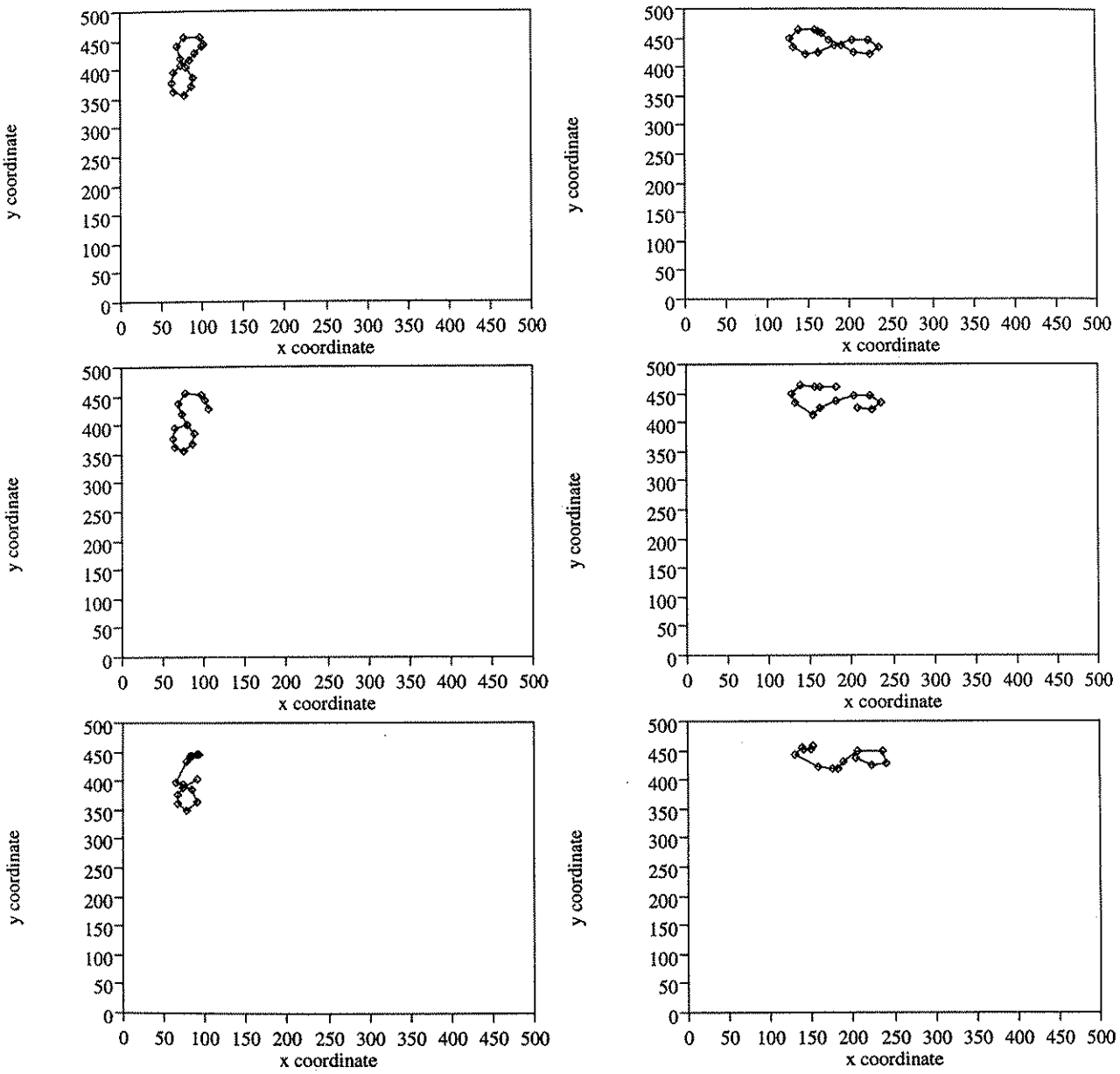


Figure 6.8: *Top: Trajectory examples 1 (left) and 2 (right). Middle: Trajectory represented by continuous curvature and speed. Bottom: Trajectory reconstruction by network curvature prediction.*

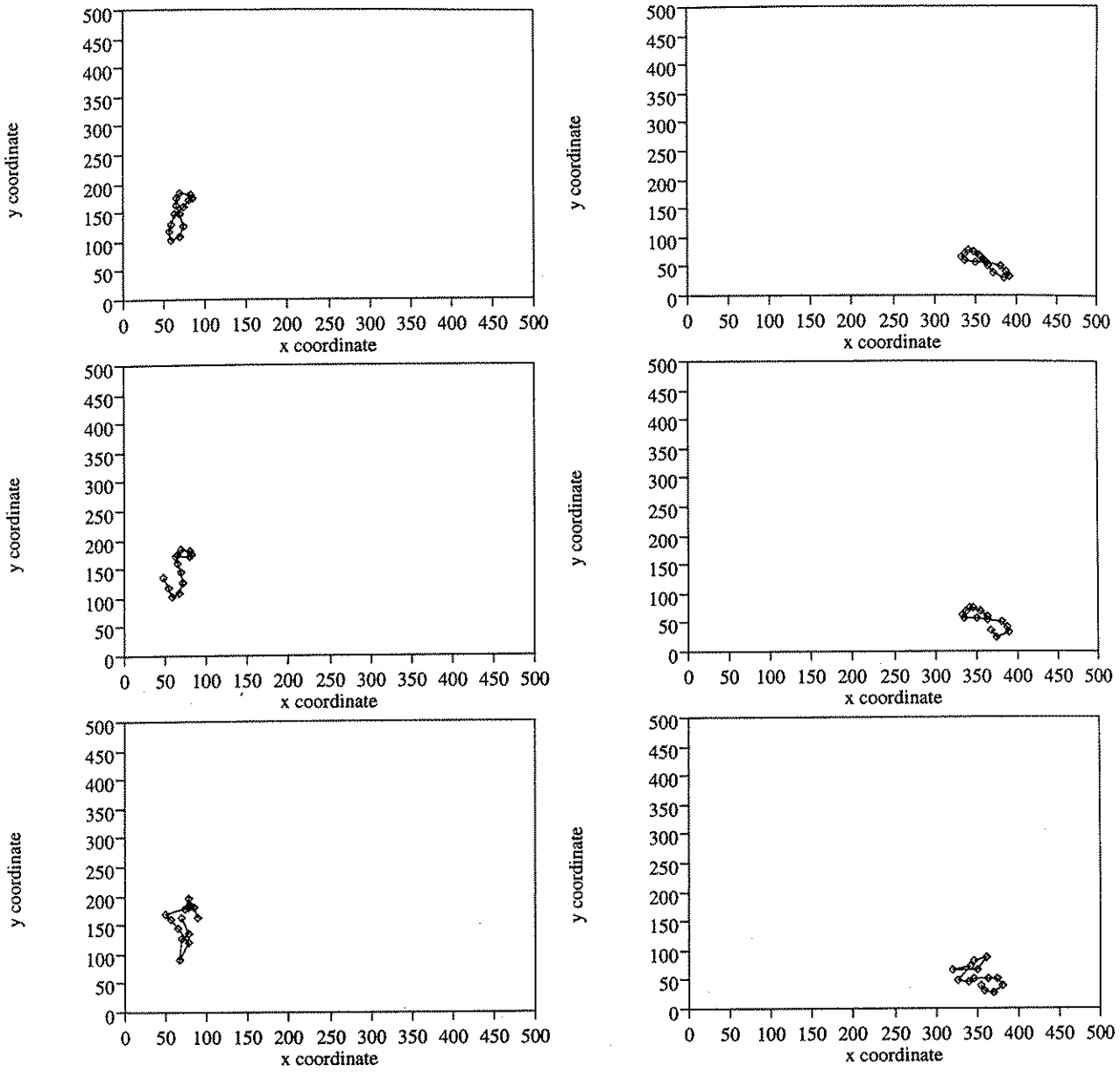


Figure 6.9: *Examples 3 (left) and 4 (right), represented by curvature and speed, reconstructed by network prediction.*

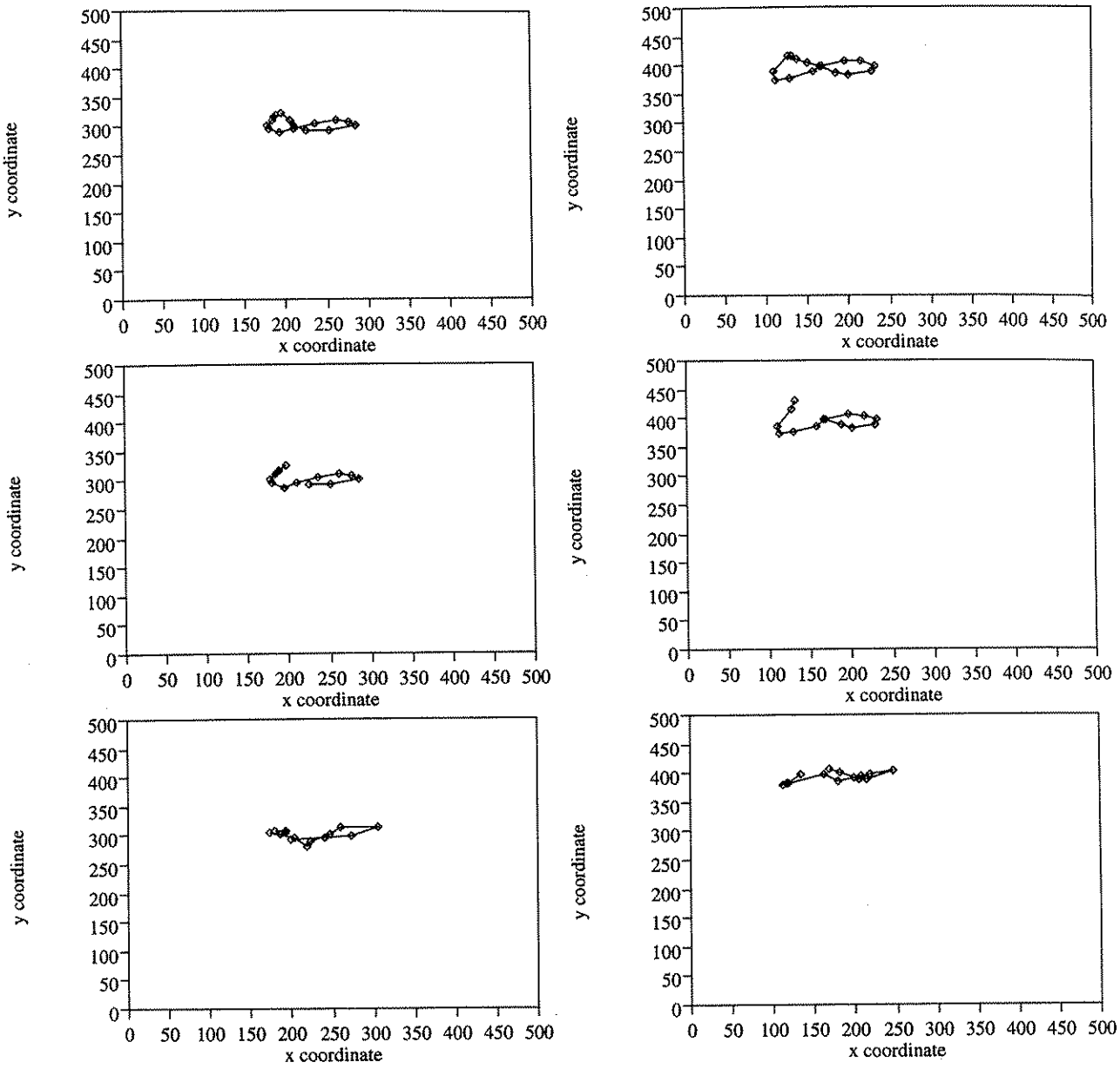
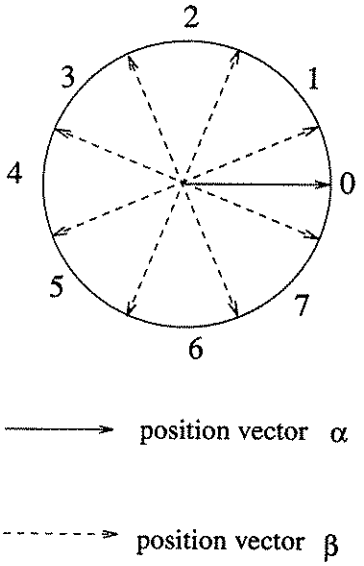


Figure 6.10: *Examples 5 (left) and 6 (right), represented by curvature and speed, reconstructed by network prediction.*

6.3.3 Discrete Curvature Representation

Another way of representing curvature information is to describe it by using the eight qualitative direction variables shown in Figure 6.11(left). Each variable corresponds to a range of angles between two successive position vectors (α, β) on the trajectory. The angles are presented to the network by using the binary representation shown in

Figure 6.11(right). This results to a coarser representation of the curvature described in the previous section, but it reduces the degrees of freedom in the training data. An Elman network (4 input, 12 context, 12 hidden and 4 output) was trained with learning rate of 0.001, momentum of 0.5 and exponential parameter μ set to 0.5. After a training session of 140000 epochs the network was able to give a good qualitative prediction on the evolution of trajectories. Some results are shown in Figures 6.12, 6.13 and 6.14. These results correspond to the same trajectory examples used in the previous section.



binary code representation	angle difference between two vectors
0 0 0	-22.5 to 22.5
0 0 1	22.5 to 67.5
0 1 0	67.5 to 112.5
0 1 1	112.5 to 157.5
1 0 0	157.5 to -157.5
1 0 1	-157.5 to -112.5
1 1 0	-112.5 to -67.5
1 1 1	-67.5 to -22.5

Figure 6.11: *The eight directional variables that band-limit the computed angle representation between two successive position vectors* (α, β) *on the trajectory (left) and their coding on partially recurrent networks (right).*

The top picture in Figures 6.12, 6.13 and 6.14 corresponds to trajectories represented by discrete curvature and an indication of the distance between two points on a trajectory is also shown. The bottom picture shows the reconstruction of the trajectories by network prediction. The results verified that a discrete representation provides a more precise prediction of the changes along trajectories, although the representation is coarser. A more fine-grain sampling would provide a more accurate

representation of the trajectories but also increase the training time of a network.

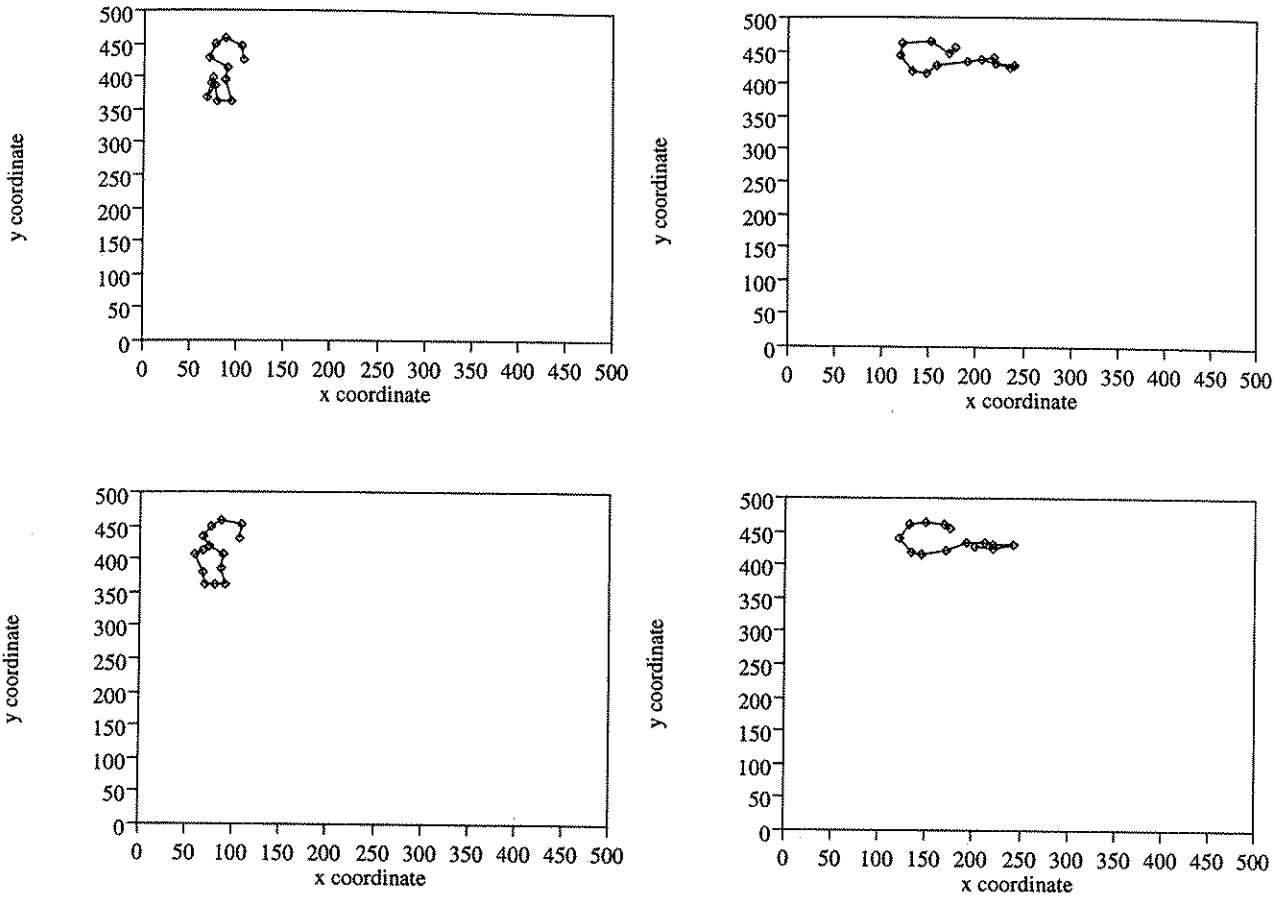


Figure 6.12: *Top: Trajectories 1 (left) and 2 (right) represented by discrete curvature. Bottom: Reconstruction of the trajectory by network prediction on curvature values.*

6.4 Summary

In this chapter we have shown that partially recurrent networks such as the one proposed by Elman [27] can be used to model complex motion trajectories. The motion models constructed with the Elman networks can be used to predict fairly accurately the next position of a moving object, and thus track an object by reducing the search space in the scene. The prediction of the trajectories is independent of their position, orientation and spatial scale.

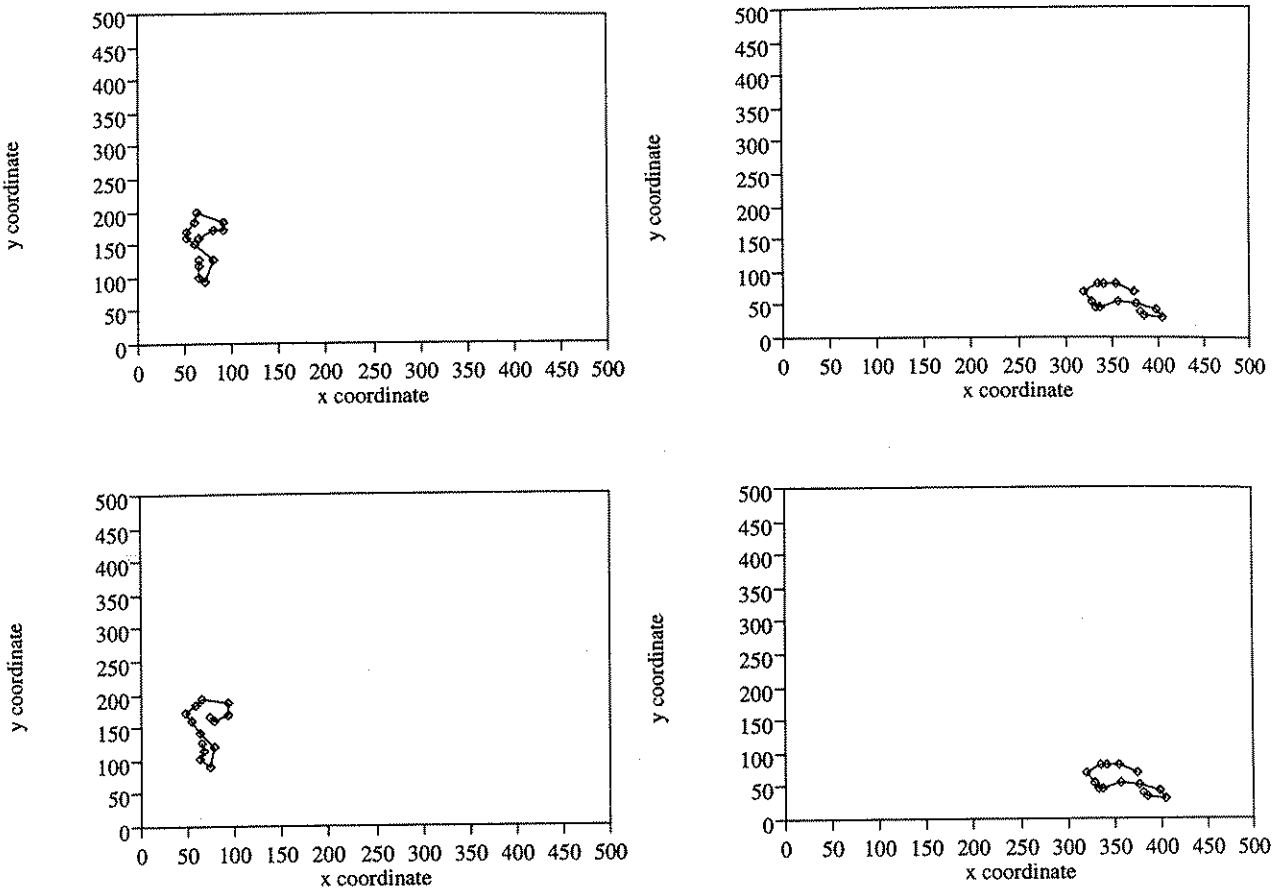


Figure 6.13: *Prediction of examples 3 (left) and 4 (right) by discrete curvature representation.*

In the construction of motion models on the partially recurrent networks we examined two different issues: (1) the memory of the network and (2) the data representation of trajectories.

Our results have shown that a network with exponential memory gives a more accurate description of the evolution of trajectories. This is attributed to the fact the exponential encoding of the activation of the hidden elements allows the network to both "learn" averages of past information from the training data but also estimate future averages using the past information of the test data. This property is particularly desirable in the case of overcoming scale variance.

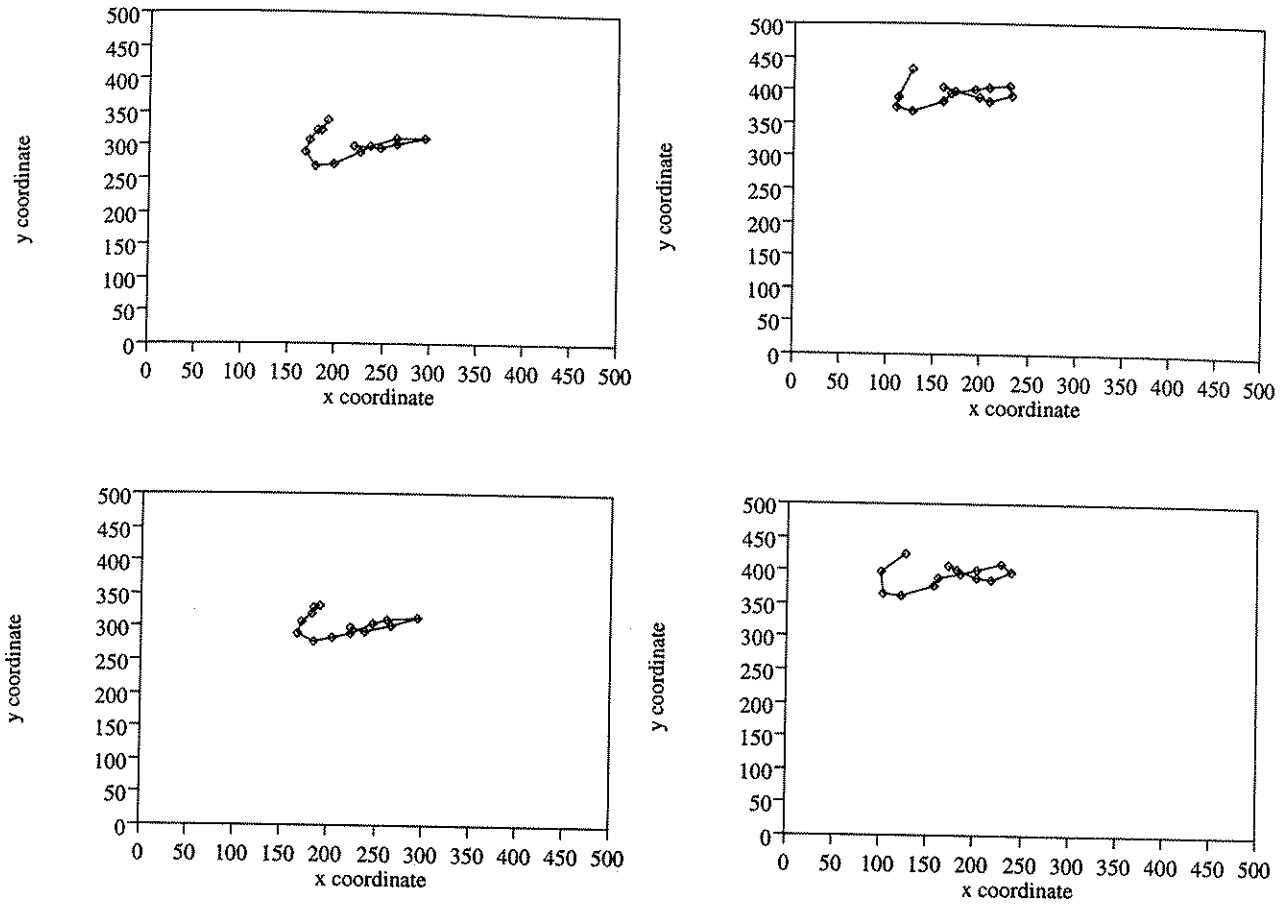


Figure 6.14: *Prediction of examples 5 (left) and 6 (right) by discrete curvature representation.*

Furthermore, we used three different representations to encode our motion trajectories on the recurrent networks: (1) coordinate representation (2) curvature and speed representation and (3) discrete curvature representation.

By using the coordinate representation, the networks were able to predict the evolution of test trajectories after only 600 iterations. The test data varied in position, orientation and size. However, such a representation does not provide temporal scale invariance. An alternative representation is that of the coupled curvature and speed measurements.

A network trained with coupled curvature and speed measurement was able to predict the evolution of trajectories but the infinite degrees of freedom given by the association of real valued data reduced the generalisation ability of the network. However, as was shown, a discrete modelling of the curvature coupled with the speed measurement along the trajectories allows for a more accurate prediction of the evolution of trajectories. This is attributed to the fewer degrees of freedom in the association between the input and output of the network and the band-limited description of the curvature.

Chapter 7

Discussions

In this thesis we investigated the potential of neural network architectures in applications related to image understanding and in particular, to that of motion-based recognition. Most of this work was conducted at the time of the re-emergence of the paradigm of neural computation and the realisation that neural networks trained with the back-propagation learning algorithm can learn complex functions through presentation of examples.

One of the main goals of computer vision is to establish the relationship between 2D images and the 3D world, and to describe the structure and behaviour of the objects in the scene. This is an ill-posed problem [7, 58, 86, 109] that requires the use of *a priori* knowledge to constrain its solution space. However, even under such constraints, establishing a function that relates objects shown in 2D images to that of the 3D world is a non-trivial task and a computationally expensive process. On the other hand, the machine learning capabilities of neural networks could provide an attractive alternative paradigm to the development of automated image understanding systems. In addition, due to the computational cost of the recognition of visual motion and the inappropriateness of the von Neumann architectures for such a task, we were also interested in exploiting any benefit that neural networks, coupled with their implementation on an SIMD parallel machine, had to offer.

For this purpose we examined the main processes required for motion-based recognition. These are (a) the extraction of appropriate information from the images in the form of observations and (b) the use of such information in the creation of motion models. Such motion models can then be used to (a) predict an observation in the next image frame and (b) verify whether the observed movement in an image sequence corresponds to the motion model.

In this thesis we addressed the problem of motion-based recognition using a unified neural network approach. In this approach neural network architectures were used for both the extraction of information from an image sequence and the creation of motion models.

Elements of this approach were described in detail in Chapters 4, 5 and 6. In particular, in Chapters 4 and 5 we described the extraction of information from an image sequence and concentrated on (a) the determination of discontinuities in an optic flow field using Hopfield models and (b) the computation of the centroid of non-rigid objects using feed-forward networks. In Chapter 6 we showed how motion models based on the geometrical positions of moving objects can be created using partially recurrent neural networks. Table 7.1 gives a summary of the neural network architectures used in this thesis and their potential in modelling processes required in motion-based recognition.

The processes required for motion-based recognition are computationally expensive and their parallel implementation is desirable. The SIMD platform provided by DAP is very suitable to the parallel implementation of both feed-forward and recurrent neural network architectures. The layers of the networks map naturally to the 2D array processors of the DAP, and maximum efficiency is achieved when the processing elements of the neural networks are updated synchronously. However, as was shown in Chapter 4, asynchronous operations can be performed efficiently on the

Neural network architectures	Motion-based recognition processes	Advantages	Disadvantages
Hopfield models	Computation of optic flow field	Fast minimisation of energy functions if implemented in hardware	Stop at local minima
Feed-forward networks	Computation of centroids	Learn through examples Robust and fast computations	Long training times
Partially recurrent neural networks	Creation of motion models	No need for temporal windows Create moving averages from past information Can change predictions according to past observations On-line recognition Deduce results from partial information	Only one step ahead prediction

Table 7.1: *Summary of the advantages and disadvantages of the neural networks used in this thesis for the modelling of motion-based recognition processes.*

DAP by using the masking and indexing operators available. This results in a very compact programming code. Partially recurrent neural networks can also be mapped on the DAP, however their implementation was not attempted because the DAP was no longer fully maintained by the Department of Computer Science at Queen Mary College. In addition, at the time the partially recurrent networks were implemented much more powerful SUN 4 workstations were available, and therefore therefore their coding on the DAP was not considered to be as vital as before.

7.1 Contributions of the Thesis

The major contributions of this thesis can be summarised as:

- **Computation of optic flow on the DAP and the use of Hopfield models in the detection of motion discontinuities:** Optic flow corresponds to the apparent motion in a sequence of images. It is a computationally expensive process and its efficient computation is one of the essential requirements in motion-based recognition. Following the work of Koch *et al*, we showed how the algorithm developed by Horn and Schunck and extended by Koch *et al* can be seen as the minimisation of the Liapunov function of a Hopfield model with linear processing elements. Furthermore, following Koch *et al* [71, 72] we showed how their energy function that describes depth measurements can be altered for the determination of motion discontinuities. Finally, we showed how the process for the detection of motion discontinuities in optic flow can be mapped onto a continuous Hopfield network.

- **The use of feed-forward networks in the extraction of centroids of cancer cells from cytological images:** To determine the location of an object in a sequence of images requires the extraction of features, such as the centroid of the cancer cells. In cytological images, this is a difficult and computationally expensive process because of the bad quality of the images and the computational limitations of the traditional feature extraction techniques. We used "all connected" and "locally connected" feed-forward network architectures to extract the centroid of the cancer cells through the presentation of examples. Through the experiments, we confirmed that the generalisation ability of a network is better described by error measurements which are related to the function that the network is trying to "learn". During our experiments, an indication of the generalisation ability of the network was given by the difference between the

expected and computed locations of the centroid of the cancer cells. Finally, we have shown that the "all connected" network was able to extract the position of the centroid of the cancer cells and that its generalisation ability was better than that of the "locally connected" network. This indicated some lack of local structure in the cancer cell images used.

- **The use of partially recurrent networks in the construction of motion models for trajectories that include sharp curvature changes:** An important aspect of motion-based recognition is the construction of motion models. We showed that simple partially recurrent neural networks can be used for modelling motion models through the presentation of examples. We based our motion models on both coordinate and coupled curvature and speed representations. We experimented with a set of complex motion trajectories of shape of "figure 8". Such trajectories contain sharp curvature changes that are difficult to model by analytic motion models. The best prediction of the evolution of these trajectories was given by a network with exponential memory and a coupled discrete curvature and speed representation.

- **The implementation of Hopfield models and feed-forward networks on the DAP:** The efficiency of neural networks can be further increased when their processing power is coupled with a parallel implementation. We showed how the processing elements of a Hopfield model can be updated asynchronously on a DAP and why DAP is naturally suited for the implementation of feed-forward networks.

7.2 Future Work

There are many issues that remain to be addressed on the application of fully and partially recurrent neural networks to the prediction of temporal information and

particularly, in the context of motion-based recognition. These include data representations, network architectures, learning algorithms, system dynamics and the extent to which neural networks can learn and memorise temporal correlations and dependencies. In particular, we would like to investigate the following issues:

- The extension of the discrete curvature representation to trajectories of varying length. In this case the temporal variance problem can be overcome by describing the curvature of the trajectories using regular grammars. This is possible since most objects in a scene move purposefully in their environment and there are spatio-temporal regularities associated with their motions [39]. The shape of "figure **8**" can be described with a regular grammar by using eight direction symbols $\{0 \cdots 7\}$ that denote the next possible curvature along the trajectory and four state symbols $\{S_1, S_2, S_3, S_4\}$:

$$\begin{aligned} S_1 &\rightarrow 5S_2|6S_2| < rest > S_1 \\ S_2 &\rightarrow 5S_3|6S_3| < rest > S_2 \\ S_3 &\rightarrow 2S_4|3S_4| < rest > S_3 \\ S_4 &\rightarrow 2S_1|3S_1| < rest > S_4 \end{aligned}$$

  where the notation $< state > \rightarrow < symbol > < state >$ indicates a legal string in the grammar.

  Regular grammars can be recognised by finite state machines (FSM) [53]. Figure 7.1 shows a trajectory of shape "**8**" and its representation by FSM using discrete curvature information.

  The states of an FSM correspond to the location of sharp curvature changes and the symbols on the transition arcs correspond to a set of allowed curvature changes along the arc. Techniques for learning FSMs using partially recurrent neural networks were first explored by Elman and Cleeremans [19, 27]. This

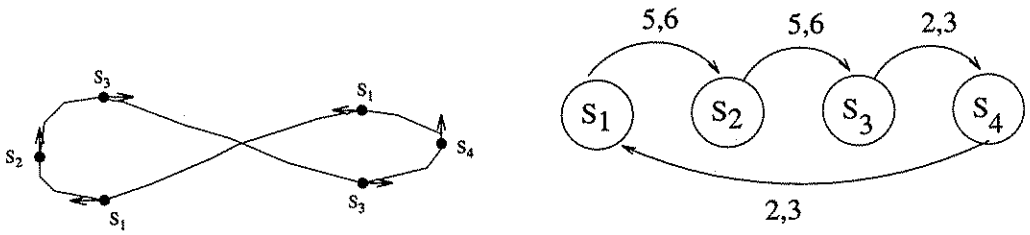


Figure 7.1: *A trajectory of figure eight (left) and its finite state machine representation (right).*

FSM representation of the trajectories has close connections with representation of moving objects using Hidden Markov Models [39]. However, by learning the association of the location of sharp curvature changes with the states of a finite machine, a recurrent network will be able to exhibit a behaviour that is invariant to temporal scale. Furthermore, analysis of the activation of the elements in the hidden layer can provide an insight on the positions of sharp curvature changes along the trajectories.

- The recognition of temporally invariant information: This requires the encoding of long term time dependencies. For this purpose, elaborate networks such as second order networks [34, 43] may be more appropriate.

- The prediction of the evolution of trajectories needs to be investigated in relation to long term occlusions. Such tasks require networks that behave well in the case of a large amount of missing data. A promising solution is the use of time-delay radial basis function networks [2].

- Our experiments on the memory of the recurrent networks have shown that although memory played a major role in the prediction of the evolution of trajectories, the predictions did not vary significantly with different memory functions. This is an area that needs further investigation and a relationship

needs to be drawn between the predictions of different memory functions.

- Motivated by the work carried out in this thesis, we have already shown that partially recurrent networks can be applied to the dynamic recognition of faces [41, 115]. We are currently exploiting a unified approach where faces are both tracked and recognised using partially recurrent neural networks.

In summary, we showed that neural networks can be used for modelling processes for extracting observations and constructing motion models. In the past years, progress has been made in the computation of optic flow and the extraction of features from cytological images. However, only until very recently researchers have started to exploit the potential of recurrent neural networks and their ability to encode structured information. We have shown that partially recurrent networks are able to model highly structured information, although there are many difficult issues that remain to be addressed as in the cases of: segmentation in the light of conflicting information, the use of long term dependencies and prediction for the discrimination of partially overlapped trajectories.

Bibliography

[1] J.K. Aggarwal and N. Nandhakumar. "On the Computation of Motion from Sequences of Images - A Review". *Proceedings IEEE*, 76(8):917–935, 1988.

[2] S. Ahmad and V. Tresp. "Some Solutions to the Missing Feature Problem in Vision". In Hanson, S.J and Cowan, J.D. and Giles, C.L., editor, *Neural Information Processing Systems, 5*. San Mateo, CA: Morgan Kaufmann, 1993.

[3] K. Akita. "Image Sequence Analysis of Real World Human Motion". *Pattern Recognition*, 17(1):73–83, 1984.

[4] M.C. Allmen and C.R. Dyer. "Cyclic Motion Detection Using Spatio-temporal Surfaces and Curves". In *10th International Conference on Pattern Recognition*, pages 365–370, Atlantic City, NJ, USA, 1990. IEEE.

[5] M.C. Allmen and C.R. Dyer. "Computing Spatio-temporal Relations for Dynamic Perceptual Organization". *Computer Vision Graphics and Image Processing: Image Understanding*, 58(3):338–351, November 1993.

[6] L.B. Almeida. "A Learning Rule for Asynchronous Perceptrons With Feedback in a Combinatorial Environment". In M. Caudill and C. Butler, editors, *IEEE First International Conference on Neural Networks*, volume II, pages 609–618. New York:IEEE, San Diego 1987.

[7] Y. Aloimonos. "Purposive and Qualitative Active Vision". In *The Proceedings of DARPA Image Understanding Workshop*, Pittsburg, NY, USA, September

1990.

[8] S. Amari. "Characteristics of Random Nets of Analog Neuron-like Elements". *IEEE Transactions on Systems, Man, and Cybernetics*, pages 741–748, 1972.

[9] D.H. Ballard. "Generalising the Hough Transform to Detect Arbitrary Shapes". *Pattern Recognition*, 13:111–122, 1981.

[10] D.H. Ballard. "Parameter nets". Technical report, Computer Science Department, University of Rochester, Rochester, NY., 1983.

[11] D.H. Ballard. "Animate Vision". *Artificial Intelligence*, 48:57–86, 1991.

[12] A.M. Baumberg and D.C. Hogg. "An Adaptive Eigenshape Model". In *British Machine Vision Conference*, 1994.

[13] A.M. Baumberg and D.C. Hogg. "Learning Flexible Models from Image Sequences". In *European Conference on Computer Vision*, volume 1, pages 299–308, May 1994.

[14] A. Blake, R. Curwen, and A. Zisserman. "A Framework for Spatio-temporal Control on the Tracking of Visual Contours". *International Journal of Computer Vision*, 11(2):127–146, October 1993. Special Issue on Active Vision.

[15] V. Bruce and P. Green. *Visual Perception.* Lawrence Erlbaum Associates Ltd, Hove, Sussex, UK, 1990.

[16] H. Buxton and S. Gong. "Visual Surveillance in a Dynamic and Uncertain World". *Artificial Intelligence*, 78(1-2):431–460, October 1995. Special Volume on Computer Vision.

[17] G.A. Carpenter and S. Grossberg. "The ART of Adaptive Pattern Recognition by a Self-organising Neural Network". *Computer*, pages 77–88, March 1988.

[18] C. Cédras and M. Shah. "Motion-based Recognition: A Survey". *Image and Vision Computing*, 13(2):129–155, March 1995.

[19] A. Cleeremans. "Finite State Automata and Simple Recurrent Networks". *Neural Computation*, 1, 1989.

[20] J.D. Cowan. "Statistical Mechanics of Nervous Nets". In E.R. Caianiello, editor, *Neural Networks*. Berlin:Springer-Verlag, 1968.

[21] J.D. Cowan. "Neural Networks: The Early Days". In Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 828–842. San Mateo, CA: Morgan Kaufmann, 1990.

[22] J.E. Cutting and L. Kozlowski. "Recognising Friends by their Walk: Gait Perception without Familiarity Cues". *Bulletin of the Psychonomic Society*, 9(5):353–356, 1977.

[23] T.J. Darrell and A.P. Pentland. "Space-time Gestures". In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 335–340, New York, NY, USA, June 15-17, 1993.

[24] van A.M.J. Driel-Kulker and J.J. Ploem-Zaaijer. "Image Cytometry in Automated Cervical Screening". *Analytical Cellular Pathology*, 1:63–67, 1989.

[25] R.O. Duda and P.E. Hart. "Using the Hough Transforms to Detect Lines and Curves in Pictures". *Communications of the ACM*, 15(1):11–15, 1972.

[26] H. Dytch and G.L. Wied. "Artificial Neural Networks and their Use in Quantitative Pathology". *Analytical Quantitative Cytology and Histology*, 12(6):379–393, 1990.

[27] J. Elman. "Finding Structure in Time". *Cognitive Science*, 14, 1990.

[28] J.A. Feldman and D.H. Ballard. "Connectionist Models and their Properties". *Cognitive Science*, 6:205–254, 1982.

[29] B.M. Forest. "Restoration of Binary Images Using Networks of Analogue Neurons". Technical report, Department of Physics, Edinburgh University, 1988.

[30] K. Fukushima. "A Neural Network for Visual Pattern Recognition". *Computer*, 21(3):65–74, March 1988.

[31] S. Geman and D. Geman. "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[32] J.J. Gibson. *The Perception of the Visual World.* Houghton,Mifflin, Boston, Massachusetts, USA, 1950.

[33] J.J. Gibson. "What Gives Rise to the Perception of Motion". *Psychological Review*, 75:335–346, 1968.

[34] C.L. Giles and C.B. Miller. "Learning and Extracting Finite State Automata with Second-order Recurrent Neural Networks". *Neurocomputing*, 4:393–405, 1992.

[35] N. Goddard. "Representing and Recognising Event Sequences". In *AAAI Workshop on Neural Architectures for Computer Vision*, pages 485–496. March 22-29 1989.

[36] N. Goddard. *The Perception of Articulated Motion: Recognising Moving Light Displays.* PhD thesis, University of Rochester, 1992.

[37] S. Gong. *Parallel Computation of Visual Motion.* PhD thesis, Department of Engineering Science,Oxford University, Oxford, U.K., 1989.

[38] S. Gong. "Visual Observation as Reactive Learning". In *SPIE International Conference on Adaptive and Learning Systems*, Orlando, Florida., 1992.

[39] S. Gong and H. Buxton. "On the Expectations of Moving Objects". In *European Conference on Artificial Intelligence*, Vienna, Austria, 1992.

[40] S. Gong and H. Buxton. "Advanced Visual Surveillance Using Bayesian Nets". In *IEEE Workshop on Context-based Vision*, Cambridge, Massachusetts., 1995.

[41] S. Gong, A. Psarrou, I. Katsoulis, and P. Palavouzis. "Head Tracking and Dynamic Face Recognition". In *European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production*, Hamburg, Germany, 1994.

[42] I.E. Gordon. *Theories of Visual Perception.* John Wiley & Sons, 1989.

[43] M. Goudreau, C.L. Giles, S. Chakradhar, and Chen D. "First-order versus Second-order Single Layer Recurrent Neural Networks". *IEEE Transactions on Neural Networks*, 1993.

[44] K. Gould and M. Shah. "The Trajectory Primal Sketch: A Multiscale Scheme for Representing Motion Sequences". In *IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, California, USA, June 4-8 1989.

[45] J. Graham. "Automation of Routine Clinical Chromosome Analysis". *Analytical Quantitative Cytology and Histology*, 9(5):383–390, 1987.

[46] S. Grossberg. "Non-linear Difference - Differential Equations in Prediction and Learning Theory". *Proceedings of the National Academy of Sciences of the U.S.A.*, 58:1329–1334, 1967.

[47] S. Haykin. *Neural Networks: A Comprehensive Foundation.* MacMillan, 1994.

[48] D.O. Hebb. *The Organization of Behaviour.* New York: Wiley, 1949.

[49] R. Hecht-Nielsen. "Applications of Counterpropagation Networks". *Neural Networks*, 1:131–139, 1988.

[50] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison Wesley, Reading, Massachusetts, USA, 1991.

[51] E. Hildreth and C. Koch. "The Analysis of Visual Motion: From Computational Theory to Neuronal Mechanisms. *Annual Review of Neuroscience*, 10:477–533, 1987.

[52] D.C. Hogg. "Model-based Vision: A Program to See a Walking Person". *Image and Vision Computing*, 1(1):5–20, 1983.

[53] J. E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

[54] J.J. Hopfield. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". In *Proceedings of the Natural Academy of Science USA*, volume 79, pages 2554–2558, April 1982.

[55] J.J. Hopfield. "Neurons with Graded Response Have Collective Computational Properties Like those of Two-state Neurons". In *Proceedings of the Natural Academy of Science USA*, volume 81, pages 3088–3092, 1984.

[56] J.J. Hopfield and D.W. Tank. "Neural Computation of Decisions in Optimization Problems". *Biological Cybernetics*, 52:141–152, 1985.

[57] J.J. Hopfield and D.W. Tank. "Computing with Neural Circuits: A Model". *Science*, 233:625–633, August 1986.

[58] B.K.P. Horn. *Robot Vision.* The MIT Press, Cambridge, Massachusetts, 1986.

[59] B.K.P. Horn and B.G Schunck. "Determining Optical Flow". *Artificial Intelligence*, 17:185–203, 1981.

[60] G. Humphreys and V. Bruce. *Visual Cognition.* Lawrence Erlbaum Associates Ltd, Hove, Sussex, UK, 1989.

[61] J. Illingworth and J. Kittler. "Survey of the Hough Transform". *Computer Vision, Graphics and Image Processing*, pages 87–116, 1988.

[62] G. Johansson. "Visual Perception of Biological Motion and a Model for its Analysis". *Perception and Psychophysics*, 14(2):201–211, 1973.

[63] G. Johansson. "Visual Motion Perception". *Scientific American*, pages 76–88, June 1975.

[64] G. Johansson. "Spatio-temporal Differentiation and Integration in Visual Motion Perception". *Psychological Research*, 14(38):379–393, 1976.

[65] M. Jordan. "Serial Order: A Parallel Distributed Processing Approach". In *Advances in Connectionist Theory.* Erlbaum, 1989.

[66] M. Kass, A. Witkin, and D. Terzopoulos. "Snakes: Active Contour Models". *International Journal of Computer Vision*, 1(4):133–144, 1987.

[67] M. Kass, A. Witkin, and D. Terzopoulos. "Snakes: Active Contour Models". In *First International Conference on Computer Vision, London, England.* IEEE, Piscataway,NJ, 1987.

[68] W.B. Kearney, J.K. Thompson and D.L. Boley. "An Error Analysis of Gradient-Based Methods for Optical Flow Estimation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 229–244, 1987.

[69] M. Kirby and L. Sirovich. "Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1), 1990.

[70] M. Kirby, F. Weisser, and G. Dangelmayr. "A Model Problem in the Representation of Digital Image Sequences". *Pattern Recognition*, 26(1):63–73, 1993.

[71] C. Koch, J. Hutchinson, J. Luo, and C. Mead. "Computing Motion Using Analog and Binary Resistive Networks". *Computer*, 21(3):52–63, March 1988.

[72] C. Koch, J. Marroquin, and A. Yuille. "Analog Neuronal Networks in Early Vision". *Proceedings of the National Academy of Science, USA*, 83:4263–4267, 1986.

[73] T. Kohonen. "Self-organized Formation of Topologically Correct Feature Maps". *Biological Cybernetics*, 43:59–69, 1985.

[74] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 1989.

[75] D. Koller, K. Daniilidis, T. Thorhallson, and H. Nagel. "Model-based Object Tracking in Traffic Scenes". In *European Conference on Computer Vision*, Genoa, Italy, 1992.

[76] D. Koller, N. Heinze, and H. Nagel. "Algorithmic Characterization of Vehicle Trajectories From Image Sequences From Motion Verbs". In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 90–95, Maoui, HI, USA, 1991. IEEE.

[77] L. Koss. "Automated Cytology and Histology: A Historical Perspective". *Analytical Quantitative Cytology and Histology*, 9(5):369–374, 1987.

[78] L. Kozlowski and J.E. Cutting. "Recognising the Sex of a Walker from a Dynamic Point-light Display". *Perception and Psychophysics*, 21(6):575–580, 1977.

[79] A. Lanitis, C. Taylor, and T. Cooles. "An Unified Approach to Coding and Interpreting Face Images". In *IEEE International Conference on Computer Vision*, pages 368–373, Cambridge, Massachusetts, 1995.

[80] Y. Le Cun. "Generalization and Network Design Strategies". Technical Report CRG-TR-89-4, 1989.

[81] Y. Le Cun, J.S. Boser, B. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. "Handwritten Digit Recognition with a Back-propagation Network". In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems II*, pages 396–404. San Mateo:Morgan Kaufmann, Denver 1989.

[82] R. Linsker. "Self-organization in a Perceptual Network". *Computer*, 21(3):105–117, March 1988.

[83] W.A. Little. "The Existence of Persistent States in the Brain". *Mathematical Biosciences*, 19:101–120, 1974.

[84] D.G. Lowe. "Fitting Parameterised Three-dimensional Models to Images". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5), 1991.

[85] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* W.H. Freeman & Co., San Francisco, California, 1982.

[86] J. Marroquin, S. Mitter, and T. Poggio. "Probabilistic Solution of Ill-posed Problems in Computational Vision". *Journal of the American Statistical Association*, 87(397):76–89, March 1987.

[87] G.A. Martin and M. Shah. "Lipreading Using Optical Flow". In *National Conference on Undergraduate Research*, USA, March 1992.

[88] K. Mase and A.P. Pentland. "Lip Reading: Automatic Visual Recognition of Spoken Words". Technical Report 117, MIT Media Lab Vision Science, 1989.

[89] W.S. McCulloch and W. Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[90] M.L. Minsky. *Computation: Finite and Infinite Machines.* Englewood Cliffs: Prentice-Hall, 1967.

[91] M.L. Minsky and S. Papert. *Perceptrons.* The MIT Press, Cambridge, Massachusetts, 1988. Expanded edition.

[92] A. Mitiche, Y.F. Wang, and J.K. Aggarwal. "Experiments in Computing Optical Flow with the Gradient-based, Multiconstraint Method". *Pattern Recognition*, 20(2):173–179, 1987.

[93] M.C. Mozer. "A Focused Back-propagation Algorithm for Temporal Pattern Recognition". *Complex Systems*, 3:349–381, 1989.

[94] M.C. Mozer. "Neural Net Architecture for Temporal Sequence Processing". In A. Weigend and N. Gershenfeld, editors, *Time Series Prediction: Predicting the Future and Understanding the Past.* Addison-Wesley Publishing, 1993.

[95] D.W. Murray and B.F. Buxton. "From an Image Sequence to a Recognised Polyhedral Object". In *Alvey Vision Conference*, Cambridge, England, September 1987.

[96] D.W. Murray and B.F. Buxton. *Experiments in the Machine Interpretation of Visual Motion.* MIT Press, Cambridge,Massachusetts,USA, 1990.

[97] D.W. Murray, A. Kashko, and H. Buxton. "A Parallel Approach to the Picture Restoration Algorithm of Geman and Geman on an SIMD Machine". *Image and Vision Computing*, 4(3):133–142, August 1986.

[98] J. von Neumann. "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components". In C.E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 43–98. Princeton University Press, 1956.

[99] J. von Neumann. *The Computer and the Brain.* Yale University Press, 1958.

[100] E. Oja. "A Simplified Neuron Model as a Principal Component Analyzer". *Journal of Mathematical Biology*, 15:267–273, 1982.

[101] W.R. Oliver. "Evaluation of Cytologic Deformation by Boundary Energy Minimisation". In Ortendahl and Llacser, editors, *Information Processing in Medical Imaging*, pages 355–368. Wiley-Liss, 1991.

[102] J. O'Rourke and N.I Badler. "Model-based Image Analysis of Human Motion Using Constraint Propagation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(4):522–536, 1980.

[103] A. Ovalle and C. Garbay. "KIDS: A Distributed Expert System for Biomedical Image Interpretation". In *Automation in Cytology:Advances in Systems and Techniques*, pages 419–433. Springer-Verlag, Heidelberg, 1987.

[104] B.A. Pearlmutter. "Learning State Space Trajectories in Recurrent Neural Networks". *Neural Computation*, 1(1):263–269, 1989.

[105] E.D. Petajan, B. Bischoff, D. Bodoff, and N.M. Brooke. "An Improved Automatic Lipreading System to Enhance Speech Recognition". In *"SIGCHI '88: Human Factors in Computing Systems"*, pages 19–25, October 1988.

[106] F.J. Pineda. "Generalisation of Back-propagation to Recurrent Neural Networks". *Physical Review Letters*, 59:2229–2232, 1987.

[107] J. Piper. "Towards a Knowledge-based Chromosome Analysis System". In J. Piper and C. Lunsteen, editors, *Automation in Cytology:Advances in Systems and Techniques*, pages 275–294. Springer-Verlag, Heidelberg, 1989.

[108] T. Poggio and C. Koch. "Ill-posed Problems in Early Vision: From Computational Theory to Analogue Networks". In *Proceedings of the Royal Society London B*, volume 226, pages 303–323, 1985.

[109] T. Poggio, V. Torre, and C. Koch. "Computational Vision and Regularization Theory". *Nature*, 317:314–319, 1985.

[110] R. Polana and R. Nelson. "Recognition of Motion from Temporal Texture". In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 129–134, Champaign, IL, June 15-18 1992.

[111] R. Polana and R. Nelson. "Detecting Activities". In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2–7, New York, NY, USA, June 15-17 1993.

[112] A. Psarrou and H. Buxton. "A Neural Network Approach to the Computation of Vision Algorithms". In *First IEE International Conference on Artificial Neural Networks*, London, U.K., 1989.

[113] A. Psarrou and H. Buxton. "Hybrid Architecture for Understanding Motion Sequences". *Neurocomputing*, 5:221–241, 1993.

[114] A. Psarrou and H. Buxton. "Motion Analysis Using Recurrent Neural Networks". In *International Conference in Artificial Neural Networks*, Sorrento, Italy, 1994.

[115] A. Psarrou, S. Gong, and H. Buxton. "Modelling Spatio-temporal Trajectories and Face Signatures on Partially Recurrent Neural Networks". In *IEEE International Conference in Neural Networks*, Perth, Australia, 1995.

[116] L.R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". *Proceedings of the IEEE*, 77(2):257–286, 1989.

[117] R. Rashid. "Towards a System for the Interpretation of Moving Light Displays". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, November 1980.

[118] M. Recce and P.C. Treleaven. "Parallel Architectures for Neural Computers". In v.d. C. Malsberg, editor, *Neural Networks*, pages 487–495. Springer-Verlag, Heidelberg, 1989.

[119] R. Rohwer. "The Moving Targets Training Algorithm". In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 558–565. San Mateo:Morgan Kaufmann, Denver, 1989.

[120] F. Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". *Psychological Review*, 65:386–408, 1958.

[121] R. Rowher and B. Forest. "Training Time-dependence in Neural Networks". In M. Caudill and C. Butler, editors, *IEEE First International Conference on Neural Networks*, volume vol II, pages 701–708, San Diego 1987.

[122] D. Rumelhart, G. Hinton, and R. Williams. "Learning Representations by Back-propagation Errors". *Nature*, 223, 1988.

[123] D. Rumelhart and J. McClelland. *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Volume 1: Foundations.* The MIT Press, Cambridge, Massachusetts, 1987.

[124] S. Runeson and G. Frykholm. "Kinematic Specifications of Dynamics as an Informational Basis For Person-and-action Perception: Expectation, Gender-recognition, and Deceptive Intention". *Journal of Experimental Psychology: General*, 112:585–615, 1983.

[125] R.J. Schalkoff. *Digital Image Processing.* Wiley, USA, 1989.

[126] L. Sirovich and M. Kirby. "Low-Dimensional Procedure for the Characterization of Human Faces". *Journal of The Optical Society of America*, 4(3), 1987.

[127] T. Stornetta, T. Hogg, and B.A. Huberman. "A Dynamical Approach to Temporal Pattern Processing". In D.Z. Anderson, editor, *Neural Information Pro-*

cessing Systems, pages 750–759. New York: American Institute of Physics, Denver, 1987.

[128] S. Sumi. "Upside-down Presentation of the Johansson Moving Light-spot Pattern". *Perception*, 13:283–286, 1984.

[129] C. Taylor and D. Cooper. "Shape Verification Using Belief Updating". In *British Machine Vision Conference*, Oxford, England, 1990.

[130] J.K. Tsotsos, J. Mylopoulos, H.D. Covvey, and S.W. Zucker. "A Framework for Visual Motion Understanding". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):563–573, November 1980.

[131] M. Turk and A. Pentland. "Eigenfaces for Recognition". *Journal of Cognitive Neuroscience*, 3(1), 1991.

[132] M. Turner, J. Austin, N.M. Allinson, and P. Thompson. "Chromosome Location and Feature Extraction Using Neural Networks". *Image and Vision Computing*, 11:39–46, 1993.

[133] S. Ullman. *The Interpretation of Structure from Motion.* The MIT Press, 1979.

[134] S. Ullman. "Analysis of Visual Motion by Biological and Computer Systems". In M. Fischler and O. Firschein, editors, *Readings in Computer Vision: Issues, Problems, Principles and Paradigms*, pages 132–144. Morgan Kaufmann, Los Altos, CA, USA, 1987.

[135] P. Vesely, M. Vesely, D. Koblic, and P.N. Dilly. "Computer-simulated Analysis of in Vitro Cell Migration". *Folia Biologica*, 33:177–190, 1987.

[136] P. Vesely, M. Vesely, D. Koblic, and P.N. Dilly. "Patterns of in Vitro Locomotor Behaviour Characterising Cells of Spontaneously Metastasizing Rat Sarcomas". In *Proceedings of 14th International Congress of Biochemistry*, Prague, Czechoslovakia, 1988.

[137] N. Walker. *Biomedical Image Interpretation.* PhD thesis, University of London, 1991.

[138] H. Wallach and D.N. O'Connell. "The Kinetic Depth Effect". *Journal of Experimental Psychology*, pages 205–217, 1953.

[139] E.A Wan. "Temporal Back-propagation: An Efficient Algorithm for Finite Impulse Response Neural Networks". In D.S. Touretzky, J.L. Elman, T.J. Sejnowski, and G.E Hinton, editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 131–140. San Mateo,CA: Morgan Kaufmann, 1990.

[140] E.A. Wan. "Time-series Prediction by Using a Connectionist Network with Internal Delay Lines". In A.D. Weigend and N.A. Gershebfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 195–217. Addison-Wesley, Reading, Massachusetts, 1994.

[141] J.A. Webb and J.K. Aggarwal. "Visually Interpreting the Motion of Objects in Space". *Computer*, 14(8):40–46, August 1981.

[142] M. Wertheimer. "Experimental Studies on the Seeing of Motion". In T. Shipley, editor, *Classics in Psychology.* New York: Philosophical Library, 1912. Reprinted (1961).

[143] B. Widrow and M.E Hoff. "Adaptive Switching Circuits". In *IRE WESCON Convention Record*, pages 96–104, 1960.

[144] B. Widrow and M.A. Lehr. "30 years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation". *Proceedings of the IEEE*, 78:1415–1442, 1990.

[145] R.J. Williams and D. Zipser. "A Learning Algorithm for Continually Running Fully Recurrent Networks". *Neural Computation*, 1:270–280, 1989.

[146] R.J. Williams and D. Zipser. "Experimental Analysis of the Real-time Recurrent Learning Algorithm". *Connection Science*, 1:87–111, 1989.

[147] S. Winograd and J.D. Cowan. *Reliable Computation in the Presence of Noise.* MIT Press, Cambridge, Massachusetts, USA, 1963.

[148] A.D. Worrall, R.F. Marslin, G.D. Sullivan, and K.D. Baker. "Model-based Tracking". In *Proceedings of the British Machine Vision Conference*, pages 310–318, Glasgow, Scotland, 1991.

[149] A.D. Worrall, G.D. Sullivan, and K.D. Baker. "Advances in Model-based Traffic Vision". In *Proceedings of the British Machine Vision Conference*, pages 559–568, Guildford, Surrey, 1993.

[150] Q. Wu. "Model-based Contour Analysis in Chromosome Segmentation System". In J. Piper and C. Lunsteen, editors, *Automation in Cytology:Advances in Systems and Techniques*, pages 217–229. Springer-Verlag, Heidelberg, 1989.

[151] J. Yamato, J. Ohya, and K. Ishii. "Recognising Human Action in Time-sequential Images Using Hidden Markov Models". In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 379–385, Champaign, IL., USA, June 15-18 1992.

[152] D. Zicha. Personal communication.

[153] D. Zicha and P. Vesely. "The Use of a Production System for Simulation Analysis of Tumor Cell Migration in Vitro - Development of a Specialised Control Strategy". In *Second European Conference on Artificial Intelligence in Medicine*, pages 269–275. Springer-Verlag, 1990.