

Motion Detection by Cooperation

– A case study of the time dimension in neural networks

Robert Pallbo

LU-CS-TR: 94-125



Department of Computer Science, Lund University

1994

Lund University Dept. of Computer Science,
Box 118, S-221 00 LUND, Sweden.
Email: robert.pallbo@dna.lth.se

© Robert Pallbo 1994

LUNFD6/(NFCS-3086)/1-71/(1994)

Abstract

This is a thesis for a Swedish licentiate degree. It consists of three papers along with a general introduction. The work described is the construction of a model of motion direction detection by the use of a neural network. The model is unusual in that it does not make use of any intermediate units. Instead, the input units are directly connected to the units that constitutes the detectors. The capacity of the network to detect motion is an emergent phenomenon which is dependent on the presence of spontaneous activity among the units. In addition to a presentation of the model and some results from simulations, it is discussed how the methodology can be used in applications other than motion detection.

PREFACE

This work is a thesis for a degree of “filosofie licentiat” which is a Swedish degree between a Master of Science and a PhD. The thesis consists of three papers along with a general introduction. The three papers serve as a summary of the research of the author. The introduction serves mainly to introduce the reader to the problems that are discussed in the papers.

The papers have the following origin:

- Robert Pallbo, “*Visual motion detection based on a cooperative neural network architecture*”, in Scandinavian conference on artificial intelligence – ’93, E. Sandewall and C. G. Jansson (ed.), pp. 193–201, IOS Press, 1993.
- Robert Pallbo, “*Motion Detection – A neural model and its implementation*”, LUCS Minor, 1, Lund University Cognitive Science, 1994.
- Robert Pallbo, “*Ontogenesis in neural networks*”, Lund University Cognitive Studies, 24, 1993.

ACKNOWLEDGEMENTS

I am grateful to the following people for their contribution of comments, references, and/or discussions: Eric Astor, Christian Balkenius, Steen Christensen, Paul Davidsson, Peter Gärdenfors, Henrik Gedenryd, Christer Johansson, Lars Kopp, Sievert Lindström, Lucia Vaina, and Simon Winter. I also thank my colleagues at the department of computer science and at the cognitive science group for providing a stimulating research environment.

This work has been supported by Lund University Sweden during January 1992–June 1993 and the Swedish Council for Research in the Humanities and Social Sciences during the period July 1993–February 1994.

CONTENTS

Part I Introduction	1
1 Background	3
2 Neural networks	3
2.1 Units	4
2.2 Networks	5
2.3 Learning	6
2.3.1 Supervised learning	6
2.3.2 Unsupervised learning	8
2.4 Distributed representation	9
2.5 Recurrent networks	11
2.6 Neural networks and the brain	11
3 Visual motion detection	12
3.1 The visual system	13
3.2 Models from neurophysiology	14
3.2.1 Complex cells	15
3.3 A one-step motion detector	16
4 Discussion	17
4.1 From re-cognition to cognition	19
4.2 Evolutionary systems and future research	20
A Simulation tools	23
A.1 Early simulators	23
A.2 Recent programs and tools	23
References	27
Part II Visual motion detection based on a cooperative neural network architecture	31
Part III Motion Detection – A neural model and its implementation	33
Part IV Ontogenesis in Neural Networks	35

Part I
Introduction

1 BACKGROUND

Even if this work is mainly about motion detection, as the title suggests, my personal interests are not so much directed to the problems of detecting motion *per se*. Rather, my interests have been focused on the *method*. Initially, I wanted to study how *time* could be handled by neural networks. A search through the literature revealed that the time dimension seemed in every case to be transformed in one way or another to the spatial domain. As I wanted to avoid such an approach, I tried to define a minimal problem in which time is an important aspect. That was how I came to start with motion detection models.

The model that I present here, does not transform time into a spatial problem. Time is represented by time. This turned out to be a natural choice. Further, there is no final output. The network operates in a continuum of input images. This feature was actually used in the design of the model. In addition, I also would like to stress that the activity in the network is not caused by the input – all motion detection that occurs are generated internally.

The work that has been done have been both theoretical and technical in nature. The theoretical part consists of the creation of the model and its definition. The technical part consists of the construction of computer programs able to simulate the model. Several simulators have been constructed throughout this work. Further, a lot of time has been spent to run these simulators with various inputs under various conditions.

Three papers are supplied to serve as a summary of the work done. The first paper outlines the general idea. At the time this paper was produced, only preliminary computer simulation had been made on artificially constructed stimuli.

The second paper, a technical report, describes how a more complete implementation was made. In these simulations, video recordings were used as input. Further, this paper also describes some experiments that were made on this implementation to study the effects of adjusting the parameters.

The last paper is a first attempt to generalize the approach beyond motion detection and perception. This work is only in its early stages and will be the subject of my research for a doctoral degree.

In addition to these papers, I will also try to introduce the reader to the concept of neural networks, and the problem of detecting motion in a visual scene. At the end of the discussion, I will discuss the methodology in more general frames.

2 NEURAL NETWORKS

A neural network can be understood as two interacting dynamical systems (Fig. 1) (Ivan Havel, personal communication). One is the activity patterns of the units. The other one is the connection strength configuration defining the architecture. The former undergoes fast changes – the activity pattern typically changes from one iteration¹ in the network to another. The latter is adapting much slower, only minor changes are made at each iteration.

The two systems interact and influence each other. The activity patterns affect the connection configuration, and the connection configuration in turn affects the activity patterns. The former interaction we call plasticity, the latter activity transferring. Below, we will first see how a unit

¹We will in this introduction only consider neural networks that operate in discrete time.

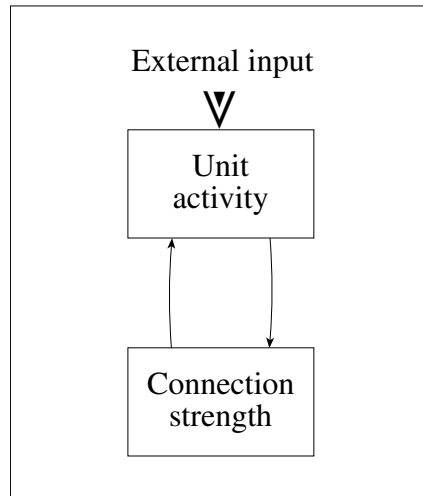


Figure 1: A neural network can be viewed as two interacting dynamical systems.

in the network operates and how they can be connected to form networks. Thereafter, various forms of learning will be described, some with the help of examples.

2.1 UNITS

A neural network consists of one or more connected units. The operation of the units is fundamental to the operation of the entire system. I will therefore give a brief description on how these units are implemented.

The basic model of the unit originates from 1958 – Rosenblatt’s units in the “perceptron”. The units he used are simple, but adaptive, devices. Each unit receives signals, (x_1, \dots, x_n) , from other units or from the environment. These signals are multiplied by weights, (w_1, \dots, w_n) , associated with the connections and the sum is calculated. If this sum exceeds a pre-specified threshold, θ , the unit generates a positive output, otherwise the output is zero (Fig. 2). We write:

$$y = f\left(\sum_{i=0}^n w_i x_i - \theta\right),$$

where

$$f(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases} .$$

These simple devices are able to perform categorization by dividing the input space by a hyper-plane. In Figure 3 this is shown for the case of two input signals.

To enable learning, the weights can be modified by means of the delta rule (Rumelhart et al., 1986a). The application of this rule requires that the desired output from each unit is known after each presentation. To adapt the weights, the following iteration formula is repeated for each connection:

$$w_i(t+1) = w_i(t) + \eta(d(t) - y(t)) \cdot x_i(t)$$

where $d(t)$ is the desired output, $y(t)$ is the actual output, and $0 < \eta < 1$ is a gain term. If $d(t) = y(t)$ there will be no changes of the weights. If $d(t) \neq y(t)$, the change will be $\eta x_i(t)$.

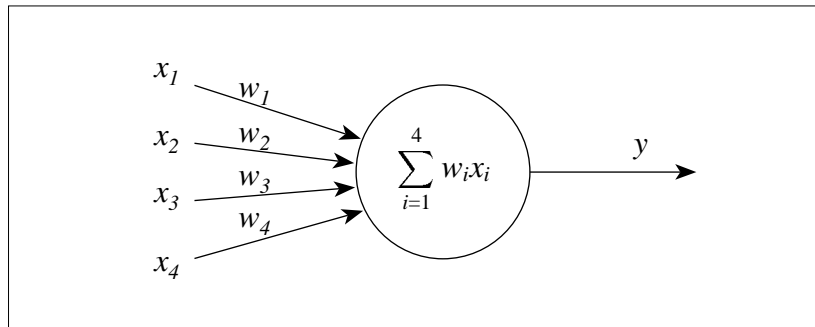


Figure 2: The unit integrates its weighted inputs and generates an output.

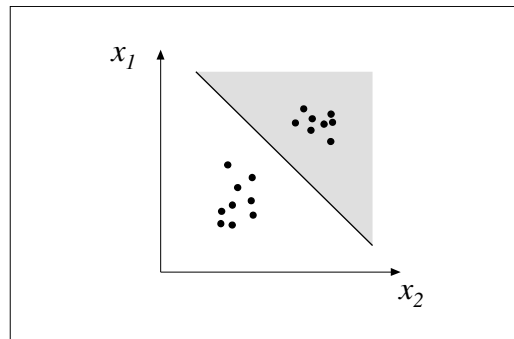


Figure 3: A single unit is able to divide the input space into two categories. All signals at the left of the boundary line will belong to one category and all signals at the right will belong to the other. Depending on which of these two categories the input belongs to, the unit will generate +1 or 0 as output.

2.2 NETWORKS

The number of units used in a neural network are frequently more than one. A main reason for this is that multi-unit networks allows more complex decision regions (Gibson and Cowan, 1990), and therefore more advanced categorizations.

A network is formed by connecting the output signals of the units as input signals of other units. In addition, external input signals can be used. Such input is called the *network*. In a similar way, some of the output signals of the units can be used as the output of the network. The units that are connected to the network's input is usually called *input units*. The units that constitute the output of the network is called *output units*. All other units are called *hidden units*. The connections that are present in the network is called the (*connection*) *architecture*, and the set of strengths associated with each of these connections the *connection strength configuration*.

It is possible to construct a neural network where all units are mutually interconnected. In fact, all neural networks could be considered to be fully connected if the unused connections are given a strength that is fixed to zero. However, the units in a neural network are usually organized in *layers*. One example of such an architecture is the popular feed-forward network (frequently used with the back-propagation learning rule). Figure 4 illustrates an example of a

feed-forward network architecture.

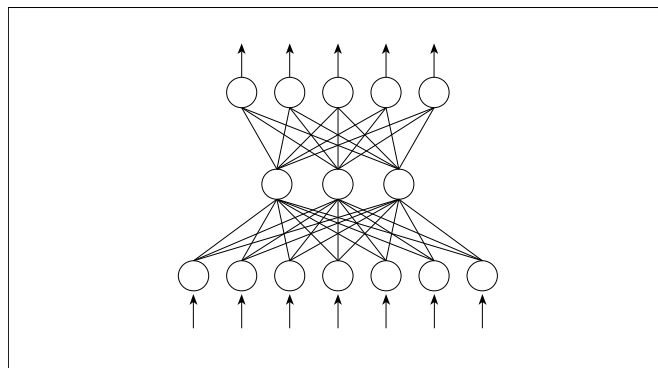


Figure 4: A feed-forward network with three layers.

In a feed-forward network the input signal is presented to the first layer of units (the bottom layer in Figure 4). The units in this layer integrate their weighted input signals and generates an output which in turn is propagated to the next layer. After a while, the activity has reached the final layer (the top layer in Figure 4), and the signals generated by these units will be taken as the final output of the computation of the network.

2.3 LEARNING

The way in which a neural network will respond to a certain input signal, depends on the weights associated with each connection in the network. If the weights are changed, the behaviour (i.e., the output) of the network will also change.

This basic property is the ground for adaption in a network. The learning algorithm of a neural network describes how the weights are to be modified. Usually, this occurs during a learning phase of the network. Before the network is trained during the learning phase, the weights in the network are given random values. Then, examples of input are presented to the network and the learning algorithm will use these examples, and optionally some additional information to gradually modify the weights. The number of examples that are presented to a network during the learning phase usually ranges from a few dozen to hundreds of thousands depending on the approach and the complexity of the task.

The learning algorithms of neural networks can be distinguished in two categories. One category is algorithms that use additional information, i.e., supervised learning, and another is algorithms that use no other information than the examples themselves, i.e., unsupervised learning. Below, a brief description of one example for each of these categories will be given. A full review is beyond the scope of this text, and the interested reader is directed to the large amount of literature that is available about this field.

2.3.1 Supervised learning

The most well-known supervised learning algorithm is back-propagation, which is also known as the generalized delta rule (e.g., Rumelhart et al., 1986b). It is associated with multi-layer feed-forward networks of the kind illustrated in Figure 4. Typically, the network consists of

three layers of units; one input layer, one hidden layer, and one output layer. The hidden layer is called “hidden” because it has no direct connections with neither the input signals, nor the output and is thus hidden from an external observer. If the network consists of four layers, there would be two hidden layers, and so on.

The additional information required by a supervised learning algorithm is the *desired* output. The algorithm compares the desired output with the actual output and modifies the weights in such a way that the actual output will be more in correspondance with the desired output if it is presented a second time.

This might sound fairly simple, but it is complicated by the fact that the desired output is known only for the output layer. What the desired “output” of the hidden layers (the “input” to the output layer) should be, the algorithm must find out itself. How this is done will be described below.

First, the weights of the input to the output units are adjusted:

$$w_{ij}(t + 1) = w_{ij}(t) + \eta \delta_j y_i$$

where η is a small gain parameter that determines the speed of the learning, and δ_j is an error parameter determined by:

$$\delta_j = f'_j(z)(d_j - y_j),$$

where d_j is the desired output and y_j is the actual output. The other connections, in the hidden layers, are adjusted in the same way but the error parameter for each unit is computed by:

$$\delta_j = f'_j(z) \sum_k \delta_k w_{jk}$$

where k is an index over the units in the layer above (after) the target unit j . In this way, the errors of the output units are recursively propagated backward in the network.

As the function $f(z)$ needs to be derivable, the threshold function used in section 2.1 will not do. An alternative is to use the sigmoid function (Fig. 5),

$$f(z) = \frac{1}{1 + e^{-z}},$$

which has the derivative,

$$\frac{\partial f}{\partial y_j} = y_j(1 - y_j), \text{ where } y_j = \sum_{i=0}^n w_{ij}x_i - \theta.$$

It has been shown (Rumelhart et al., 1986b) that the adaption of the weights, in the manner described above, will minimize the square of the error between the desired and the achieved output. We can think of the error for each pattern, p , as energy using:

$$E_p = \frac{1}{2} \sum_j (d_{pj} - y_{pj})^2.$$

The procedure will then minimize the error of the total population of patterns,

$$E = \sum_p E_p.$$

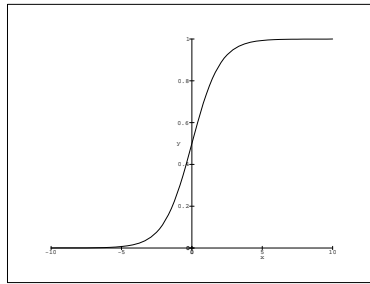


Figure 5: The sigmoid function.

A feed-forward network of this kind can map any function² from one finite dimensional space to another with desired accuracy (Hornik et al., 1989). Furthermore, using the back-propagation algorithm, it can *learn* such a mapping (White, 1990). This does not, however, say anything about in what degree the network has found any general structure in the data. There is always a risk that the network will become an “idiot savant”, and memorize only the data set used for learning. To avoid this, the number of units in the hidden layer is usually limited in number, so that the number of hidden units will be fewer than the number of input and output units. The effect is that the network then *has to* generalize in order to learn a set of pattern. There will simply not be enough units to memorize every input-output pair.

2.3.2 Unsupervised learning

When learning is not supervised, the system receives no information of what the desired output should be. Hence, the output produced by such networks have an arbitrary representation and must be interpreted by an observer. The network will use the input *per se* to control the learning process. This usually means that any structural relationships in the input patterns will be evident in the output, and, typically, it is structural information the designer desires.

Even if a network in itself is run without supervision, it would not be fair to say that the *usage* of such systems is unsupervised. In most cases, the network has to be carefully designed, and many versions evaluated, before the desired output is achieved (which is also the case in supervised learning). However, an important distinction from the case of supervised learning is that the desired result does not need to be *precisely* defined. The designer might be interested to see if there are any learnable structural relationships in the raw data.

A popular, and widely used, unsupervised network is the self-organizing map (SOM), or, as it is more known, the Kohonen network (Kohonen, 1989, 1990). This category of networks consists of one layer of topologically arranged units. Every unit in the network receives signals from each of the input sources.

In a Kohonen network, we want the similarities in the (usually) high dimensional input to be reflected as distances in the (usually) low dimensional topology. Similar categories should be represented by nearby units. This is achieved by associating a topological neighbourhood, $N_i(t)$, to each unit. At each presentation of input signals, a distance, d , between the weight vector,

²Any Borel measurable function, that is.

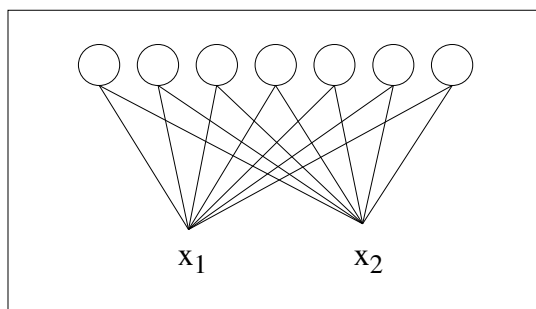


Figure 6: A Kohonen network with one-dimensional topology and two-dimensional input.

(w_{1j}, \dots, w_{mj}) , and the input vector, (x_1, \dots, x_m) , is determined for each of the N units, e.g.,

$$d_j = \sum_{i=0}^{N-1} (x_i(t) - w_{ij}(t))^2.$$

We then select the unit with the minimal distance and updates the weight vector of that unit *and* those in its neighbourhood (Fig. 7) by,

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(x_i(t) - w_{ij}(t)).$$

The weight vectors of the units outside the neighbourhood is not updated.

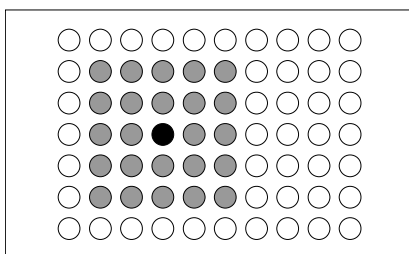


Figure 7: The neighbourhood, N_i (gray circles) of the unit i (black circle). Note that the neighbourhood function defines the topological dimension of the network. In this figure it is two-dimensional.

To make the learning converge, the gain factor, $\eta(t)$, should be a decreasing function of time. Likewise, the topological neighbourhood, $N_i(t)$, should also decrease in size. These parameter properties will make the initial learning coarser, and the final learning more precise. After a while, the gain factor will reach zero and no more learning will take place. The result must be evaluated and interpreted. If the result is satisfactory, the network can be employed in an application.

2.4 DISTRIBUTED REPRESENTATION

In the field of artificial intelligence, knowledge is traditionally represented by *symbols* (Newell and Simon, 1976). These are arbitrarily chosen and carry no information *per se* about the

represented entities. Relations between the entities must be captured by arranging the symbols in structures (e.g., semantic networks).

Even if knowledge could be represented the same way in neural networks, by letting one unit represent some specific knowledge, neural networks offer another alternative for representation. Knowledge is separated into knowledge atoms. Each unit will represent one such atom. More complex knowledge is represented by a set of knowledge atoms. This methodology is called *distributed representation*. The distinction between distributed and symbolic representation is fuzzy, however. The more abstract the knowledge atoms are, the more similar they will be to symbols.

We can think of the activity state (the pattern of active units) in a network as a string of bits that can be either on or off. A specific state can then be represented by a unique string. In this way, the knowledge representation in the neural network is very similar to the address of a symbol in a traditional artificial intelligence system. There is, however, one important respect – the string in the neural network system is simultaneously *both* the address *and* the content. This feature has some important advantages.

First, it makes the representation resistant to damage. If one, or a few, of the bits in the string are changed, the content is still similar to the original.

Second, “[distributed representations] automatically give rise to generalizations” (Hinton et al., 1986 p. 82). This is caused by the fact that two similar entities will have similar activity patterns. If something is learned about one of the entities, it will strengthen the connections between the active units representing that entity and the active units representing the new knowledge. As the two similar entities have a lot of units in common, the connections between the second entity and the new knowledge will increase in strength as well.

Finally, if we observe two different states in the neural network, representing two different entities or situations, we can say something about how they relate to each other. The more they overlap in their activity patterns, the more similar they are. This cannot be said about two symbols in AI without consulting the appended structure.

There are not only advantages with distributed representations, but also disadvantages. The most important is the conflicts that may arise when two entities are to be represented at the same time. Consider, for instance, that we wish to represent a blue car and a red bike simultaneously. We will then activate the pattern for blue, for car, for red and for bike. If we suppose (a bit unrealistic) that the activity pattern for car and bike is not in conflict and do not overlap, we still have a problem. There is no way to tell whether it is the car that is blue or if it is the bike. Such conflicts must somehow be avoided. There are several approaches to this problem (e.g., Balkenius, 1992). I will not discuss them here, but just mention that they all complicate the initially rather simple representation in one way or another.

A second problem, that distributed representation shares with symbol systems, is the grounding problem (see Harnad, 1990). The meaning or interpretation of a representation is attached by an external observer. To avoid this problem, the representations must somehow be grounded in the physical world. Some researchers have even argued that there should be no representations at all, only connections to the physical world (e.g., Brooks, 1990).

Neural networks appear to offer a perfect platform for this kind of physical grounding – all we have to do is to connect the input units to sensors and the output units to effectors. There are problems with such an approach, however. In most applications, the input signals to the network have been either carefully pre-processed sampled data, or artificially constructed data.

In addition, the output is usually presented at an abstract level that needs posterior interpretation. As long as data has to be prepared and adjusted, neural networks offer no real solution to the grounding problem.

2.5 RECURRENT NETWORKS

In the networks that we have considered so far, the generated activity pattern of the units depends on the history of the network through the influence of the architecture and the connection strengths. There exists, however, another class of networks that have activity patterns that more directly depend on the history of the network – recurrent networks. In these systems, the current *activity pattern* will influence the next such pattern. That is, the activity pattern will depend on both the architectural configuration as well as the previous activity pattern along any external input.

Recurrent networks can be subdivided into two categories. The first kind is what is sometimes called *auto-associators* (Kohonen, 1989), or *fully connected* networks. In these systems each unit is connected to all the other units and the network thus consists of one layer. Typically, the recurrent nature of the auto-associator is used to reach a stable state of the activity pattern. An initial pattern is presented to the network and the network will after a few iterations reach a stable state. The state that it will reach is the “closest” in some respect or the one that has least energy. Hence, auto-associators use time only as a mean to converge into a stable state in which it then resides.

The second kind of recurrent networks uses the old state of activity pattern to generate a *new* pattern. This makes it possible to generate a whole sequence of patterns. One way to construct such a network would be to use the feed-forward network described above, and use the output as part of the input signal. This kind of network is usually called Jordan networks (Jordan, 1986). Another possibility is to connect the output from the *hidden* units to the input. In this case, we will get an Elman network (Elman, 1990). An important aspect of the Elman network is that the reused signals from the hidden layer are not determined by the external user, as in the case of the Jordan network, but are determined by the network itself.

Any recurrent network of this second kind might learn and reproduce sequences. They can be used to forecast a sequence, or to find structure in the temporality of a set of sequences. They have one flaw in common, however – they are all supervised. This is not an objective drawback, however, but depends on the intention of the user.

The intention of the current work has been more toward constructing a network that can deal with time and still be self-organizing, or – at least – have the potential to be so. The present model of motion detection outlined in this thesis, is not self-organizing. In the final discussion of this introduction, I will however argue that the proposed model has the potential to be self-organizing.

2.6 NEURAL NETWORKS AND THE BRAIN

Sometimes it is stated that neural networks operate in a manner “similar to the brain”. This is however an overstatement in at least three respects. First, the units in neural networks are much simpler than the neurons in the brain. Second, the number of neurons in the brain is usually estimated to be between 10^{11} and 10^{12} . As a comparison, the motion detecting network presented

in this thesis consists of 300 000 units – a number considered very large in the field of neural networks. Finally, our knowledge about the processes in the brain is only fragmentary. To state that neural networks operate in a manner similar to the brain, suggests a fuller understanding of the brain than what we have to date.

3 VISUAL MOTION DETECTION

There are very good reasons for detecting motion in a visual scene. In the animal kingdom, anything that moves is worthy attention as a moving object is likely to be either a predator or a prey. It is no coincidence that some species can see nothing but moving objects. In artificial systems, motion detection is useful as well. The supervision of prohibited areas is only one example where motion detecting devices can play a crucial role. Another example where motion detection is fundamental, is visually guided mobile robots, even though they are not as common today.

At the first glance, the detection of motion in a visual scene appears to be a simple problem. Partly, I guess, because this process feels so effortless and immediate in our personal perception. But the simplicity is only illusionary, the task of detecting motion has many hidden pitfalls.

One problem that models of motion detection will encounter is the *aperture problem* (Marr, 1982). If motion is detected locally – seen only through a small aperture – there is no way to tell whether a moving profile is moving perpendicular, or in any other angle, to the orientation of the profile (Fig. 8a).

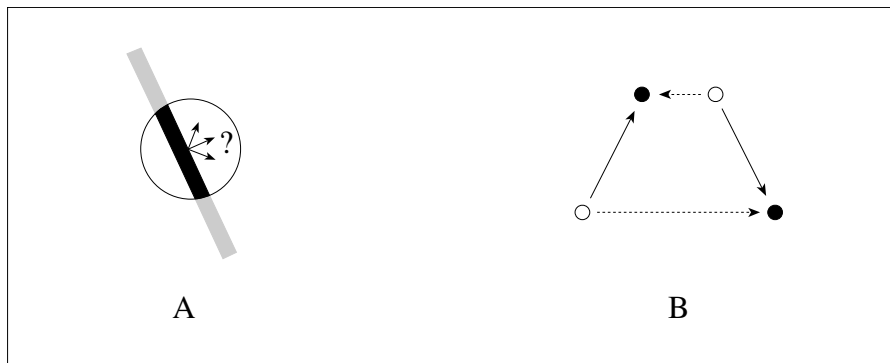


Figure 8: Problems with local motion detection. A: The aperture problem. There is no way to tell in which direction the motion is occurring if only a fraction of the scene is considered. B: The correspondence problem. The items at time t (empty circles) must be matched with the items at time $t + 1$ (filled circles). In the example shown, there are two ways of accomplishing a match. The items can be matched either by the solid lines, or by the dashed lines.

Another problem emerges if the visual scene is presented as a sequence of frames in discrete time intervals. Technically, there cannot be any motion in those frozen frames. However, we can reconstruct the motion that has occurred between successive frames. The result of such a procedure is known as *apparent motion* (Marr, 1982). The problem, then, is to tell which item in a frame that corresponds to the items in a successive frame – the *correspondence problem* (Fig. 8b). Usually, the items in a frame can match the items in the successive frame in many

different ways. The motion detecting algorithm has to select the most plausible matching – a task that is far from trivial.

The conclusion to draw from all this is that there exists no straight-forward approach to the problem of motion detection. A wide variety of models have been suggested in the literature, but as the current work is directed toward the use of neural networks, we will in the following consider only the models belonging to this field. Typically, neural networks that detect motion are found in the neurophysiological community. Below, the most popular such models will be presented. Thereafter, the proposed model will be introduced.

3.1 THE VISUAL SYSTEM

Before the description of any model, it is useful to know something about the basic organization of the visual system. This system varies between different species, but for primates it is essentially the same.

The visual system begins with the retina in the eye. Here, the photo-sensitive rods and cones are situated. As a consequence of the lens, each cell receives stimuli from only a fraction of the visual scene. This sub-field is called the *receptive field* of a cell. Whenever light is present in the receptive field of a rod or a cone, that cell will become active. The more light that reaches the cell, the more intense will the activation (the membrane potential) be. Some cells are only sensitive to a limited interval of the light spectrum, but as the current interest is not colour perception, this will be ignored in the following.

The image that is obtained by the photo-sensitive cells is processed by other cells, also situated in the retina. These are the ganglion cells and the intermediate bipolar, amacrine, and horizontal cells. The result of this process is such that a ganglion cell will respond best to either a bright dot surrounded by a dark area, or a dark dot surrounded by a bright area in its receptive field (Fig. 9). A homogenous brightness in this field will give no response at all. This effect is similar to what is obtained by the “mexican hat” function that is frequently used in artificial vision systems (refer to section 3 in part III).

The axons of the ganglion cells form the optic nerve. This nerve projects mainly³ to the lateral geniculate nucleus (LGN) in the thalamus. This site, which is organized in six layers, in turn projects to several other areas in the brain. Furthermore, it also *receives* projections from other sources than the optic nerve. What purpose all these projections have, is still unknown. However, the optic stimuli from the retina seem to be dominant in function (but not in number). For our purposes, it will suffice to note that the characteristics of the receptive field response are preserved in the LGN such that the cells that form its output, have a behaviour similar to the ganglion cells.

The primary visual cortex (Area 17), like the (neo)cortex in general, is organized into six distinct layers. The output from the LGN terminates mainly in layer 4. The cells located here, connect to cells in other layers of the primary visual cortex. Furthermore, the cells do also make lateral connections (Gilbert et al., 1990). In the primary visual cortex, the characteristics of the receptive field changes dramatically. The cells located here respond best to stimuli shaped like a bar in the receptive field. One category of cells called *simple cells* responds only when the bar is oriented in a certain angle. Another category is *complex cells* that respond only to moving bars. Many of the complex cells are only selective to one direction of movement. The simple

³The retinal output also projects to the pretectum and to the superior colliculus.

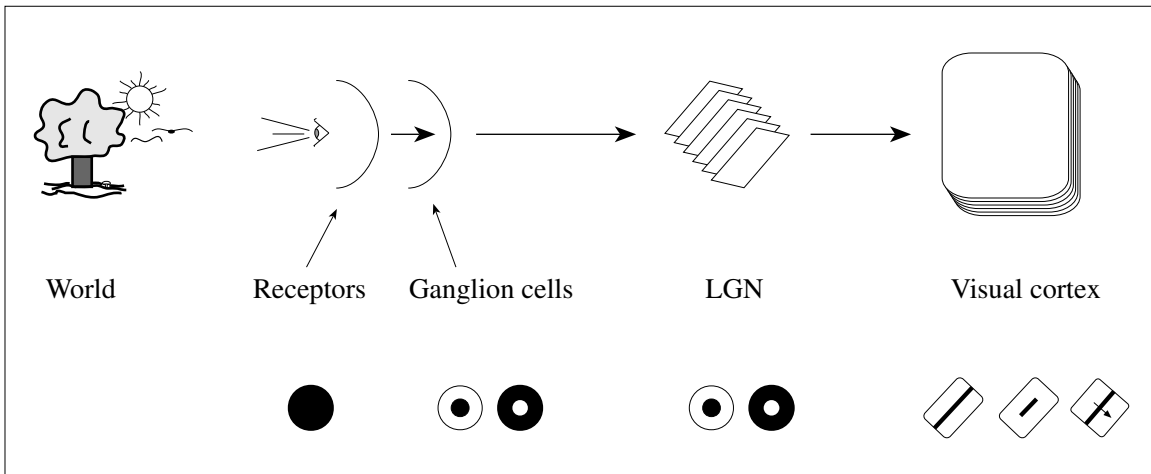


Figure 9: Outline of the primate visual system. The receptors in the retina give a stronger response the more light that enters the eye (represented by the filled circle). The ganglion cells, also situated in the retina, give best responses to dark dots on bright background or a bright dot on a dark background. The ganglion cells then project to the lateral geniculate nucleus (LGN). The selectivity of the ganglion cells is preserved. The cells in LGN then project (among other places) to the primary visual cortex. Here we will find cells that responds to lines of a certain direction (simple cells), cells responding to lines of a certain orientation and length (end stopped cells), cells that respond to motion (complex cells), and cells that respond to motion in a certain direction (directional selective complex cells). The primary visual cortex then projects to other areas of the brain, but that is beyond the current interest. From Pallbo, 1992.

cell was called simple as it was believed that their response could be obtained in one logical step from the geniculate projection (Hubel and Wiesel, 1962; Chapman, 1991). The complex cells were believed to require more complex curcuity and was thus called complex.

The visual system does not end here, in fact, we have only scratched the surface. The primary visual area project to secondary visual areas (area 18 and 19). These, in turn, contains a lot of different areas with a complex web of interconnections (Essen et al., 1992). But these details are not in the scope of the current interest, and, therefore, we will not consider them here.

3.2 MODELS FROM NEUROPHYSIOLOGY

The first model that was ever proposed for a motion detecting circuitry was not for complex cells, but for cells in the rabbit's retina (Barlow and Levick, 1965; cf. Grzywacz et al., 1991). In opposition to primates, directionally selective cells were found at this early stage in the rabbit's visual system. Barlow and Levick measured the response characteristics of these cells, but were unable to observe the underlying circuitry. As neural tissue is very obscure, we are still unable to extract specific circuitry of larger neuronal systems in the brain. However, Barlow and Levick constructed two hypothetical models that could explain the observations.

The two models can be thought of as two variations on the same theme. The idea is to use two versions of the input patterns. One that is obtained at time t , and the other at time $t + 1$. These two inputs then converge to the target cell which becomes directionally selective. In one arrangement of these inputs, the delayed signal is inhibiting the target cell whereas the un-delayed cell is exciting it (Fig 10a). If a stimulus appears first at the position of the delayed

input and thereafter at the un-delayed position, the delayed inhibitory connection will cancel an activation of the target cell. This direction of the stimulus is called the *null* direction and the target cells will respond only to stimuli moving in the opposite direction – the *preferred* direction.

In the other arrangement, both connections to the input signals are excitatory. If the target cell requires input from both sources in order to get activated, the cell will be directionally selective (Fig 10b). When a stimulus first appears in the delayed input position and thereafter at the position of the un-delayed input, the two input signals will reach the target cell simultaneously and, hence, activate the cell. If the stimulus moves in the other direction, the input signals will reach the target cell one at a time and thus not be sufficient for triggering an activity.

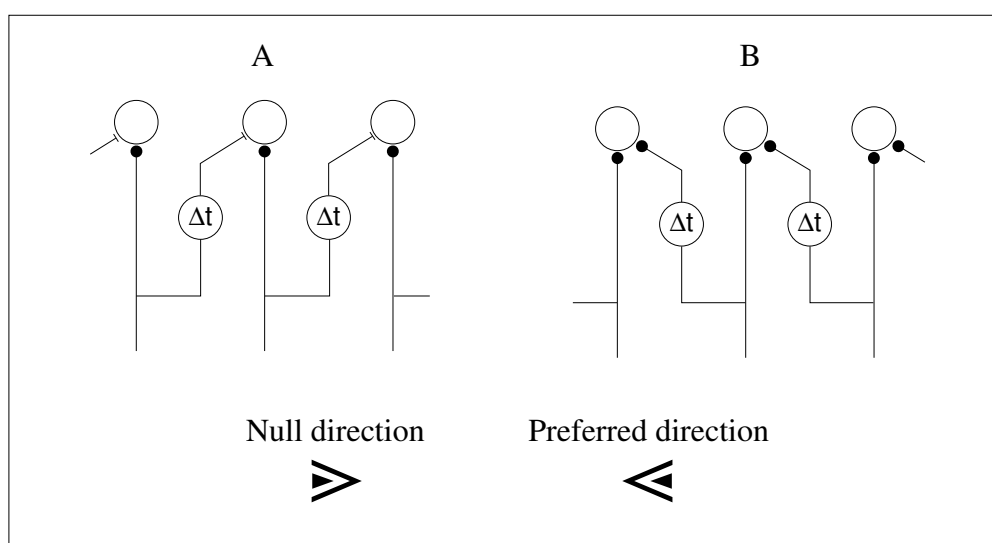


Figure 10: Two models proposed by Barlow and Levick 1965. The idea is to use two input signals to the target cell. These two inputs should be obtained at slightly different positions in the visual field. Furthermore, one of the signals should be delayed by the network. A: The delayed input is inhibiting the target cell and cancels activity caused by stimulus moving in the null direction. B: Both inputs are excitatory and the target cell requires input from both these sources in order to become activated. This will happen when stimuli move in the preferred direction.

3.2.1 Complex cells

The complex cells that are not directional selective can be modeled by a simple projection of many simple cells (Hubel and Wiesel, 1962). For directional selective cells though, a model is not as easily found. However, it is widely believed that the circuitry is of a type similar to the model proposed for the rabbit's retinal ganglion cells outlined above (Hubel, 1988), or at least that some interneurons are involved (e.g., Nagano and Fujiwara, 1979). This category of models is called *correlation type* motion detectors (Ullman, 1983; Borst and Egelhaaf, 1989).

Another type of models proposed as a model of direction detection is *gradient schemes*. In these models, the local temporal gradient of the light intensity is compared with the spatial gradient of the local pattern. By these measures, it is possible to derive not only the direction of

the motion, but also the speed. These models are, however, rather complex in their computations and have obtained limited support from empirical studies (Ullman, 1983).

3.3 A ONE-STEP MOTION DETECTOR

The absence of other competing ideas has made the correlation scheme put forward as the most probable explanation of motion detection in the primary visual cortex.

“The main reason for thinking that complex cells may be built up from center-surround cells, with a step in between, is the seeming necessity of doing the job in two logical steps” (p. 78, Hubel, 1988).

“[Directional selectivity] probably cannot be explained by any simple projection of simple cells onto complex cells, but seems to require inhibitory connections with time delays of the sort proposed for rabbit retinal ganglion cells by Barlow and Levick” (p. 518, Hubel, 1982).

However, there exist alternative ways of implementing a motion detector. What the traditional models have in common is that they all consider the initial phase of motion detection – how motion can be extracted from scratch using two successive frames. Here we will take another approach and consider the case when motion has already been detected and a new frame is presented. Instead of making a detailed and technical description of the model (which can be found in the enclosed papers), I will invite the reader to make a thought-experiment.

Think of yourself as being a unit amongst other in a motion detecting and direction selective network. Your task is to signal when motion occurs in a specific direction in your small visual field. What would you do? The task seems nearly impossible, especially if you start to worry about the aperture and correspondence problems. Well, I think you would cheat! Just think of the temptation. You are surrounded by units that have similar tasks. If a pattern moves toward you in the specific direction that you are obliged to detect, a lot of neighbouring units will signal this motion before it reaches you. All that you have to do is to watch them carefully. When they are signalling motion, then you simply await the pattern to appear in your visual field. As you already know that it is moving in the proper direction you can, with confidence, signal an occurrence of motion and hope that no one will discover your false play.

A probable objection to this is that your trick will work fine as it is only one unit that is cheating and *one* unit will not have any considerable impact on the network’s computation. Anyhow, I will, a bit contra-intuitive, state that we can construct a motion detecting network where *all* the units are cheaters. After all, the arrangement will work *as long as an initial detection is put into the system*. A pattern that is once detected, will be propagated by the cheating units until it disappears or halts. If the patterns halts, the units do get the information from their neighbours, but the pattern will never appear in their visual field, and, consequently, they will not signal.

The remaining problem of initial detection can be ignored by the use of a simple trick. If we at every step randomly select a small percentage of the units to signal irrespective of any motion, they will falsely inform their neighbours that a moving pattern is approaching. What will happen is that *if* a pattern really is approaching, the cheaters will propagate the information and a detection of the moving pattern will take form in the network. If the falsely induced signal

is *not* in correspondence with any moving pattern, no pattern will appear in the cheaters' visual fields and hence, the "detection" will not be propagated any further and disappear.

The idea of using false signals, or *spontaneous activity*, was inspired by a similar phenomenon in biological systems. We find spontaneous activity in most neural tissues, that is, activity without any obvious cause (Evarts, 1964; Burns et al., 1976; see also Freeman and Skarda, 1990). Furthermore, the spontaneous activity is especially evident in the 5:th layer of the primary visual cortex where direction selective complex cells are situated (Heggelund, 1981; Hubel, 1982). However, many researchers in the field think of spontaneous activity as not contributing to the processes in the brain:

"Through dissections suggested by his anatomist colleagues, or with the aid of anesthetic agents, the physiologist has minimized the intrusion of so-called 'spontaneous' electrical activity, in order that he may better study single-cell discharges evoked by central or peripheral stimuli, or the characteristics of evoked potentials elicited on an essentially silent electrical background. It is inherent in such an approach that so-called spontaneous activity is viewed as noncontributory in processes of information transaction, or that it contribute in only minor or secondary ways." (Adey, 1970, p. 224)

Even if the above discussion summarises the underlying idea of the motion detecting system presented in this work, the final system is a bit more complex. However, the model is still very simple in comparison to models proposed by other researchers. There are no intermediate units and the computation carried out by the individual units is indeed very simple. Still, its performance is remarkable good, especially when the input is scattered with noise. For further details, see the technical report in part III.

What regards the aperture and the correspondence problem, they are partly avoided by this arrangement. A moving pattern is assumed to persist in moving in the same direction. Further, a pattern can be reported to move in more than one direction simultaneously, one of which is the actual direction. This is an effect of that each direction has it own set of indicating units. To sharpen the detection of directions, some inhibitory connections between the sets of units detecting different directions were introduced. Again, a more detailed picture can be obtained from the technical report.

The correspondence problem is more easily avoided. As the previous motion direction is known to the system, it only has to match these items with the new input frame. The network also knows in what direction the pattern to match is expected to appear. If a pattern is moving too fast, however, the network will be unable to propagate a detection. This limitation is discussed in the technical report.

4 DISCUSSION

The neural network that constitutes the model of motion detection outlined above, exhibits some interesting properties. First of all, the detections that take form in the system originate from the internal spontaneous activity. The external input is no direct causation of the network's activity, but contributes only indirectly by supporting the internally generated percepts that are in phase with the environment. We can say that the system is in *resonance* with the environment

(Shepard, 1984) or that the perception is *self-created* (cf. Gedenryd, 1993), a feature unusual in neural network – especially at such a basic level. In most other approaches, the input causes (creates) the perception.

A second interesting property of the network is that not only is the representation distributed, but also the *computation*. The task of computing the motion detection is not localized to any unit at any level (cf. Douglas and Martin, 1991). It is distributed amongst the units of the network. Furthermore, there exists no *description* in the network of how motion detection shall be computed – it is instead an *emergent* property of the network.

A feature *absent* in the network is the influence of the unit activity on the underlying architecture (Fig 11). In the simulations, all connections and strengths were pre-set from start. This arrangement puts all the burden of design on the creator of the system. And, indeed, to bring forth a functional network with all connection strengths in harmony was a difficult task – especially since the system involves circular dependences (see the appended technical report).

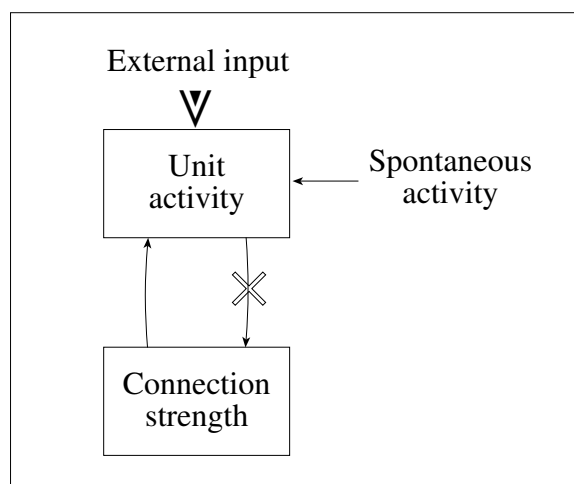


Figure 11: The network constituting the motion detecting model described in this work lacks the influence of the unit activity on the connection strengths. An added feature, though, is the spontaneous activity that is operating directly on the unit activity.

To reveal the cognitive burden from the designer, it would be desirable to make the network self-organizing. This could be realised by some simple adaption of the connection strengths in a way similar to what was suggested by Hebb (1949). The basic idea in his proposal is that whenever two units are active simultaneously, their connection will be strengthened. In addition, some way to decrease the strengths of the connections is also desirable. A way to achieve this, is to let the strengths slowly decay as a function of time. Anyhow, with self-organizing capabilities, the designer only has to create a crude system and leave the details to the network itself to arrange. What is interesting to note is that a system along the line of the proposed approach, would constitute a self-organizing neural network dealing with temporal sequences.

The main reason that self-organization was not studied within the frame of this work was the size of the network. The final system that constitutes the model consists of around 300 000 units and more than 6 million connections. If self-organization was to be introduced in this

system, it would considerably increase the time consumed for each iteration in the network (as each connection would require a more complex calculation). Further, the number of iterations required for the connection strength configuration to converge would be very large. If one considers that a simulation of 100 input frames takes about one hour on the available equipment⁴, a simulation of the same system with self-organizing abilities would be unrealistic. Anyhow, it can be interesting to speculate about such systems and how they can be applied to tasks other than motion detection.

4.1 FROM RE-COGNITION TO COGNITION

Let us first consider what the model presented above would be like if it was given self-organizing abilities. As for self-organizing networks in general, just any arbitrary architecture will not do. Some tendency in the connections must be present. In the actual model, the initial architecture should be such that there is a projection from the “input units” to the motion detection units. Furthermore, this projection should preserve the topology from the input. In addition, the motion detection units should be locally connected by excitatory and inhibitory connections.

When an input to such a system is presented, some activity will already be present due to the spontaneous activity. This activity will then propagate through the lateral connections with support from the input pattern. At first, the selectivity will be obscure, but (as each unit trusts its neighbours) the units within a specific neighbourhood will converge to signal the same event. We can expect several such neighbourhoods to emerge that will signal different events. Such an organization is in fact found in the primary visual cortex where such neighbourhoods are called *columns* (Hubel and Wiesel, 1963).

To generalize the approach is not as simple as just exchanging the input signal to some other modality. Most probably, the architecture of the connections and their associated strengths must also be modified. What these modifications should be, depends on which modality that is connected, and what we expect the network to do. However, local lateral connection and topological projections between areas are probably general principles to be employed.

In addition to generalize the system to use more external sources, we can also enrich the system with internal modules. These modules will not be connected to external sources, but will receive their input from other subsystems. For instance, the motion detection network could project its signals to another system dealing with saccades⁵ or some other aspects of vision.

The greatest advantage of such a multi-module approach, is probably that the choice of representations is totally determined by the system itself. There need not be a description of the interface between two modules. How the representations are made by one subsystem will simply be learned by the other systems that are connected to it. The alternative (to use fixed interfaces) is a hazardous approach. There will always be a risk that certain information is lost during the conversion to the interface because the designer did not anticipate a certain need of communication (you cannot expect the unexpected). For this reason, it is beneficial to keep the influence from external sources to a minimum. There is a price to pay for giving a system too much freedom, however – it considerably increases the effort needed in order to interpret what the system is doing and how well it is performing.

⁴A 32 MB SUN Sparcstation 2.

⁵A saccade is a quick movement of the eye.

Another interesting aspect of this kind of neural networks is that the learning could be described as an evolutionary process. This was the idea of the last of the enclosed papers, and I will end this introduction by trying to make this idea a little bit clearer.

4.2 EVOLUTIONARY SYSTEMS AND FUTURE RESEARCH

Plasticity in neural networks does not necessarily have to be associated with *learning*. Even if “[a]lmost every neural network model that has appeared to date is instructive.” (Anderson et al., 1990, p. 296), a few research groups have lately adopted the idea to model the plasticity as an evolutionary process (e.g., Changeux and Danchin, 1976; Changeux, 1983; Changeux et al., 1984; Edelman, 1979, 1987; Calvin, 1987, 1988). The idea to use selection instead of learning can be found as early as 1967 by Jerne, he wrote:

“Looking back into the history of biology, it appears that wherever a phenomenon resembles learning, an instructive theory was first proposed to account for the underlying mechanisms. In every case, this was later replaced by a selective theory.” (Jerne, 1967, p. 204).

He mentions the evolution of the species, the resistance of bacteria to antibacterial agents, and the antibody formation in the immune system as examples. Jerne then continues:

“It thus remains to be asked if learning by the central nervous system might not also be a selective process; i.e., perhaps learning is not learning either” (ibid).

Before I go into the detail of how a general neural network of the kind suggested in this work can be understood as an evolutionary model, I will try to clarify what I mean by evolution. For many people, evolution is *the Evolution*, i.e., the evolution of the species. But the principles of its operation can be adopted by other systems as well (Balkenius and Pallbo, 1994).

First of all, evolution operates on a population of some kind. We will label the current population in the system P , an individual in this set p and the set of all possible populations as Π . Evolution can then be understood as a function $Ph : \Pi \rightarrow \Pi$ (from Phylogenesis), i.e., a transformation of a population into a new one (Fig. 12). We can write $P' = Ph(P)$ where P' is the new population generated. When the phylogenetic function is recursively applied to a population, we will get an evolutionary process.

What characterizes an evolutionary system, in my view, is that a new population is generated in relation to the current one, which should be contrasted with a random generation of populations. The current population in the system is thus dependent on the history of the development. We can say that the evolution in these systems is *accumulative*. This feature is achieved by dividing the function Ph in two pieces – a function of *selection*, $S : \Pi \rightarrow \Pi$, and a function of *variation*, $V : \Pi \rightarrow \Pi$ (Balkenius and Pallbo, 1994).

The selection mechanism selects a subset, $P_S = S(P)$, from the current population. This subset is the set of individuals that will be considered and used in the creation of the new population. The new population is generated by the variation mechanism, $P' = V(P_S)$. It is important that this mechanism have some degree of uncertainty so that it can produce a new population that is different from the previous one. Otherwise, there would be no evolution of the population and hence the process would not be an instance of evolution.

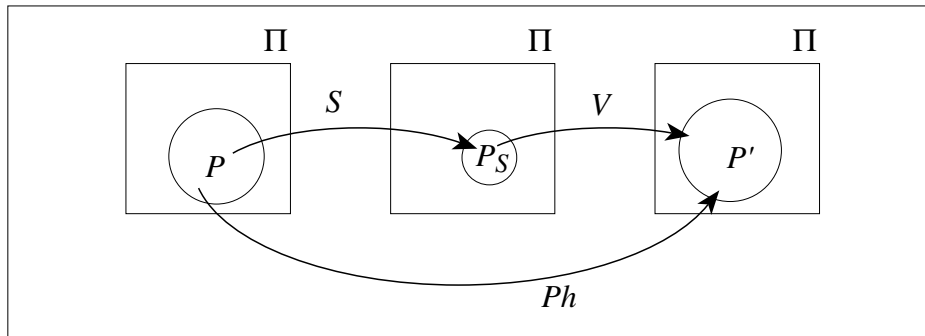


Figure 12: Evolution is a process of creating a new, or changing an old, population under considerations of the current one. The process can be divided into two pieces – one mechanism of selection and one mechanism of variation.

A requirement that raise complications in the design of an evolutionary system, is that the mechanisms of selection and variation must not be associated with a specific (fixed) population. They cannot be constructed in order to direct the evolution toward a fixed goal, but must be dependent on the current population itself. That is, the mechanisms of variation and selection determine the transformation of the current population, but the current population will also determine these mechanisms. This circular dependency is, in my view, the core of all evolutionary systems.

Given such requirements, the task of creating an evolutionary neural network appears impossible. But I would like to remind the reader that we already have a circular dependency present in a neural network – the one between the activity patterns and the configuration of the architecture. What remains to be done, is to equip the system with generative capabilities. But first, we need to identify the evolutionary components of the system.

In the class of neural network that have been under consideration, the population that we want to evolve consists of *representations*. In the following we will distinguish between *passive* and *active* representations. The active representations are those that are currently forming the activity pattern of the units. The passive representations are those patterns that the network has, as an effect of experience, the potential to activate. These representations are thus preserved by the configuration of the connection strenghts. For these reasons, we will name this category of networks *structure supporting networks* (SSN) to emphasize that we are interested in the (representational) structures that reside in the network and that this structure is distinct from the physical structure of the connection architecture.

The selected representations in structure supporting networks are the *active* representations. Upon these patterns, spontaneous activity is added to constitute the mechanism of variation. If the extra activity is “in phase” (i.e., not in conflict) with the current active and passive representations, as well as with the perception (input), then this added activity will remain in the system (i.e., be selected) and become a part of the current active representations. Spontaneous activity not in phase with the rest of the system will quickly vanish and will not influence the connection configuration that preseserves the passive representations. However, the spontaneous activity that *is* selected, will remain in activity long enough to influence the passive knowledge. Compare with the motion detection network, where the spontaneous activities that are in phase

with the actual motion are preserved.

The activity patterns that will be selected are dependent on the current degree of evolvement of the system, that is, its current population of passive representations. Therefore, the selection mechanism will be dependent on the population it selects - a requirement of evolutionary systems that was mentioned above. The same yields for the mechanism of variation. The influence that the spontaneous activity will have on the active representation is dependent on the active representation itself. The influence of the added activity will rely on how the activated units are relating to the rest of the population. Furthermore, the *complexity* of this influence is dependent on the complexity of the relations between the current population of representations.

To date, the study of the generalization of the method employed in the network of motion detection is only at its early stages. The future research of the author will be directed to explore the potentials of such systems and what implications it has for cognitive science when the brain is viewed as a fast evolutionary system.

A SIMULATION TOOLS

Throughout the development of the current model of motion detection, a number of computer programs for various purposes have been constructed. Some are simulators of the model, or versions of it, and others are dealing with the processing of the input and the results.

A.1 EARLY SIMULATORS

The first program that was created in this project was a simple real-time simulator of a motion direction detector. Its purpose was to establish a platform to test and study the primal idea of cheating neurons. This model included only an excitatory connection with the input, along with excitatory lateral connections corresponding to the “cheating”. Furthermore, the simulation involved only detection in one direction. As stimuli, a vertical bar (adjustable in width) was used (Fig. 13).

The simulator was given a user interface that allows the user to change (during run-time) a lot of the parameters affecting the simulation. The stimulus bar could be made to move either right or left, or to become stationary. Also, the level of spontaneous activity could be adjusted. Furthermore, the connection strengths as well as some aspects of the architecture could easily be modified.

This experience from the simulations carried out by this device was very valuable in the design of the final model. Not least since the simulator allowed the model to be tested in real-time, which was not possible with the succedent simulator.

At this point, I also come to realize that the model could easily be generalized to the detection of line orientations in an image as well. To try this hypothesis, a second simulator was created by modifying the first (Fig. 14). In this simulator, two orientations of lines could be detected. The stimuli used in the experiment, could be placed in four different orientation by the operator.

During the initial experiments with the model of line orientation detection, it became evident that inhibitory connections between the two different detection matrices were required. Furthermore, temporal summation in the units⁶ turned out to significantly increase the acuity.

A.2 RECENT PROGRAMS AND TOOLS

As the initial simulations of the model were encouraging, a more realistic evaluation of the model became desirable. For this purpose, simulations using video recordings were prepared.

First, a program capable of recording video sequences was created in collaboration with Christian Balkenius. The program was simple, but allowed the operator to view what he was recording (Fig. 15). Further, the need to view more than one video sequence at the time was anticipated. For this reason, a program capable of presenting up to twenty such sequences simultaneously was constructed (Fig. 16), again in collaboration with Christian Balkenius.

The next program developed was one that could filter the captured video sequences.⁷ The images obtained from the camera was (for various reasons) not suitable as input to the model. These filters used a video sequence as input and generated a new filtered video sequence as

⁶Temporal summation in principle mean that not all activation, or summed input, are lost between iterations.

⁷The purpose of the filter was to mimic the processes that occurs in the retina.

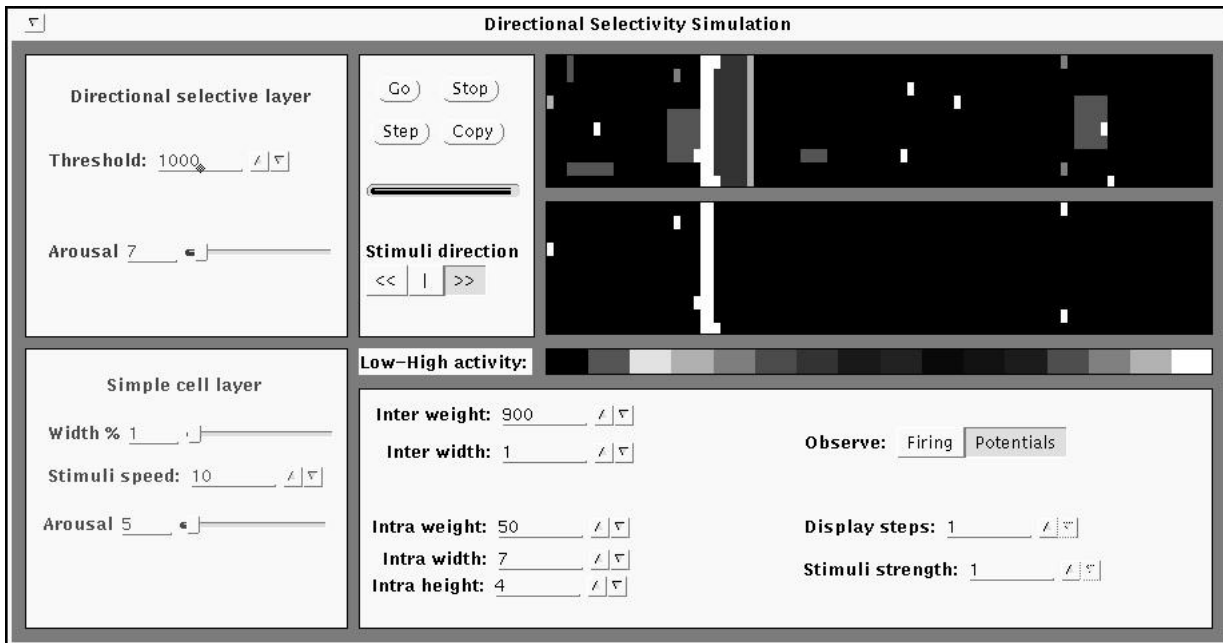


Figure 13: A simple simulator of direction detection.

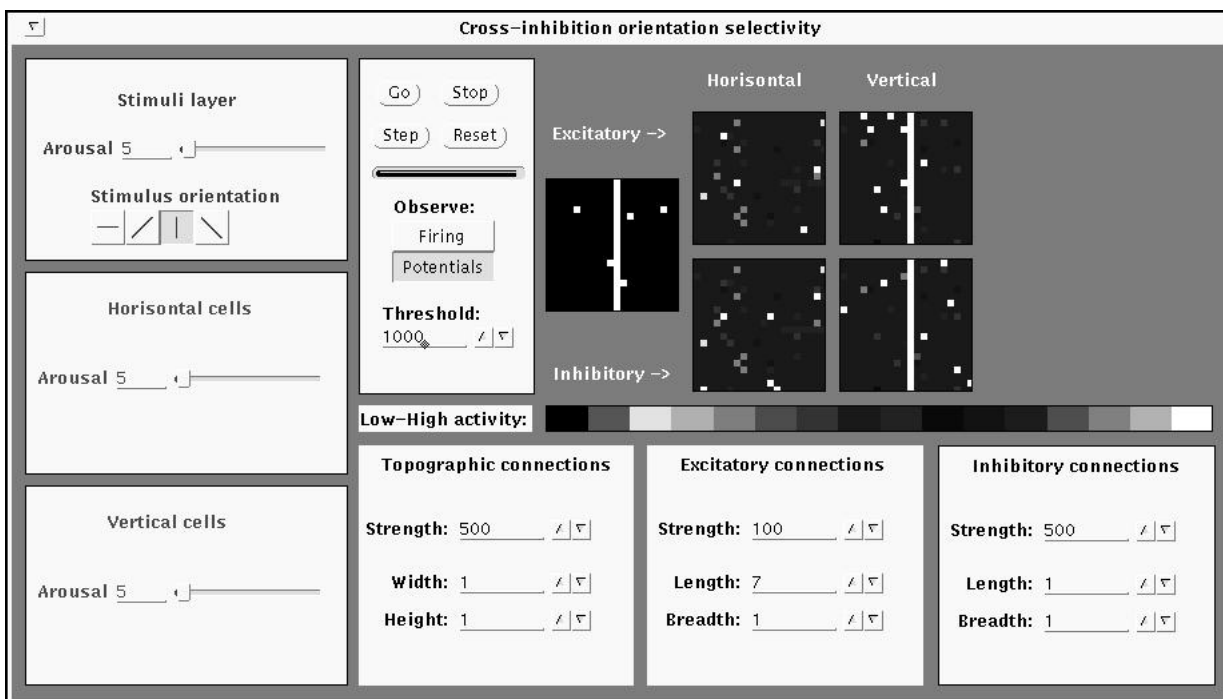


Figure 14: A simple simulator of line orientation detection.

output. This sequence could then be viewed and evaluated using the display program mentioned above.

Finally, the new simulator of the motion detection model was implemented. In opposition to the earlier simulators, it was not given a graphical interface. The main reason for this is the size of the network (it incorporated eight different directions and four different orientations), which caused the simulation to be too slow to be used interactively. Instead, the simulator produced a set of video sequences that could be viewed later using the graphical tools.



Figure 15: The interface of the recording program.

For practical matters, other tools were also created. These include one tool to invert the colour of the frames, a second tool to cut off the tail of video sequences to make them shorter, a third tool to restore corrupted headers of the video sequence files, a fourth tool to add noise to an image, and, finally, one tool that merges two video sequences into one.

All programs have been written in SUN C. The graphical interfaces have been made with the help of OpenWindow Developer's Guide 1.1. The most difficult thing to implement was the colour graphics which was not supported by the Developer's Guide. This was needed in order to display the gray-scale video recordings as well as the colour coded activity of the units in the early simulators. My experiences is, however, that the effort was not in vain. A clear graphical representation has considerably helped to visualize the simulation process. If I would have to do it all again, I would probably not do it much differently from the way I did.

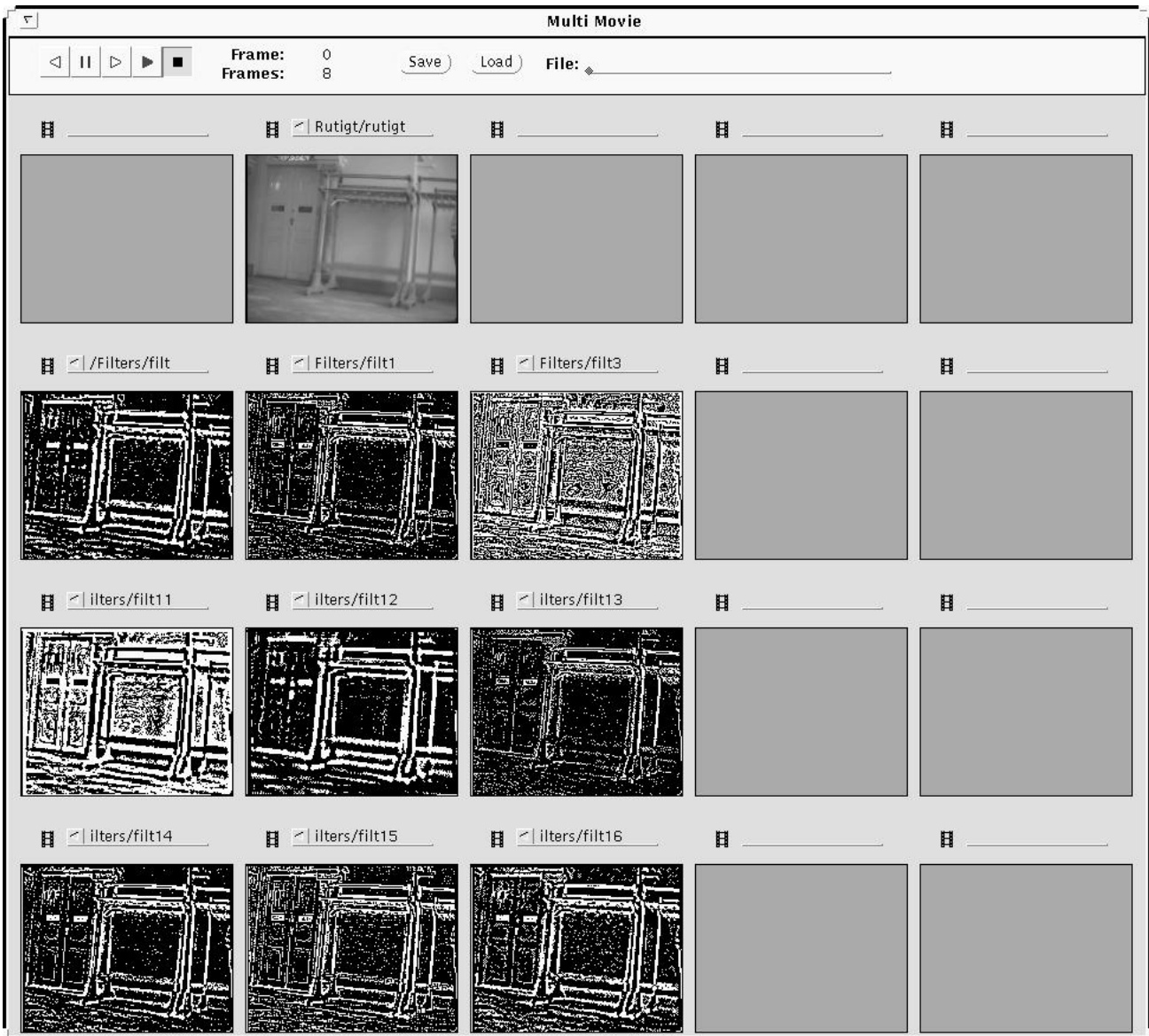


Figure 16: A task-designed program was used in order to present up to twenty video sequences simultaneously on the screen. Here various filters are evaluated.

REFERENCES

- Adey, W. R. (1970). Spontaneous electrical brain rhythms accompanying learned responses. In Schmitt, F. O., editor, *The Neurosciences: Second Study Program*, pages 224–243. Rockefeller University Press.
- Anderson, J. A., Pellionisz, A., & Rosenfeld, E., editors (1990). *Neurocomputing 2: Directions for Research*. The MIT Press.
- Anderson, J. A. & Rosenfeld, E., editors (1988). *Neurocomputing: Foundations of Research*. The MIT Press.
- Balkenius, C. (1992). Neural mechanisms for self-organization of emergent schemata, dynamical schema processing, and semantic constraint satisfaction. *Lund University Cognitive Studies*, 14.
- Balkenius, C. & Pallbo, R. (1994). Evolution and learning. In preparation.
- Barlow, H. B. & Levick, W. R. (1965). The mechanism of directionally selective units in rabbit's retina. *Journal of Physiology*, 178:477–504.
- Borst, A. & Egelhaaf, M. (1989). Principles of visual motion detection. *Trends in neuroscience*, 12:297–306.
- Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15. Also in (Maes, 1991).
- Burns, B. D., F.R.S., & Webb, A. C. (1976). The spontaneous activity of neurones in the cat's cerebral cortex. *Proceedings of the Royal Society London (Biology)*, 194:211–223.
- Calvin, W. (1987). The brain as a Darwin machine. *Nature*, 330:33–34.
- Calvin, W. H. (1988). A global brain theory. *Science*, 240(June 24):1802–1803.
- Changeux, J.-P. (1983). *Neuronal Man*. Oxford University Press. Translation of L'homme neuronal (1983).
- Changeux, J.-P. & Danchin, A. (1976). Selective stabilization of developing synapses as a mechanism for the specifications of neural networks. *Nature*, 264:705–712.
- Changeux, J.-P., Heidmann, T., & Patte, P. (1984). Learning by selection. In Marler, P. & Terrace, H. S., editors, *The Biology of Learning*, pages 115–133. Springer-Verlag.
- Chapman (1991). Relation of cortical cell orientation selectivity to alignment of receptive fields of the geniculocortical afferents that arborize within a single orientation column in ferret visual cortex. *Journal of Neuroscience*, 11(5):1347–1358.
- Douglas, R. J. & Martin, K. A. C. (1991). Opening the grey box. *Trends in Neuroscience*, 14:286–293.
- Edelman, G. M. (1979). Group selection and phasic reentrant signaling: A theory of higher brain function. In Schmitt, F. O. & Worden, F. G., editors, *The Neurosciences: Fourth Study Program*, pages 1115–1139. Rockefeller University Press.
- Edelman, G. M. (1987). *Neural Darwinism: The theory of Neuronal Group Selection*. Basic Books.
- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.
- Essen, D. C. V., Anderson, C. H., & Felleman, D. J. (1992). Information processing in the primate visual system: An integrated system perspective. *Science*, 255:419–422.
- Evarts, E. V. (1964). Temporal patterns of discharge of pyramidal tract neurons during sleep and waking in the monkey. *Journal of Neurophysiology*, 27:152–171.
- Freeman, W. J. & Skarda, C. A. (1990). Representations: Who needs them. In McGaugh, J.,

- Weinberger, N. M., & Lynch, G., editors, *Brain Organization and Memory: Cells, Systems and Circuits*, pages 375–380. Oxford University Press.
- Gedenryd, H. (1993). Active perception and the “resonance metaphor”. *Lund University Cognitive Studies*, 22.
- Gibson, G. J. & Cowan, C. F. N. (1990). On the decision regions of multilayer perceptrons. *Proceedings of the IEEE*, 78(10):1590–1594.
- Gilbert, C. D., Hirsch, J. A., & Wiesel, T. N. (1990). Lateral interactions in visual cortex. *Cold Spring Harbor Symposia on Quantitative Biology*, 50:663–677.
- Grzywacz, N. M., Amthor, F. R., & Mistler, L. A. (1991). Applicability of quadratic and threshold models to motion discrimination in the rabbit retina. *Biological Cybernetics*, 64:41–49.
- Harnad, S. (1990). The symbol grounding problem. *Physica D*, 42:335–346.
- Hebb, D. (1949). *Organization of Behaviour*. Wiley.
- Heggelund, P. (1981). Receptive field organization of complex cells in cat striate cortex. *Experimental Brain Research*, 42:99–107.
- Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In Rumelhart, D. & McClelland, J., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol.1: Foundations*, pages 77–109. MIT Press.
- Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366.
- Hubel, D. H. (1982). Exploration of the primary visual cortex, 1955–78. *Nature*, 299(7):515–524.
- Hubel, D. H. (1988). *Eye, brain, and vision*. Scientific American Library.
- Hubel, D. H. & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. *Journal of Physiology*, 160:106–154.
- Hubel, D. H. & Wiesel, T. N. (1963). Shape and arrangements of columns in cat’s striate cortex. *Journal of Physiology*, 165:559–568.
- Jerne, N. K. (1967). Antibodies and learning: Selection versus instruction. In Quarton, G. C., Melnechuk, T., & Schmitt, F. O., editors, *The Neurosciences: A Study Program*, pages 200–205. Rockefeller University Press.
- Jordan, M. I. (1986). Serial order: A parallel distributed processing approach. Technical Report 8604, University of California, Institute for Cognitive Science, San Diego.
- Kohonen, T. (1989). *Self-organization and associative memory*. Springer-Verlag.
- Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.
- Maes, P., editor (1991). *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*. MIT Press.
- Marr, D. (1982). *Vision*. W. H. Freeman and Company.
- Nagano, T. & Fujiwara, M. (1979). A neural network model for the development of direction selectivity in the visual cortex. *Biological Cybernetics*, 32:1–8.
- Newell, A. & Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126.
- Pallbo, R. (1992). Neuronal selectivity without intermediate cells. *Lund University Cognitive Studies*, 13.
- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408. Also in (Anderson and

- Rosenfeld, 1988).
- Rumelhart, D., Hinton, G., & McClelland, J. (1986a). A general framework for parallel distributed processing. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*. MIT Press.
- Rumelhart, D., Hinton, G., & Williams, R. (1986b). Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*. MIT Press.
- Shepard, R. N. (1984). Ecological constraints on internal representation: Resonant kinematics of perceiving, imagining, thinking, and dreaming. *Psychological Review*, 91(4):417–447.
- Ullman, S. (1983). The measurement of visual motion: Computational considerations and some neurophysiological implications. *Trends in Neuroscience*, 6:177–179.
- White, H. (1990). Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings. *Neural Networks*, 3:535–549.

Part II

**Visual motion detection based on a
cooperative neural network architecture**

Part III

**Motion Detection – A neural model and
its implementation**

Part IV

Ontogenesis in Neural Networks