ORIGINAL RESEARCH

Using weightless neural networks for vergence control in an artificial vision system

Karin S Komati and Alberto F De Souza

Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, ES, Brazil

Abstract: This paper presents a methodology we have developed and used to implement an artificial binocular vision system capable of emulating the vergence of eye movements. This methodology involves using weightless neural networks (WNNs) as building blocks of artificial vision systems. Using the proposed methodology, we have designed several architectures of WNN-based artificial vision systems, in which images captured by virtual cameras are used for controlling the position of the 'foveae' of these cameras (high-resolution region of the images captured). Our best architecture is able to control the foveae vergence movements with average error of only 3.58 image pixels, which is equivalent to an angular error of approximately 0.629°.

Keywords: vergence, binocular vision, weightless neural network

Introduction

Our visual perception of the world is egocentric, ie centred in our perceived position in relation to the environment. Normally, we do not even notice this. If our perception were oculocentric, ie centred in the position of our eyes in relation to the environment, it would be awful, as we would perceive the world to be moving with the movements of our eyes. Nevertheless, in an apparent paradox, the visual brain areas that first analyse the attributes of the visual input possess an accurate oculocentric neural organisation (Kandel et al 2000); yet, when moving our eyes, we feel that the exterior world is steady. This shows that human visual perception depends not only on the image on the retina, but also on the knowledge of the position of the eyes, head and body. Therefore, the oculomotor system influences visual perception, and information regarding its status is important for visual perception of the world (Ebenholtz 2001).

This paper presents the first step of a large research effort whose main objective is to implement a system, based on artificial weightless neural networks (WNNs) (Austin 1998; Ludermir et al 1999), capable of emulating the human biological visual system capacity for creating a steady internal representation of the environment. The authors believe that the oculomotor system plays a significant role in building this internal representation, and started the study by modelling this system. WNNs were chosen as building blocks for modelling the biological visual system because they are simple, but powerful, highly flexible and have fast learning algorithms, which allow the study of large networks using standard computers currently available.

Here, a system is presented that models the human vergence oculomotor subsystem (Ebenholtz 2001). Vergence refers to the disjunctive eye movements whereby the eyes move symmetrically in opposite directions to ensure that, for the object of interest, the image in each eye falls on the fovea (central part of the retina) (Kandel et al 2000).

In the implementation presented here, images captured by virtual cameras are used for controlling the position of the cameras' 'foveae'. The control signals for positioning the foveae are generated by WNNs, which receive as input the images captured by the cameras. To implement this vergence system, current knowledge of the neurophysiology of the biological visual system and a software tool developed to emulate WNNs architectures were used. The tool, named MAE (Portuguese acronym for Event Associative Machine), allows simulation of complex WNN architectures with hundreds of thousands of neurons in a standard PC running Linux®. An example architecture of a vergence control system studied using MAE is shown in Figure 1, while Figure 2 shows a screenshot of MAE running the neural architecture of Figure 1.

In Figure 1, Image_left and Image_right are images captured by two virtual cameras positioned in distinct places in the virtual space to emulate the physical separation of the eyes.

Correspondence: Alberto Ferreira De Souza, Departamento de Informática, Universidade Federal do Espírito Santo, Av Fernando Ferrari, S/N, Vitória, ES 29060-970, Brazil; tel +55 27 3335 2641; fax +55 27 3335 2650; email alberto@inf.ufes.br

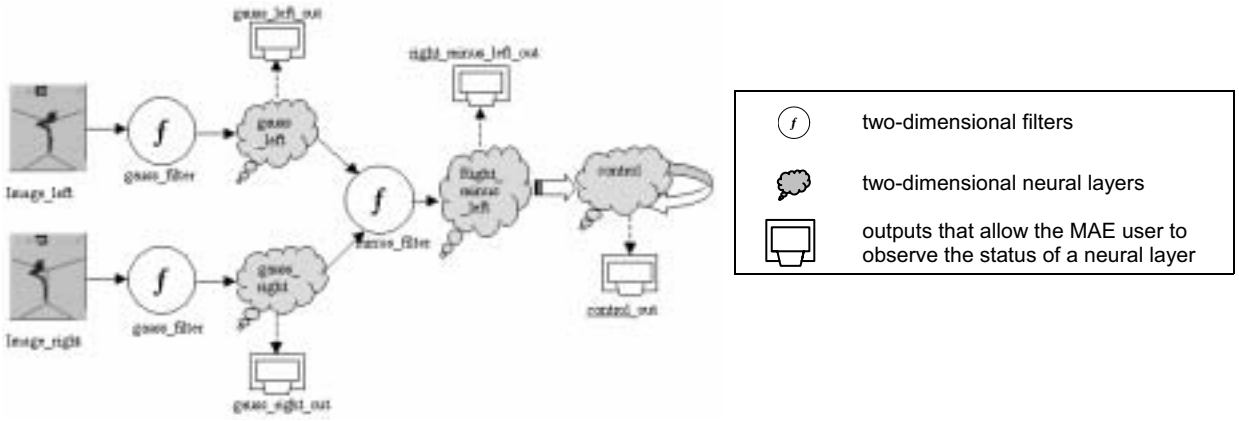

Figure 1 Simple neural architecture.

Several neural architectures like the one shown in Figure 1 were implemented and tested using filters and trained neural layers to emulate neurophysiological properties observed in the biological visual system. The best performing architecture produces vergence movements with average error of only 3.58 image pixels, which is equivalent to an angular error of approximately 0.629°.

This paper briefly describes some parts of the human biological visual system associated with vergence control. It introduces WNNs and the WNN neurons used to implement the experimental architectures, and then describes the framework (MAE) for modelling those WNNs. The methodology used to design and evaluate the architectures is discussed, followed by the experimental results. An overall discussion of the paper followed by related work, and then conclusions and directions for future research close the paper.

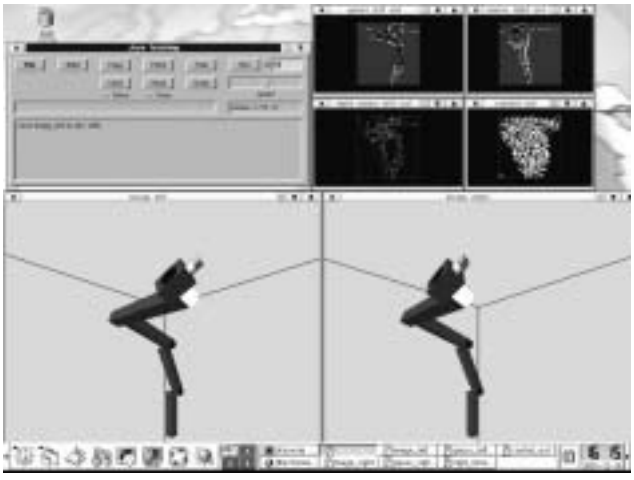

Figure 2 MAE screenshot of a simple neural architecture.

Biological vision

Vision starts with two external sensors – the eyes. The retina, situated at the back of the eyeball, is where the image of the outside world is captured. The neurons of the retina transform light energy into electrical signals that are transmitted to the brain via the optic nerve. The retina ganglion cells, which drive the optic nerve, can be separated into two categories: parvocellular (small cells) and magnocellular (large cells). These two categories of cells are the starting point of parallel pathways that can be identified throughout most parts of the brain devoted to visual processing (Kandel et al 2000).

Visual pathways

The optic nerve connects to the lateral geniculate nucleus (LGN) of the thalamus, and this to the visual cortex. The visual cortex is usually separated into 5 areas: V1–V5 (Kandel et al 2000). The LNG connects to the primary visual cortex, or V1 (also called striate cortex) (see Figure 3). V1 possesses cells that detect several basic attributes of the visual stimuli; some neurons detect stimuli that have a certain angular orientation, others are sensitive to colour, others detect disparities between stimuli coming from each eye, etc.

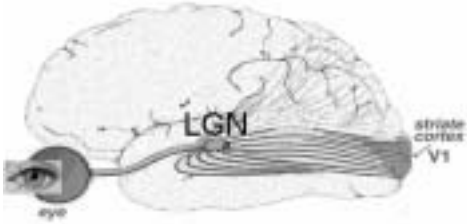

Figure 3 Eye, LGN, V1.

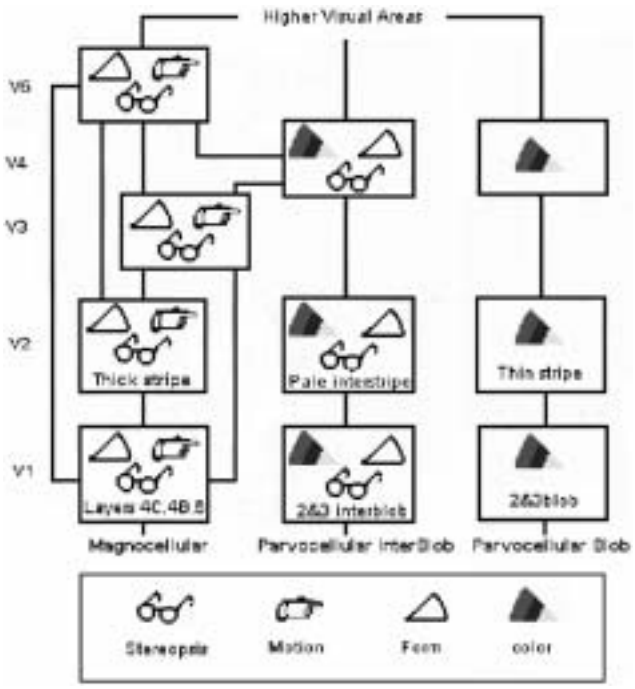

Figure 4 Parallel visual pathways. Adapted from Kandel et al (2000).

From V1, the visual neuronal impulses are directed to V2, which then projects to V3, V4 and V5 (middle temporal cortex or MT), and each of these areas then sends information to any of the 20 or more other areas of the brain that process visual information. This general arrangement can be subdivided into three parallel pathways, which start in the parvocellular and magnocellular cells of the retina; in V1, the parvocellular pathway is divided into two, (parvocellular interblob and parvocellular blob) as can be seen in Figure 4. Although each pathway is somewhat distinct in function, there is intercommunication between them.

The magnocellular cortical pathway starts in the layer 4Cα of V1 (Figure 4). This projects to layer 4B, which then projects to the thick stripes of V2. Area V2 then projects to V3, V5 and the medial superior temporal cortex (MST). This pathway is an extension of the magnocellular pathway from the retina and LGN, and processes visual detail leading to the perception of shape in area V3, and movement or motion in areas V5 and MST. Retinal magnocellular ganglion neurons show a high sensitivity to contrast, low spatial resolution and high temporal resolution or fast transient responses to visual stimuli. These characteristics make the magnocellular branch of the visual system especially suitable to quickly detect novel or moving stimuli.

In the second visual pathway, neurons of the parvocellular interblob areas of V1 project to the pale stripes of V2, and these to the inferior temporal cortex (Figure 4). This pathway possesses several feature detectors (simple, complex and hypercomplex cells) and is responsible for complex form recognition.

Finally, in the third and mixed visual cortical pathway, neurons in the blobs of V1 project to the thin stripes of V2. The thin stripes of V2 then project to V4. Area V4 receives input from both the parvocellular and magnocellular pathways of the retina and LGN. This parvocellular portion of V4 is particularly important for the perception of colour and maintenance of colour (colour constancy) regardless of lighting.

Retinal parvocellular ganglion neurons show a low sensitivity to contrast, high spatial resolution and low temporal resolution or sustained responses to visual stimuli. These cellular characteristics make the parvocellular visual pathways especially suitable for the analysis of detail in the visual world.

These separated parallel pathways decode/extract the elementary properties of the objects in the visual field, such as colour, form and orientation, producing a rather complex internal representation of what we see. However, this representation is not captured instantly: we examine our surroundings all the time, focusing our attention in different parts of our field of view through the movement of our eyes. The authors believe that the mind creates this internal representation in a way similar to that of a painter working on a canvas, where the eyes serve as the painter's brush.

Oculomotor system

Several image-preprocessing operations are performed in the retina, and two are of interest for this work. First, the retina's ganglion cells act like contrast discriminators, being rather insensitive to uniform light. Second, the amount of ganglion cells devoted to different areas of the retina is larger at the fovea and decreases in the direction of the borders, following approximately a Gaussian density distribution. This is perhaps one of the main reasons for the existence of ocular movements.

Humans have five types of eye movements that put the foveae on a target and keep them there. They are listed in Table 1 (see Ebenholtz 2001). The first four are conjugate movements: both eyes move the same amount in the same direction. The fifth – vergence – is disconjugate: the eyes move in different directions and sometimes by different amounts.

Thanks to the vergence oculomotor subsystem, when we look at an object that is near or far from us, each eye

Table 1 A functional classification of eye movements

Movement	*Function*
Movements that stabilise the eye when the head moves	
Vestibulo-ocular	For brief or rapid head rotation
Optokinetic	For sustained or slow head rotation
Movements that keep the fovea on a visual target	
Saccade	Brings new objects of interest onto the fovea
Smooth pursuit	Holds the image of a moving target on the fovea
Vergence	Adjusts the eyes for different viewing distances in depth

moves differently to keep the image of the object aligned precisely on each fovea. If the object is near, the eyes converge; if the object is far, the eyes diverge. The difference in the retinal position of an object in one eye compared with its position in the other is referred to as retinal disparity, and the detection of this disparity is important for vergence control.

Disparity sensitivity in the visual system

Numerous studies made on monkeys show that a large population of visual cells is capable of encoding the amplitude and sign of horizontal disparities. Horizontal disparity detectors were found in cortical visual areas V1, V2, V3, V5 and MST. According to their disparity tuning functions, which describe how cells respond to stimulus with different disparities, these cells can be classified into several groups (Gonzalez and Peres 1998). Six such groups are shown in Figure 5. Models of four of these were created and used in the vergence control system and are discussed below.

TE (tuned excitatory) cells have a disparity tuning function that displays binocular facilitation over a narrow range of disparities around zero and binocular suppression for negative or positive disparities (see Figure 5b). That is, they have a positive peak at zero disparity, are symmetrical around zero disparity and have a narrow tuning width. TN (tuned near) and TF (tuned far) cells have disparity tuning functions similar to TE cells but peak at negative or positive disparities, respectively (see Figures 5a, 5c). TI (tuned inhibitory) cells also have a disparity tuning function similar to TE cells, with the difference that the peak is negative. Note that many disparity-sensitive cells are not equally driven by both eyes but show different degrees of preference by one of the eyes (Gonzalez and Peres 1998). This may make their response slightly different from those shown in Figure 5.

In addition to helping control vergence movements, disparity-sensitive cells, together with those that report the position of the eyes and others responsible for visual memory, are possibly important for creating the internal representations of form and distance of objects in the visual field (Semmlow et al 1994).

Weightless neural networks

WNNs are based on artificial neurons that have binary inputs and outputs and no modifying weight between neurons

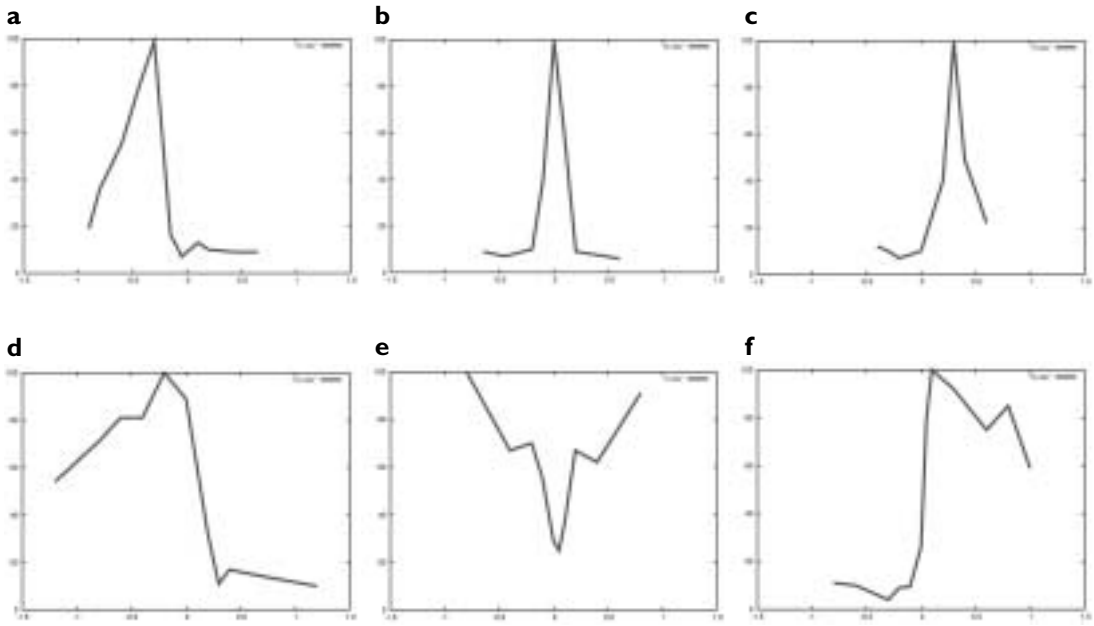

Figure 5 Disparity-detection cells: **(a)** tuned near (TN); **(b)** tuned excitatory (TE); **(c)** tuned far (TF); **(d)** near (NE); **(e)** tuned inhibitory (TI); and **(f)** far (FA).

(Ludermir et al 1999). Like biological neurons, and neurons of other types of artificial neural networks, WNN neurons have synapses connecting them together, forming the WNN and connecting the WNN to its inputs and outputs. However, each synapse is used to collect only a single bit (0 or 1) of the neuron's total input. Each such neuron has a look-up table that can be implemented using random access memory (RAM). The neuron's total input forms a bit pattern that is used as the address of an entry of its look-up table, and the look-up table entry value is used to compute the neuron's output. Instead of adjusting weights, WNN training consists of changing the contents of look-up tables' entries, which allows highly flexible and fast learning algorithms. The flexibility comes from the WNN neuron capacity of implementing any logical function of its inputs (an input pattern can be any address of the neuron RAM, and the addressed position may hold any binary value) and the expedite learning from the mutual independence between look-up table entries, which permits changing an entry without interfering with others.

In the mid-1960s, Igor Aleksander (1966) proposed universal logic circuits, which can be implemented with RAM, as neurons of WNN. Other weightless models were later proposed, but, until the late 1980s, all of them only generalised at the network level (individual neurons did not have generalisation properties). To overcome this, Aleksander (1989) suggested the inclusion of a spreading phase, responsible for generalising information from the training set through storing clusters of extra slightly changed input patterns having each training input as centroid. Generalising neurons are known as generalising RAMs (GRAMs).

GRAMs are often implemented as virtual GRAMs (VGRAMs) (Mrsic-Flogel 1991). A VGRAM is a GRAM in which space is not allocated for patterns that are never found during the training phase, and generalisation is achieved through calculation. During the training phase, a VGRAM neuron stores each input pattern and the corresponding desired output into its look-up table. During the recall phase, input patterns are compared with each input of the stored input–output pairs. If there is an exact match, the neuron responds with the stored output; otherwise, one of two possible outcomes occurs: (1) the neuron responds with the output of the nearest stored pattern; or (2) if there is more than a single stored pattern at the same Hamming distance of the input, the neuron responds with a random value.

VGRAM neurons can be made sensitive to colour or shades of grey (Austin 1998). This is achieved using neuron inputs dedicated to a colour or grey level. These inputs respond with a binary 1 when receiving a value equal or near its colour or grey level, and with a binary 0 otherwise. The binary outcomes of these inputs form the input patterns of the VGRAM neurons.

In this work, VGRAM neurons were used to implement WNN-based architectures for controlling vergence.

The MAE tool

The MAE is an open-source framework for modelling VGRAM WNNs. MAE is similar to the Neural Representation Modeller (NRM),[1] but differs in three main aspects: it is open source; it runs on UNIX® (currently, Solaris™ and Linux®); and it uses a textual language to describe WNNs.

MAE allows designing, training and analysing the behaviour of modular WNNs whose basic modules are two-dimensional neural layers and two-dimensional filters. The neurons that form these neural layers can have several characteristics (type, input sensitivity, memory size etc), and the user, using C programming language, can freely design filters. MAE users design modular WNNs using a Neural Architecture Description Language (NADL) we have created. NADL source files are compiled into MAE applications, which have a built-in graphical user interface and a C-like Control Script Language (CDL) interpreter. The users can train, recall and alter the state of a neural layer using the graphical interface or scripts in CDL. NADL and CDL allow the creation of integer variables and the use of 'for', 'while' and 'do-while' C-like loops, which enable the creation of elaborate WNNs and powerful scripts for analysing them.

Methodology

The retina model

A simple retina model was created to provide visual input to the vergence control system. An artificial retina that follows this model, like its biological counterpart, is sensitive to contrast and has higher resolution at its centre (fovea) and low resolution at its borders. The retina model was implemented using an MAE filter named 'gauss' (Figure 1), which receives as input the image that enters the virtual cameras (Image_left and Image_right in Figure 1) and produces as output an image that maps the input following a Gaussian distribution centred in a point on the input image.

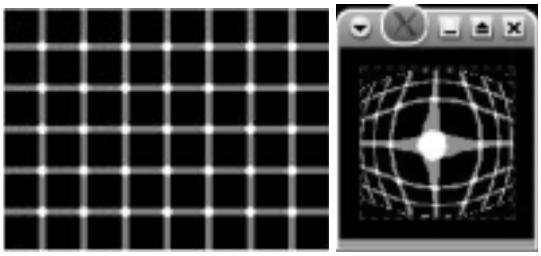

Figure 6 Test image and the corresponding gauss filter output.

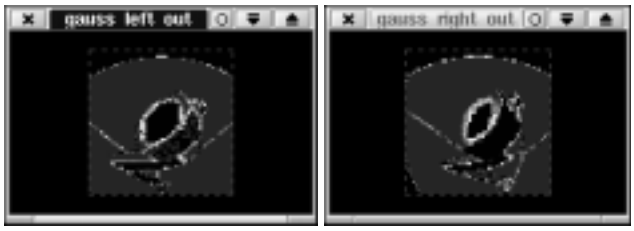

Figure 7 TE (tuned excitatory) filter input images.

The model produces a much distorted version of the input image, as shown in Figure 6, but possibly similar to the parvocellular output of the biological retina. Two retinas were used to allow binocular vision.

Our gauss filter was implemented using a 9×9 'difference of Gaussians' (DOG) kernel (Rodieck 1965). The application of this kernel to a region of the input image results in an approximation of the DOG model of retinal ganglion cells. However, the convolution of the kernel with the input image alone would not produce the result shown in Figure 6, since this image-processing technique (convolution) is applied linearly over the image (Gonzalez and Woods 1992). So, instead of applying the kernel linearly, the DOG kernel was applied according to a Gaussian distribution centred in the point of the input image to where the fovea of each virtual camera is aimed. This emulates the distribution of ganglion cells across different areas of the retina.

The TE filter

Also modelled were TE, TN and TF disparity detection cells. They were emulated with the MAE TE filter, which receives as input the images preprocessed by the left and right retinas (gauss filters). According to an integer 'displacement parameter', the TE filter emulates a TE, TN or TF layer of cells. The operation performed by the TE filter is very simple: each output pixel is the average of corresponding pixels of each input image. If the integer parameter is zero, the input images are aligned, and the output of the filter emulates a layer of TE neurons. If the displacement parameter is positive, the input images are displaced the number of pixels (the left image to the left and the right image to the right) equal to the parameter value. This emulates a layer of TN neurons, since points of the input images belonging to objects nearer to the virtual cameras than the target of both foveae tend to coincide if the images are displaced this way. If the parameter is negative, the input images are displaced the number of pixels equal to the parameter value but in the opposite direction. This emulates a layer of TF neurons.

Figure 7 shows the outputs of the left and right virtual retinas obtained from the input images shown in Figure 2 (note where each fovea is aimed), while Figure 8 shows examples of TE, TN and TF neuron layers.

The minus filter

Another filter used was the 'minus' filter. It receives two images and returns one, where each pixel is calculated in the following way: if the value of a pixel, f, in the first image is equal or larger than the value of the corresponding pixel, s, in the second image, the output is equal to $f–s$, otherwise it is equal to zero. The output of this filter emulates a layer of TI neurons with output dominated by one of the virtual cameras.

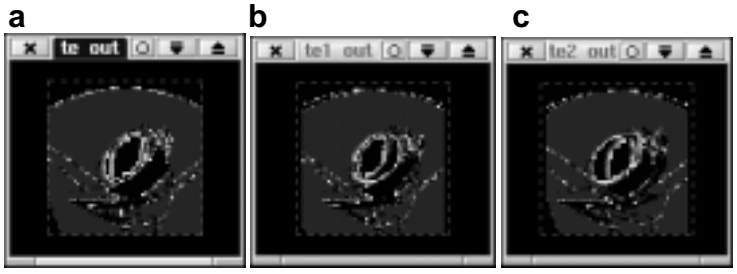

Figure 8 TE (tuned excitatory) filter output images. **(a)** Output image with displacement parameter of zero. **(b)** Output image with displacement parameter of 2. **(c)** Output image with displacement parameter of –2.

The virtual world

The virtual world used in these experiments was developed using OpenGL (Blythe 1999). The code generates pairs of stereo images as the one shown in Figure 2 (Image_right and Image_left windows). The images are two-dimensional 512×512-pixel projections of a three-dimensional (3-D) robot arm model illuminated by a single ambient light and captured by cameras with field of view of 90°. The images captured are similar to those that would be seen by a human observer (7-cm camera separation was used) approximately 1 m away from a 1-m long similar robot arm. The OpenGL code for details on colours, textures and other properties of the robot and its surroundings can be viewed at http://www.inf.ufes.br/~alberto/robot.c.

The WNN vergence control system

The WNN vergence control system in this study receives as input several filtered versions of the left and right input images and generates as output a value that is added to the current horizontal position of the fovea of a single virtual camera. In this model, the right virtual camera is the dominant camera and decides (actually, a human operator) where to look, while the left virtual camera is controlled by the WNN vergence control system. The only 'motor neuron function' emulated is the horizontal eye movement. So, the virtual camera's fovea goes to the left or to the right of its previous position depending on the control system output signal. However, to generate the correct output, the WNN must be trained.

The WNN training was performed according to the following simple algorithm:

1. The operator chooses a point and aims the dominant (right) camera's fovea at it (using a mouse click).
2. The system automatically moves the controlled (left) camera's fovea by the same amount, possibly positioning it incorrectly due to an incorrect vergence angle for the chosen point.
3. The operator informs the training system (using a mouse click) of the correct position for the fovea of the left camera.
4. The system trains the WNN to output a pattern suitable for moving the fovea of the left camera to the correct position.
5. The system allows the WNN to move the fovea of the left camera to the correct position.
6. Go back to Step 1.

This algorithm was implemented using a CDL script and used to train several WNN-based architectures for vergence control. All networks were trained using 24 points.

In 'recall mode', the system continuously acquires images, calculates the output of the filters and the outputs of the WNNs, and generates the control signals that move the fovea of the controlled camera. Note that the fovea movement modifies the output of the retina of the controlled camera, closing the control loop. The recall mode procedure was implemented with a CDL script and repeated 50 times for each testing point. To test the performance of the system, 24 unknown points (not used during training) were used.

Performance parameters

Two performance parameters were used: 'average error' and the number of 'unstable points'. The average error is the average number of pixels the controlled fovea is away from the correct point for 40 out of the 50 recall script iterations (the first 10 iterations are excluded, as the system uses it to incrementally move the controlled fovea to the target point). For some testing points, an architecture may not be able to stabilise vergence and may move the controlled fovea to the limit of the input image. These points count as unstable points.

Characteristics of the architectures

Many architectures were implemented, but all can be described using three main characteristics: type of neuron, size and type of the inputs to the control neural layers, and type of control strategy.

Two types of neuron were used: 'default' and 'rnd_mem_zrizro'. The default neuron responds exactly as in the literature, but the rnd_mem_zrizro neuron outputs zero when all inputs are zero.

The image of the virtual cameras, after being preprocessed by the retinas (gauss filters), can be filtered in five different ways, producing five possible inputs for the vergence control neural layers: (1) TE, or the output of the TE filter with displacement equal to zero, which emulates a layer of TE neurons; (2) TE1, the output of the TE filter with displacement 2, which emulates a layer of TN neurons; (3) TE2, the output of the TE filter with displacement –2, which emulates a layer of TF neurons; (4) right_minus_left, or the output of the minus filter with the right retina minus the left retina as input, which emulates a layer of TI neurons with right eye dominance; and (5) left_minus_right, which

emulates a layer of TI neurons with left eye dominance. The number of connections, or synapses, between each vergence control neuron and each of these filters can vary. In addition, the way these connections are distributed throughout the output of the filter can also vary. Two types of connection distribution were implemented: 'Gaussian' and 'random'. In the Gaussian type, the synapses of the neurons are connected to the output of a filter according to a two-dimensional Gaussian distribution centred in the point of the filters' output that corresponds to the neuron position in the control layer – this type of synapse connection pattern tries to emulate a retinotopic mapping. In the random type, the synapses of the neurons are connected to the filter randomly. The outputs of the filters can assume 256 different shades of grey, and the connections, or neuron synapses, are sensitive to specific ranges of shades of grey. A synapse responds with a 1 if the point in the filter where it is connected has a grey level within its sensitivity, or a 0 otherwise. Each synapse randomly receives one of eight different ranges of sensitivity when the WNN is first created (0–31, 32–63, 64–95, …, 224–255). The sensitivity of a synapse does not change after WNN creation.

Two types of control strategies were implemented: 'ResetY' and 'Set2Layers'. In the ResetY strategy, a single vergence control neuron layer of VGRAM neurons with binary outputs is used. This layer is divided into two halves, left and right. The number of active neurons on the right half minus the number of active neurons on the left half is the control signal. The control action consists of adding this value to the current horizontal position of the controlled fovea. In Set2Layers, two neuron layers of VGRAM neurons with binary outputs are used; one is trained to positive movements (to the right), and the other to negative movements (to the left). The number of active neurons on the right layer minus the number of active neurons on the left layer is the value added to the current horizontal position of the controlled fovea. On both strategies during training, the control neural layers are recalled 'seeing' the image of the incorrect vergence position before training. Previous learned information or random values are output according to the VGRAM neurons' behaviour. This output is then changed by turning off a percentage of the appropriate neurons to produce the correct fovea movement. The control neural layers are then trained 'seeing' the wrong image and outputting the corrective control action.

Implemented architectures

Table 2 summarises the characteristics of the eight architectures implemented. Architecture 1 of Table 2 is actually the architecture depicted in Figure 1. Its NADL specification is shown in Figure 9 and explained below.

Line 1 of Figure 9 (the lines are numbered to facilitate the description) is a global command that sets the neuron memory size of all neurons to 32. This neuron memory size was used in all architectures because the number of training points is equal to 24. Lines 3–8 specify the right and left input image sizes (512 × 512 pixel), their type of pixel (greyscale: 256 shades of grey) and their generator and controller C functions. These two images are filtered by the gauss filter, lines 13 and 14, producing the neural layers gauss_right and gauss_left, which are created by lines 10 and 11. Note that the size of these neural layers is only 64 × 64 neurons. Lines 16 and 17 create two output windows that allow the human operator to monitor gauss_right and gauss_left. These outputs are connected to gauss_right and gauss_left by lines 18 and 19. These windows are the two top-right windows of Figure 2 (both foveae are pointing to the centre of the image).

Table 2 Implemented architectures

	Architecture							
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Neuron type								
rnd_mem_zrizro	X	X	X					
default				X	X	X	X	X
Control layer inputs								
right_minus_left	32G	32G		32G	32G		32G	32R
left_minus_right		32G			32G			
TE, TE1 and TE2			32G			32G	16G	16R
Control strategy								
ResetY	X	X	X	X	X	X	X	
Set2Layers								X

```
1. set NEURON_MEMORY_SIZE = 32;
2.
3. input image_right[512][512] with greyscale outputs
4.    produced by input_generator()
5.    controlled by input_controller_right();
6. input image_left[512][512] with greyscale outputs
7.    produced by input_generator()
8.    controlled by input_controller_left(control_out);
9.
10. neuronlayer gauss_right[64][64] with greyscale outputs;
11. neuronlayer gauss_left[64][64] with greyscale outputs;
12.
13. filter image_right with gauss_filter()producing gauss_right;
14. filter image_left with gauss_filter() producing gauss_left;
15.
16. output gauss_right_out[64][64];
17. output gauss_left_out[64][64];
18. outputconnect gauss_right to gauss_right_out;
19. outputconnect gauss_left to gauss_left_out;
20.
21. neuronlayer control[64][64] of rnd_mem_zrizro neurons
22.    with b&w outputs;
23. neuronlayer right_minus_left[64][64] with greyscale outputs;
24.
25. output right_minus_left_out[64][64];
26. output control_out[64][64];
27. outputconnect right_minus_left to right_minus_left_out;
28. outputconnect control to control_out;
29.
30. filter gauss_right, gauss_left with minus_filter()
31     producing right_minus_left;
32.
33. connect right_minus_left to control
34.    with 32 random inputs per neuron and
35.    Gaussian distribution with radius 3.0;
36. associate control with control;
```

Figure 9 NADL specification of Figure 1 architecture.

Lines 1–19 of Figure 9 are common to all architectures implemented and describe an architecture up to its 2 retinas. However, lines 21–36 are specific to architecture 1. The control neural layer of this architecture is created in line 21 and is composed by rnd_mem_zrizro neurons: the first X of the column of architecture 1 in Table 2 indicates this characteristic. Note that the function input_controller_left receives the output connected to the control neural layer, control_out (Figure 9, line 28) as a parameter (Figure 9, line 8). This function is responsible for executing the control action based on the control neural layer output. In the case of architecture 1, the control strategy is ResetY (see Table 2). Therefore, the control action consists of subtracting the number of active neurons on the left from the number of active neurons on the right of the control neural layer and adding this difference to the current position of the left fovea.

The control neural layer receives input from a right_minus_left filter only – each neuron of this neural layer has 32 Gaussian-distributed connections to this filter output. This is specified by lines 33–35 of Figure 9 and indicated in Table 2 by the symbol 32G in the column of architecture 1. Lines 25, 26, 27 and 28 allow monitoring the control neural layer and the right_minus_left filter outputs.

Finally, line 36 of Figure 9 specifies that the system should train the control neural layer using its own output as target pattern.

Table 3 Results for known points

	Architecture							
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Average error	19.20	23.59	4.48	11.64	10.02	–3.84	3.73	**0.26**
Unstable points	3	4	3	7	4	7	**0**	**0**

Table 4 Results for unknown points

	Architecture							
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Average error	28.12	3.77	43.18	**1.77**	9.11	8.02	14.46	–3.58
Unstable points	**0**	3	9	7	8	5	**0**	**0**

Bold indicates the best results.

Experimental results

Tables 3 and 4 summarise the experimental results obtained with the eight architectures specified in Table 2 according to the performance parameters presented in the *Methodology* section. The architectures were tested with the points used during training (Table 3) and with unknown points (Table 4). The best results in Tables 3 and 4 are shown in bold. The number of training and testing points used was equal to 24, and they were the same for all architectures. The disparities in the training dataset ranged from 0 to 56 pixels (0°–9.8°), with average equal to 8.54 pixels (1.5°); and in the testing dataset from 0 to 77 pixels (0°–13.5°), with average equal to 14.29 pixels (2.5°). The standard deviation of the training and testing datasets were equal to 15.28 (2.7°) and 20.43 pixels (3.6°), respectively. Figure 10 shows the evolution of the disparity error observed with architecture 8 (Table 2) for one of the 24 testing points from the onset of stimulus to the end of the 50 testing iterations of the control loop.

In architecture 1 (Table 2), the neuron type of the control layer is rnd_mem_zrizro, and each of these neurons is connected to a right_minus_left filter only. The connection between each neuron of the control layer and the right_minus_left filter is of the 32G type. In the results presented in Table 2, one can note that architecture 1 was unstable for 3 points of the training dataset. It has not shown

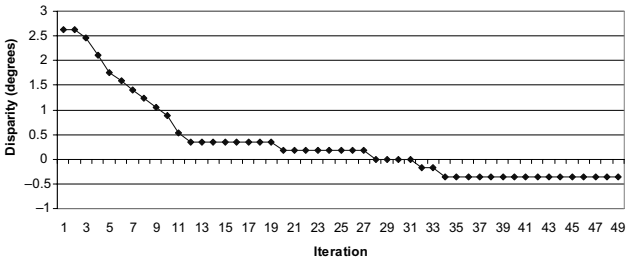

Figure 10 Evolution of the disparity error in one of the experiments.

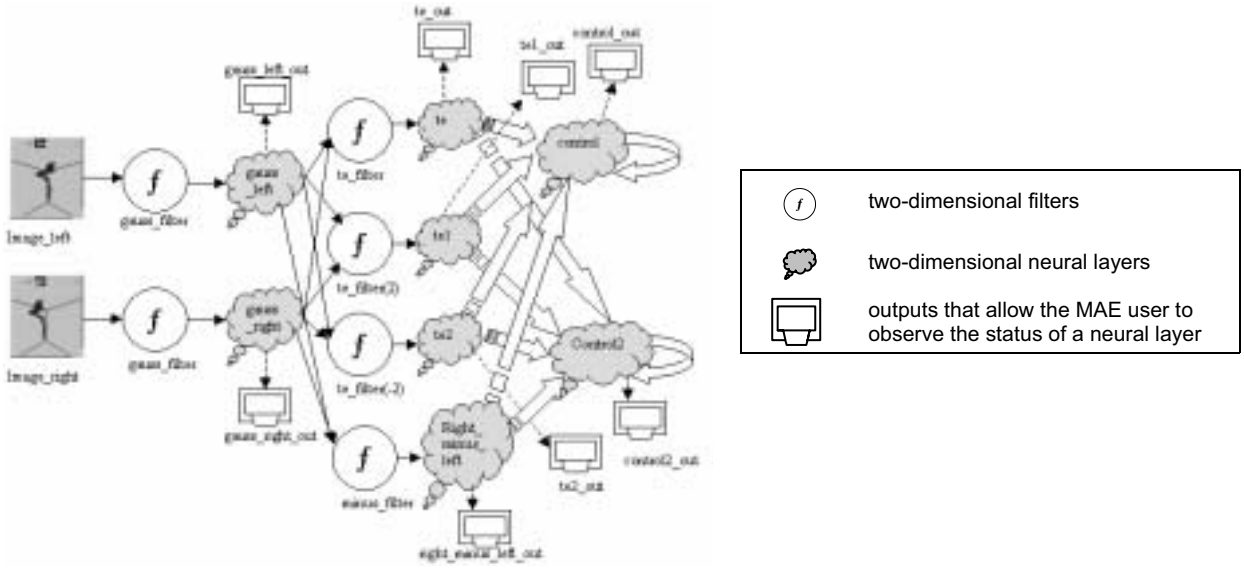

Figure 11 The best architecture (architecture 8).

any unstable points for the testing dataset; however, the average error was high for both sets. Architecture 2 is similar to architecture 1 but has double the inputs per neuron, since it possesses a 32G connection to a left_minus_right filter in addition to the 32G connection to the right_minus_left filter (Table 2). This architecture is significantly better than architecture 1 in the average error metric for the training dataset, but presents more unstable points and a worse performance for the average error with the test dataset. Architecture 3 has three times the number of synapses as architecture 1 (one 32G connection to TE, one to TE1 and one to TE2) (see Table 2), but its performance is still inferior to architecture 1 for the average error and number of unstable points with the testing dataset. Except for the type of neurons of the control neural layer (default type), architectures 4, 5 and 6 are identical to architectures 1, 2 and 3, respectively. If compared with the previous ones, none of these architectures show a consistent advantage on both metrics simultaneously. Architecture 7 presents the best overall performance for the training set, and no unstable points with either set. This performance is the result of allowing the control layer to receive tuned excitatory (TE, TE1 and TE2) and tuned inhibitory (right_minus_left) information simultaneously.

It can be concluded from the results presented in Tables 3 and 4, that architecture 8 presents the best results: showing no unstable points for known and unknown points, the smaller average error for known points and the runner-up average error for unknown points. The best performing architecture produces vergence movements with average error of only 3.58 image pixels, which is equivalent to an angular error of approximately 0.629°. Different from architecture 7, this architecture has connections of type R and Set2Layers control strategy. Type R connections allow neurons of the control neural layers to examine the whole output of the filters, and the Set2Layers control strategy allows neurons that see both sides of the field of view to influence the decision to move the controlled fovea to their specific trained direction. The performance of architecture 8 suggests that global information about the field of view is necessary for vergence control. Architecture 8 is shown in Figure 11.

Discussion

In this system, vergence control is achieved using data coming from most of the input image pixels but with greater attention devoted to the fovea region. The system allows the calculation of the distance of different points of objects in 3-D space by aiming the fovea of the dominant camera at it and calculating the vergence angle after the system achieves vergence. With this information and the fovea image of parts of the 3-D objects, the authors believe it is possible to build a representation of these objects internal to the system. However, we are still a long way from that, since we first need to build the parts of the system capable of: choosing the interesting points of the 3-D objects (saccadic movement control), recognising these points, building a 3-D representation of the objects and grouping

all these objects in a consistent internal representation of the 3-D scene.

Related work

Several works dealing with vergence in the context of binocular tracking can be found in the literature (Kita et al 1994; Marefat and Wu 1996; Bernardino and Santos-Victor 1998). The work presented in this paper is similar to these in various respects; however, to our knowledge, there is no similar work in the literature using WNN to control vergence.

Conclusion

This paper presents the methodology used to develop a vergence control system for an artificial vision system based on WNNs. The system was developed using the MAE tool created for modelling WNN architectures. Current knowledge of the human biological vision system was used in devising several architectural models of vergence control systems. The best architecture produced vergence movements with average error of only 3.58 image pixels, which is equivalent to an angular error of approximately 0.629°.

As future work, the authors plan to use real cameras to test the system and to develop a tool to create and test neural architectures automatically using genetic programming techniques. This tool would allow searching the best parameters of the WNN architectures in a straightforward way; the authors believe this will be important research towards implementing a WNN system capable of emulating the human biological visual system capacity for creating a steady internal representation of the environment.

Notes

[1] Neural Representation Modeller (NRM) is a Microsoft Windows®-based tool developed by the Neural Systems Engineering Group at Imperial College London (http://www.sonnet.co.uk/nts).

References

Aleksander I. 1966. Self-adaptative universal logic circuits. *IEE Electron Lett*, 2:231.

Aleksander I. 1989. Ideal neurons for neural computers. In Eckmiller R, Hartmann G, Hauske G, eds. Parallel processing in neural systems and computers. Amsterdam: Elsevier Science. p 225–8.

Austin J, ed. 1998. RAM-based neural networks. London: World Scientific.

Bernardino A, Santos-Victor J. 1998. Visual behaviors for binocular tracking. *Robotics Autonomous Syst*, 25:137–46.

Blythe D. 1999. Advanced graphics programming techniques using OpenGL [online]. Accessed 9 Oct 2003. URL: http://www.opengl.org/developers/code/sig99/advanced99/notes/notes.html

Ebenholtz S. 2001. Oculomotor systems and perception. New York: Cambridge Univ Pr.

Gonzalez F, Peres R. 1998. Neural mechanisms underlying stereoscopic vision. *Prog Neurobiol*, 55:191–224.

Gonzalez RC, Woods RE. 1992. Digital image processing. Reading, MA: Addison-Wesley Publ.

Kandel ER, Schwartz JH, Jessell TM. 2000. Principles of neural science. 4th ed. London: Prentice-Hall.

Kita N, Rougeaux S, Kuniyoshi Y et al. 1994. Binocular tracking based on virtual horopters. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS'94). Piscataway, NJ: IEEE. p 2052–7.

Ludermir TB, Carvalho A, Braga AP et al. 1999. Weightless neural models: a review of current and past works. *Neural Comput Surv*, 2:41–61.

Marefat MM, Wu L. 1996. Purposeful gazing and vergence control for active vision. *Robotics Computer-Integrated Manufacturing*, 12: 135–55.

Mrsic-Flogel J. 1991. Approaching cognitive system design. In: Proceedings of the International Conference on Artificial Neural Networks (ICANN'91). North Holland: Elsevier Science. p 879–83.

Rodieck RW. 1965. Quantitative analysis of cat retinal ganglion cell response to visual stimuli. *Vis Res*, 5:583–601.

Semmlow JL, Hung GK, Horng JL et al. 1994. Disparity vergence eye movements exhibit preprogrammed motor control. *Vis Res*, 34:1335–43.