9 Vision

9.1 Introduction

In this chapter we illustrate some applications of neural networks which deal with visual information processing. In the neural literature we find roughly two types of problems: the modelling of biological vision systems and the use of artificial neural networks for machine vision. We will focus on the latter.

The primary goal of machine vision is to obtain information about the environment by processing data from one or multiple two-dimensional arrays of intensity values ('images'), which are projections of this environment on the system. This information can be of different nature:

- recognition: the classification of the input data in one of a number of possible classes;
- geometric information about the environment, which is important for autonomous systems;
- compression of the image for storage and transmission.

Often a distinction is made between low level (or early) vision, intermediate level vision and high level vision. Typical low-level operations include image filtering, isolated feature detection and consistency calculations. At a higher level segmentation can be carried out, as well as the calculation of invariants. The high level vision modules organise and control the flow of information from these modules and combine this information with high level knowledge for analysis.

Computer vision already has a long tradition of research, and many algorithms for image processing and pattern recognition have been developed. There appear to be two computational paradigms that are easily adapted to massive parallelism: local calculations and neighbourhood functions. Calculations that are strictly localised to one area of an image are obviously easy to compute in parallel. Examples are filters and edge detectors in early vision. A cascade of these local calculations can be implemented in a feed-forward network.

The first section describes feed-forward networks for vision. Section 9.3 shows how back-propagation can be used for image compression. In the same section, it is shown that the PCA neuron is ideally suited for image compression. Finally, sections 9.4 and 9.5 describe the cognitron for optical character recognition, and relaxation networks for calculating depth from stereo images.

9.2 Feed-forward types of networks

The early feed-forward networks as the perceptron and the adaline were essentially designed to be be visual pattern classifiers. In principle a multi-layer feed-forward network is able to learn to classify all possible input patterns correctly, but an enormous amount of connections is needed (for the perceptron, Minsky showed that many problems can only be solved if each hidden unit is

connected to all inputs). The question is whether such systems can still be regarded as 'vision' systems. No use is made of the spatial relationships in the input patterns and the problem of classifying a set of 'real world' images is the same as the problem of classifying a set of artificial random dot patterns which are, according to Smeulders, no 'images.' For that reason, most successful neural vision applications combine self-organising techniques with a feed-forward architecture, such as for example the neocognitron (Fukushima, 1988), described in section 9.4. The neocognitron performs the mapping from input data to output data by a layered structure in which at each stage increasingly complex features are extracted. The lower layers extract local features such as a line at a particular orientation and the higher layers aim to extract more global features.

Also there is the problem of translation invariance: the system has to classify a pattern correctly independent of the location on the 'retina.' However, a standard feed-forward network considers an input pattern which is translated as a totally 'new' pattern. Several attempts have been described to overcome this problem, one of the more exotic ones by Widrow (Widrow, Winter, & Baxter, 1988) as a layered structure of adalines.

9.3 Self-organising networks for image compression

In image compression one wants to reduce the number of bits required to store or transmit an image. We can either require a perfect reconstruction of the original or we can accept a small deterioration of the image. The former is called a lossless coding and the latter a lossy coding. In this section we will consider lossy coding of images with neural networks.

The basic idea behind compression is that an n-dimensional stochastic vector $\mathbf{n}$, (part of) the image, is transformed into an m-dimensional stochastic vector

$$\mathbf{m} = \mathsf{T}\mathbf{n}. \tag{9.1}$$

After transmission or storage of this vector $\tilde{\mathbf{m}}$, a discrete version of $\mathbf{m}$, we can make a reconstruction of $\mathbf{n}$ by some sort of inverse transform $\tilde{\mathsf{T}}$ so that the reconstructed signal equals

$$\tilde{\mathbf{n}} = \tilde{\mathsf{T}}\tilde{\mathbf{n}}. \tag{9.2}$$

The error of the compression and reconstruction stage together can be given as

$$\epsilon = E\left[\|\mathbf{n} - \tilde{\mathbf{n}}\|\right]. \tag{9.3}$$

There is a trade-off between the dimensionality of $\mathbf{m}$ and the error ϵ. As one decreases the dimensionality of $\mathbf{m}$ the error increases and vice versa, i.e., a better compression leads to a higher deterioration of the image. The basic problem of compression is finding T and $\tilde{\mathsf{T}}$ such that the information in $\mathbf{m}$ is as compact as possible with acceptable error ϵ. The definition of acceptable depends on the application area.

The cautious reader has already concluded that dimension reduction is in itself not enough to obtain a compression of the data. The main importance is that some aspects of an image are more important for the reconstruction then others. For example, the mean grey level and generally the low frequency components of the image are very important, so we should code these features with high precision. Other, like high frequency components, are much less important so these can be coarse-coded. So, when we reduce the dimension of the data, we are actually trying to concentrate the information of the data in a few numbers (the low frequency components) which can be coded with precision, while throwing the rest away (the high frequency components). In this section we will consider coding an image of 256×256 pixels. It is a bit tedious to transform the whole image directly by the network. This requires a huge amount of neurons. Because the statistical description over parts of the image is supposed to be stationary, we can

break the image into 1024 blocks of size 8×8, which is large enough to entail a local statistical description and small enough to be managed. These blocks can then be coded separately, stored or transmitted, where after a reconstruction of the whole image can be made based on these coded 8×8 blocks.

9.3.1 Back-propagation

The process above can be interpreted as a 2-layer neural network. The inputs to the network are the 8×8 patters and the desired outputs are the same 8×8 patterns as presented on the input units. This type of network is called an auto-associator.

After training with a gradient search method, minimising ϵ, the weights between the first and second layer can be seen as the coding matrix T and between the second and third as the reconstruction matrix $\tilde{\mathsf{T}}$.

If the number of hidden units is smaller then the number of input (output) units, a compression is obtained, in other words we are trying to squeeze the information through a smaller channel namely the hidden layer.

This network has been used for the recognition of human faces by Cottrell (Cottrell, Munro, & Zipser, 1987). He uses an input and output layer of 64×64 units (!) on which he presented the whole face at once. The hidden layer, which consisted of 64 units, was classified with another *network* by means of a delta rule. Is this complex network invariant to translations in the input?

9.3.2 Linear networks

It is known from statistics that the optimal transform from an n-dimensional to an m-dimensional stochastic vector, optimal in the sense that ϵ contains the lowest energy possible, equals the concatenation of the first m eigenvectors of the correlation matrix R of N. So if $(\boldsymbol{e}_1, \boldsymbol{e}_2, .., \boldsymbol{e}_n)$ are the eigenvectors of R, ordered in decreasing corresponding eigenvalue, the transformation matrix is given as $\mathsf{T} = [\boldsymbol{e}_1 \boldsymbol{e}_2 \dots \boldsymbol{e}_2]^T$.

In section 6.3.1 a linear neuron with a normalised Hebbian learning rule was able to learn the eigenvectors of the correlation matrix of the input patterns. The definition of the optimal transform given above, suits exactly in the PCA network we have described.

So we end up with a $64 \times m \times 64$ network, where m is the desired number of hidden units which is coupled to the total error ϵ. Since the eigenvalues are ordered in decreasing values, which are the outputs of the hidden units, the hidden units are ordered in importance for the reconstruction.

Sanger (Sanger, 1989) used this implementation for image compression. The test image is shown in figure 9.1. It is 256×256 with 8 bits/pixel.

After training the image four times, thus generating 4×1024 learning patterns of size 8×8, the weights of the network converge into figure 9.2.

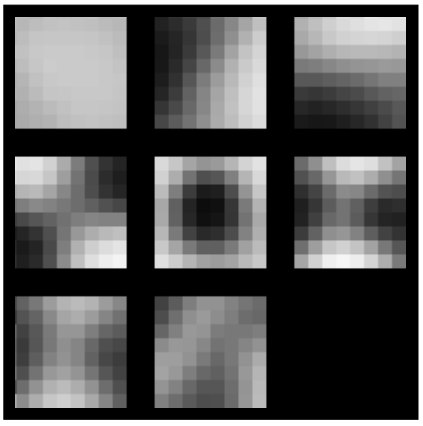
9.3.3 Principal components as features

If parts of the image are very characteristic for the scene, like corners, lines, shades etc., one speaks of features of the image. The extraction of features can make the image understanding task on a higher level much easer. If the image analysis is based on features it is very important that the features are tolerant of noise, distortion etc.

From an image compression viewpoint it would be smart to code these features with as little bits as possible, just because the definition of features was that they occur frequently in the image.

So one can ask oneself if the two described compression methods also extract features from the image. Indeed this is true and can most easily be seen in fig. 9.2. It might not be clear directly, but one can see that the weights are converged to:

Figure 9.1: Input image for the network. The image is divided into 8×8 blocks which are fed to the network.



ordering of the neurons:

neuron 0	neuron 1	neuron 2
neuron 3	neuron 4	neuron 5
neuron 6	neuron 7	(unused)

Figure 9.2: Weights of the PCA network. The final weights of the network trained on the test image. For each neuron, an 8×8 rectangle is shown, in which the grey level of each of the elements represents the value of the weight. Dark indicates a large weight, light a small weight.

- neuron 0: the mean grey level;
- neuron 1 and neuron 2: the first order gradients of the image;
- neuron 3 ... neuron 5: second orders derivates of the image.

The features extracted by the principal component network are the gradients of the image.

9.4 The cognitron and neocognitron

Yet another type of unsupervised learning is found in the *cognitron*, introduced by Fukushima as early as 1975 (Fukushima, 1975). This network, with primary applications in pattern recognition, was improved at a later stage to incorporate scale, rotation, and translation invariance resulting in the *neocognitron* (Fukushima, 1988), which we will not discuss here.

9.4.1 Description of the cells

Central in the cognitron is the type of neuron used. Whereas the Hebb synapse (unit k, say), which is used in the perceptron model, increases an incoming weight (w_{jk}) if and only if the

incoming signal (y_j) is high and a control input is high, the synapse introduced by Fukushima increases (the absolute value of) its weight ($|w_{jk}|$) only if it has positive input y_j and a maximum activation value $y_k = \max(y_{k_1}, y_{k_2}, \ldots, y_{k_n})$, where $k_1, k_2, \ldots, k_n$ are all 'neighbours' of k. Note that this learning scheme is competitive and unsupervised, and the same type of neuron has, at a later stage, been used in the competitive learning network (section 6.1) as well as in other unsupervised networks.

Fukushima distinguishes between *excitatory* inputs and *inhibitory* inputs. The output of an excitatory cell u is given by[1]

$$u(k) = \mathcal{F}\left[\frac{1+e}{1+h} - 1\right] = \mathcal{F}\left[\frac{e-h}{1+h}\right], \tag{9.4}$$

where e is the excitatory input from u-cells and h the inhibitory input from v-cells. The activation function is

$$\mathcal{F}(x) = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{9.5}$$

When the inhibitory input is small, i.e., $h \ll 1$, $u(k)$ can be approximated by $u(k) = e - h$, which agrees with the formula for a conventional linear threshold element (with a threshold of zero).

When both the excitatory and inhibitory inputs increase in proportion, i.e.,

$$e = \epsilon x, \qquad h = \eta x \tag{9.6}$$

(ϵ, η constants) and $\epsilon > \eta$, then eq. (9.4) can be transformed into

$$u(i) = \frac{(\epsilon - \eta)x}{1 + \eta x} = \frac{\epsilon - \eta}{2\eta}\Big(1 + \tanh(\tfrac{1}{2}\log \eta x)\Big) \tag{9.7}$$

i.e., a squashing function as in figure 2.2.

9.4.2 Structure of the cognitron

The basic structure of the cognitron is depicted in figure 9.3.

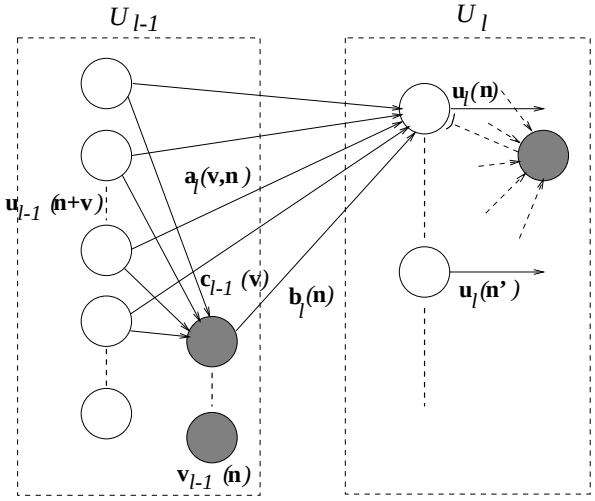


Figure 9.3: The basic structure of the cognitron.

The cognitron has a multi-layered structure. The l-th layer U_l consists of excitatory neurons $u_l(\mathbf{n})$ and inhibitory neurons $v_l(\mathbf{n})$, where $\mathbf{n} = (n_x, n_y)$ is a two-dimensional location of the cell.

[1] Here our notational system fails. We adhere to Fukushima's symbols.

A cell $u_l(\mathbf{n})$ receives inputs via modifiable connections $a_l(\boldsymbol{\nu}, \mathbf{n})$ from neurons $u_{l-1}(\mathbf{n}+\boldsymbol{\nu})$ and connections $b_l(\mathbf{n})$ from neurons $v_{l-1}(\mathbf{n})$, where $\boldsymbol{\nu}$ is in the connectable area (cf. area of attention) of the neuron. Furthermore, an inhibitory cell $v_{l-1}(\mathbf{n})$ receives inputs via fixed excitatory connections $c_{l-1}(\boldsymbol{\nu})$ from the neighbouring cells $u_{l-1}(\mathbf{n}+\boldsymbol{\nu})$, and yields an output equal to its weighted input:

$$v_{l-1}(\mathbf{n}) = \sum_{\boldsymbol{\nu}} c_{l-1}(\boldsymbol{\nu})\, u_{l-1}(\mathbf{n}+\boldsymbol{\nu}). \tag{9.8}$$

where $\sum_{\boldsymbol{\nu}} c_{l-1}(\boldsymbol{\nu}) = 1$ and are fixed.

It can be shown that the growing of the synapses (i.e., modification of the a and b weights) ensures that, if an excitatory neuron has a relatively large response, the excitatory synapses grow faster than the inhibitory synapses, and vice versa.

Receptive region

For each cell in the cascaded layers described above a connectable area must be established. A connection scheme as in figure 9.4 is used: a neuron in layer U_l connects to a small region in layer U_{l-1}.

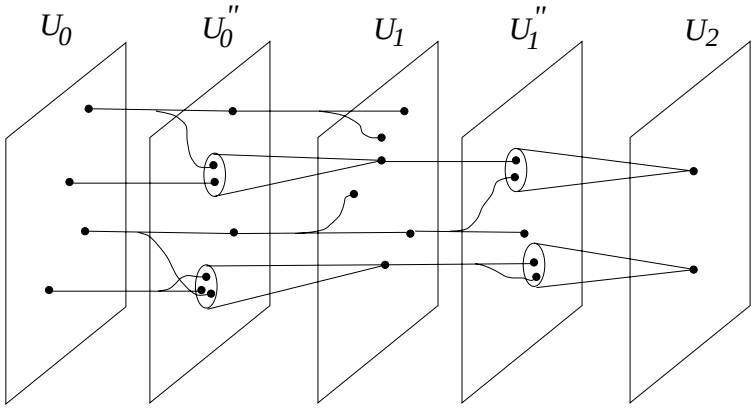


Figure 9.4: Cognitron receptive regions.

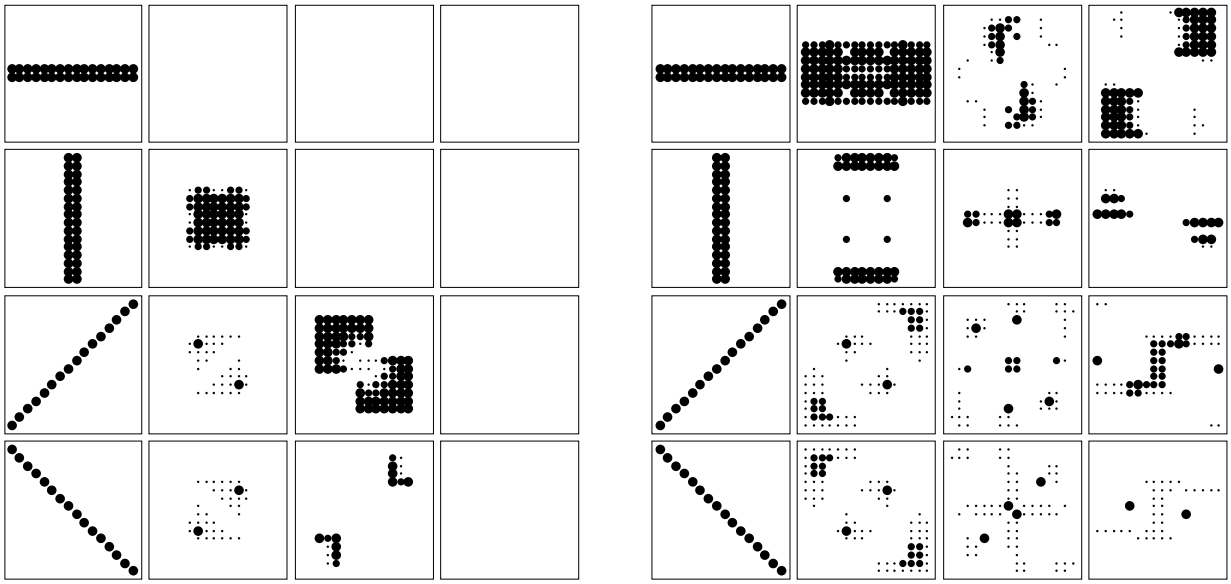
If the connection region of a neuron is constant in all layers, a too large number of layers is needed to cover the whole input layer. On the other hand, increasing the region in later layers results in so much overlap that the output neurons have near identical connectable areas and thus all react similarly. This again can be prevented by increasing the size of the vicinity area in which neurons compete, but then only one neuron in the output layer will react to some input stimulus. This is in contradiction with the behaviour of biological brains.

A solution is to distribute the connections probabilistically such that connections with a large deviation are less numerous.

9.4.3 Simulation results

In order to illustrate the working of the network, a simulation has been run with a four-layered network with 16×16 neurons in each layer. The network is trained with four learning patterns, consisting of a vertical, a horizontal, and two diagonal lines. Figure 9.5 shows the activation levels in the layers in the first two learning iterations.

After 20 learning iterations, the learning is halted and the activation values of the neurons in layer 4 are fed back to the input neurons; also, the maximum output neuron alone is fed back, and thus the input pattern is 'recognised' (see figure 9.6).



a. b.

Figure 9.5: Two learning iterations in the cognitron.
Four learning patterns (one in each row) are shown in iteration 1 (a.) and 2 (b.). Each column in a. and b. shows one layer in the network. The activation level of each neuron is shown by a circle. A large circle means a high activation. In the first iteration (a.), a structure is already developing in the second layer of the network. In the second iteration, the second layer can distinguish between the four patterns.

9.5 Relaxation types of networks

As demonstrated by the Hopfield network, a relaxation process in a connectionist network can provide a powerful mechanism for solving some difficult optimisation problems. Many vision problems can be considered as optimisation problems, and are potential candidates for an implementation in a Hopfield-like network. A few examples that are found in the literature will be mentioned here.

9.5.1 Depth from stereo

By observing a scene with two cameras one can retrieve depth information out of the images by finding the pairs of pixels in the images that belong to the same point of the scene. The calculation of the depth is relatively easy; finding the correspondences is the main problem. One solution is to find features such as corners and edges and match those, reducing the computational complexity of the matching. Marr (Marr, 1982) showed that the correspondence problem can be solved correctly when taking into account the physical constraints underlying the process. Three matching criteria were defined:

- Compatibility: Two descriptive elements can only match if they arise from the same physical marking (corners can only match with corners, 'blobs' with 'blobs,' etc.);
- Uniqueness: Almost always a descriptive element from the left image corresponds to exactly one element in the right image and vice versa;
- Continuity: The disparity of the matches varies smoothly almost everywhere over the image.

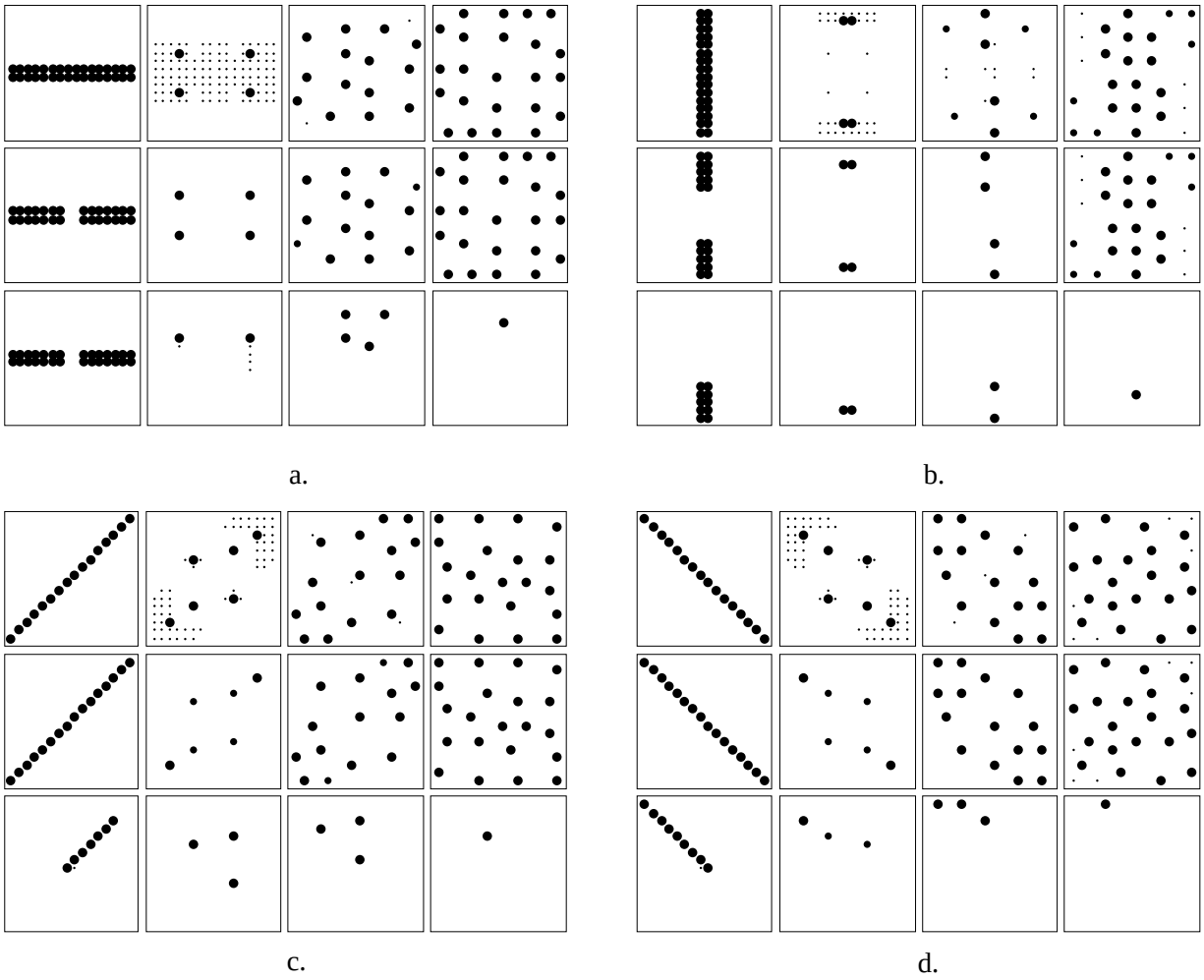


Figure 9.6: Feeding back activation values in the cognitron.
The four learning patterns are now successively applied to the network (row 1 of figures a–d). Next, the activation values of the neurons in layer 4 are fed back to the input (row 2 of figures a–d). Finally, all the neurons except the most active in layer 4 are set to 0, and the resulting activation values are again fed back (row 3 of figures a–d). After as little as 20 iterations, the network has shown to be rather robust.

Marr's 'cooperative' algorithm (also a 'non-cooperative' or local algorithm has been described (Marr, 1982)) is able to calculate the disparity map from which the depth can be reconstructed. This algorithm is some kind of neural network, consisting of neurons $N(x, y; d)$, where neuron $N(x, y; d)$ represents the hypothesis that pixel (x, y) in the left image corresponds with pixel $(x + d, y)$ in the right image. The update function is

$$N^{t+1}(x, y; d) = \sigma \left(\sum_{\substack{x', y'; d' \in \\ S(x,y;d)}} N^t(x', y'; d') - \epsilon \sum_{\substack{x', y'; d' \in \\ O(x,y;d)}} N^t(x', y'; d') + N^0(x, y; d) \right). \tag{9.9}$$

Here, ϵ is an inhibition constant, σ is a threshold function, $S(x, y; d)$ is the local excitatory neighbourhood, and $O(x, y; d)$ is the local inhibitory neighbourhood, which are chosen as follows:

$$S(x, y; d) = \{ (r, s, t) \mid (r = x \vee r = x - d) \wedge s = y \}, \tag{9.10}$$

$$O(x, y; d) = \{ (r, s, t) \mid d = t \wedge \| (r, s) - (x, y) \| \leq w \}. \tag{9.11}$$

The network is loaded with the cross correlation of the images at first: $N^0(x, y; d) = I_l(x, y)I_r(x + d, y)$, where I_l and I_r are the intensity matrices of the left and right image respectively. This network state represents all possible matches of pixels. Then the set of possible matches is reduced by recursive application of the update function until the state of the network is stable.

The algorithm converges in about ten iterations. Then the disparity of a pixel (x, y) is displayed by the firing neuron in the set $\{N(r, s; d) \mid r = x, s = y\}$. In each of these sets there should be exactly one neuron firing, but if the algorithm could not compute the exact disparity, for instance at hidden contours, there may be zero or more than one neurons firing.

9.5.2 Image restoration and image segmentation

The restoration of degraded images is a branch of digital picture processing closely related to image segmentation and boundary finding. An analysis of the major applications and procedures may be found in (Rosenfeld & Kak, 1982). An algorithm which is based on the minimisation of an energy function and can very well be parallelised is given by Geman and Geman (Geman & Geman, 1984). Their approach is based on stochastic modelling, in which image samples are considered to be generated by a random process that changes its statistical properties from region to region. The random process that that generates the image samples is a two-dimensional analogue of a Markov process, called a Markov random field. Image segmentation is then considered as a statistical estimation problem in which the system calculates the optimal estimate of the region boundaries for the input image. Simultaneously estimation of the region properties and boundary properties has to be performed, resulting in a set of nonlinear estimation equations that define the optimal estimate of the regions. The system must find the maximum a posteriori probability estimate of the image segmentation. Geman and Geman showed that the problem can be recast into the minimisation of an energy function, which, in turn, can be solved approximately by optimisation techniques such as simulated annealing. The interesting point is that simulated annealing can be implemented using a network with local connections, in which the network iterates into a global solution using these local operations.

9.5.3 Silicon retina

Mead and his co-workers (Mead, 1989) have developed an analogue VLSI vision preprocessing chip modelled after the retina. The design not only replicates many of the important functions of the first stages of retinal processing, but it does so by replicating in a detailed way both the structure and dynamics of the constituent biological units. The logarithmic compression from photon input to output signal is accomplished by analogue circuits, while similarly space and time averaging and temporal differentiation are accomplished by analogue processes and a resistive network (see section 11.2.1).