# Convolutional Network for Vergence Control

Nikolay Chumerin*, Agostino Gibaldi†, Silvio P. Sabatini† and Marc M. Van Hulle*
* Laboratorium voor Neuro- en Psychofysiologie, Katholieke Universiteit Leuven,
Campus Gasthuisberg, Herestraat 49, bus 1021, 3000 Leuven, Belgium
† Department of Biophysical and Electronic Engineering, University of Genoa,
Via all'Opera Pia 11/A, 16145 Genova, Italy

***Abstract*—We present a biologically-inspired model for the one-shot vergence control of a robotic head, which has been used for an investigation of two vergence control networks. Both networks do not work with explicitly computed disparity, but extract the vergence control signal from the postprocessed response of a population of disparity tuned complex cells, the actual gaze direction and the actual vergence angle. Training and evaluation of the networks are also discussed.**

## I. Introduction

For the depth perception humans use a number of cues, but in this work we are focusing on the *retinal binocular disparity* and the *vergence angle* of the eyes. There are experimental evidences [1], [2] showing that, specific and separate tasks of depth perception and vergence eye movements are based on the activity of complex cells of the primary visual cortex (V1), which in turn contain distributed representation of binocular disparity. Vergence control models that are based on a distributed representation of binocular disparity [3], [4], usually require first the computation of the disparity map, thus limiting the functionality of the vergence system inside the sensitivity range of the population of cells specialized for depth perception. As for the control of vergence larger disparities have to be discriminated while keeping a good accuracy around the fixation point for allowing finer refinement and achieving stable fixations, alternative strategies might be employed.

With the use of a modular framework discussed in Section II, we have investigated two vergence control models, that combine the population responses without taking a decision, but extracting, directly from the population responses, a disparity-vergence response that allows us to nullify the disparity in the fovea, even if the stimulus presented is far beyond the disparity sensitivity range. The first model, similarly to [5], obtains disparity-vergence response by linear combination of the pooled population response (see IV). Our experiments support findings in [5], where has been shown that the linear model can simulate 'dual-mode' vergence control [6] and produce accurate vergence for a *simplified* experiment (fronto-parallel planar stimuli, allowed to move only in Z-axis direction). Unfortunately, in the *general case* experiments, where restrictions on the stimuli are dropped, the linear model often produces biased results. This fact motivated us to investigate a more sophisticated vergence control model, which uses a convolutional neural network for the mapping of the disparity population response to the vergence control signal (see Section V).

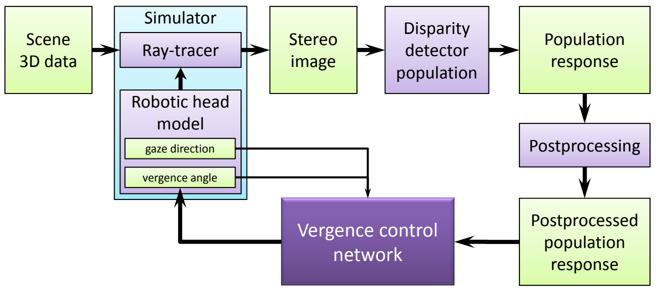


Fig. 1. The scheme of the experimental setup for vergence control model training.

## II. Vergence control model

For the vergence control paradigm modeling we used the framework shown in Fig. 1. This setup consists of the vergence simulator module, the disparity detector population module, the population response postprocessing module and the vergence control network module.

### *A. Vergence simulator*

The vergence simulator consists of a robotic head model (RHM) and a ray-tracing engine. We used the same parameters of the RHM as in [5] (the baseline $b = 70\,\mathrm{mm}$, the focal length $f_0 = 17\,\mathrm{mm}$ and field of view $\approx 20°$). The RHM can be controlled externally by the gaze direction (version) and the vergence angle. Using the RHM and scene description the ray-tracing engine renders left and right views (see Fig. 2), which then are fed to the disparity detector population module. In order to speed up the simulations we decided to use low-resolution images ($41 \times 41$).

### *B. Disparity detectors population module*

Disparity information can be extracted from a stereo image pair by using a distributed cortical architecture [7] that resorts to a population of simple and complex cells. The population is composed of cells sensitive to $N_p \times N_o$ vector disparities $\boldsymbol{\delta} = (\delta_H, \delta_V)$ with $N_p$ magnitude values distributed in the range $[-\Delta, \Delta]$ pixels and along $N_o$ orientations uniformly distributed between 0 and $\pi$ (see Fig. 3). Each simple cell has a binocular receptive field $g_L(x, y) + g_R(x, y)$ defined by a pair of Gabor functions:

$$g(x, y; \psi, \theta) = \exp\left(-\frac{1}{2\sigma^2}(x_\theta^2 + y_\theta^2)\right)\cos(2\pi k_0 x_\theta + \psi) \quad (1)$$

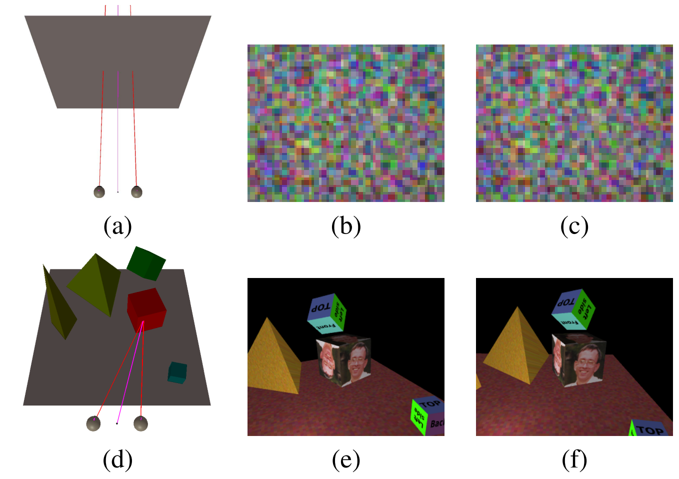


Fig. 2. An example of a simplified (a)/general (d) case synthetic scene used by the vergence simulator to render corresponding left (b)/(e) and right (c)/(f) eye views.

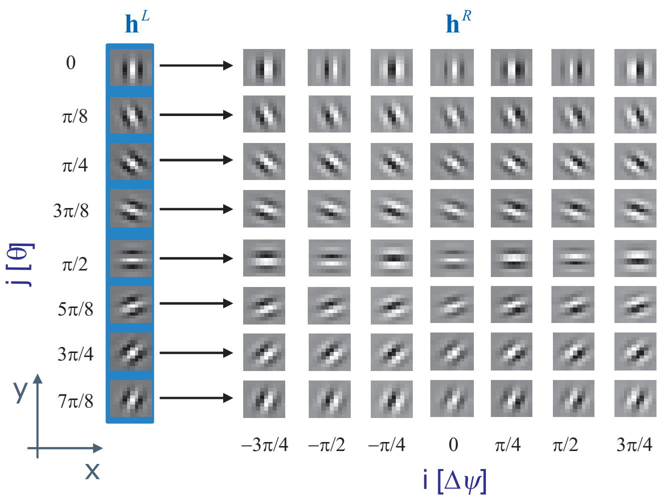


Fig. 3. The population of binocular receptive fields for each retinal location.

positioned in corresponding points $\mathbf{x} = (x, y)$ of the left and the right images, rotated by the same angle $\theta$ with respect to the horizontal axis, and characterized by the same peak frequency $k_0$ and spatial envelope $\sigma$, and by a proper binocular phase shift ($\Delta\psi = \psi_L - \psi_R$), along the rotated axis $x_\theta$, which confers to the cell its specific tuning to a disparity $\delta_{\text{pref}} = \Delta\psi/2\pi k_0$, along the direction orthogonal to $\theta$. Formally, given $I_L(\mathbf{x})$ and $I_R(\mathbf{x})$ the left and the right images and $\boldsymbol{\delta}(\mathbf{x})$ the image disparities so that $I_L(\mathbf{x}) = I_R(\mathbf{x} + \boldsymbol{\delta}(\mathbf{x}))$, for every image position $\mathbf{x}$, the response of a simple cell $r_s$ is given by:

$$r_s(\boldsymbol{\delta}(\mathbf{x}); \theta, \Delta\psi) = \iint (g_L(\mathbf{x}' - \mathbf{x}) I_R(\mathbf{x}' + \boldsymbol{\delta}(\mathbf{x}'))+ \\ + g_R(\mathbf{x}' - \mathbf{x}) I_R(\mathbf{x}')) d\mathbf{x}'. \quad (2)$$

The response of a complex cell $r_c$ is modeled by the sum of the squared response of a quadrature pair of simple cells, and its response is given by [8]:

$$r_c(\boldsymbol{\delta}(\mathbf{x}); \theta, \Delta\psi) = r_s^2(\boldsymbol{\delta}(\mathbf{x}); \theta, \Delta\psi)+ \\ + r_s^2(\boldsymbol{\delta}(\mathbf{x}); \theta, \Delta\psi + \pi/2). \quad (3)$$

For each orientation, the population is, in this way, capable of providing reliable disparity estimates in the range between $-\Delta$ and $\Delta$, where $\Delta = \Delta\psi_{\max}/k_0$ can be defined as the maximum detectable disparity of the population.

In this work we consider only single-scale architecture of the disparity detector population, but the population can be extended to multiscale mode, which is more expensive in computational sense.

### C. Postprocessing module

The response of the population through the postprocessing module reaches the vergence control network (VC-net). The actual gaze direction and actual vergence are used as an additional input to the VC-net.

The postprocessing of the population response was different for the two considered VC-nets. In the case of the linear network, the postprocessing was defined as a 2D pooling over first two (spatial) dimensions of the population response (Section IV).

On the one hand, the pooling operation reduces the amount of data to process, but on the other hand, it has a major drawback as it discards the spatial information about the disparity encoded in the population response. The simulations shows that, in the simplified case, this is still acceptable, but not in general case (Section IV).

In experiments with the convolutional network we do not postprocess population response externally, but let the network to do this in the first two layers (Section V). In this case, the postprocessing module works as an identical operator.

### D. Vergence control module

This module is the main module of the model. The purpose of the module is to convert postprocessed population response together with the actual vergence and the gaze direction into a new vergence angle. Virtually, this module can be represented by any kind of paradigms, but in this work we discuss only a linear network (Section IV) and a convolutional network (Section V).

## III. Methods

In order to train and evaluate any model, one should provide a way to learn model's parameters and at least one method to measure its performance. To this end, we adopted training from examples approach and created a *vergence databases* (see Section III-A) as a source of training/testing examples. The evaluation we propose to do in terms of *distance to the fixation point*, discussed in Section III-B.

### A. Vergence database

The vergence simulator was used also for the creation of a *vergence database*, which has been used for training and testing the vergence control network. The database contains a set of synthetic scenes and a set of samples. Each synthetic scene consists of several simple (plane triangle, cube, tetrahedron etc.) textured objects placed into room-like virtual environment (see Fig. 2). All the textures (real-world images, checkerboard-like images, random noise etc.) we used, were corrupted by 5% Gaussian noise in order to obtain a better response from the population. Samples of the database consist of the *gaze direction*, the *actual vergence angle*, the *stereo pair* (left and right eyes images), the *population response* for the stereo pair and the *desired vergence angle*. The actual vergence angle is a distorted (with Gaussian noise) version of the desired one.

With the database it is easy to prepare training pairs. As the input data vector is constructed from the gaze direction, the actual vergence angle and postprocessed population response are computed. The output consists of only one parameter: the desired vergence angle.

## B. *Vergence angle vs. distance to the fixation point*

Given a robotic head baseline $b$ and a gaze direction vector $\mathbf{g} = (g_x, g_y, g_z)^T$, ($\|\mathbf{g}\| = 1$) it is possible to infer the distance $d$ to the fixation point (from the middle of the head's baseline) using the vergence angle $\alpha$:

$$d = \frac{b}{2}\left(s + \sqrt{s^2+1}\right), \text{ where } s = \frac{1}{\tan\alpha\sqrt{1-g_x^2}} \tag{4}$$

and *vice versa*:

$$\begin{aligned} \alpha &= \arccos\left(\frac{\mathbf{v}_l^T\mathbf{v}_r}{\|\mathbf{v}_l\|\cdot\|\mathbf{v}_r\|}\right), \text{ where} \\ \mathbf{v}_r &= d\cdot\mathbf{g} + (b/2, 0, 0)^T, \text{ and} \\ \mathbf{v}_l &= d\cdot\mathbf{g} - (b/2, 0, 0)^T. \end{aligned} \tag{5}$$

From the equations (4) and (5), one can see that by considering a fixed gaze direction and fixed baseline, the vergence angle is equivalent to the distance to the fixation point (nevertheless they have a nonlinear relationship). We used the deviation of the actual distance to the fixation point from the desired one as an additional measure of vergence performance. From our point of view, this measure is more natural compared to the deviation of the vergence angle.

## IV. Linear network

The first attempt in the modeling of a network for vergence control was done with the simplest possible network consisting of only one linear unit. The simulations have revealed, that even this simple network is able to perform similarly to the model from [5].

## A. *Population response postprocessing for the linear network*

As it has been mentioned above, we have defined the post-processing of the population response for the linear network as 2D pooling over the first two (spatial) dimensions of the population response with a two-dimensional Gaussian kernel $G_\sigma$:

$$P_{ij} = G_\sigma * r_c^{ij}, \tag{6}$$

where $r_c^{ij}$ is population response map for $i$-th orientation and $j$-th phase shift. The kernel $G_\sigma$ has the same size $n_r \times n_c$ as the size of a population response map $r_c^{ij}$, so the result of the convolution is a scalar value $P_{ij}$.

This step drastically reduces the amount of data to process. After pooling, the network has to process only a two-dimensional ($N_o \times N_p$) pooled population response instead of four-dimensional ($n_r \times n_c \times N_o \times N_p$) array, where $N_p$ is the number of phase shifts, $N_o$ is the number of orientations.

## B. *Training*

As it has already been mentioned in the Introduction, we consider two cases for the experiment: a *simplified* and a *general* case. An example of the simplified case is shown in Fig. 2(a-c): the gaze direction of the robotic head is orthogonal to its baseline and the stimulus is a frontoparallel plane, thus, also orthogonal to the gaze direction. The stimulus is allowed to move only in depth. In the general case, all restrictions on the orientation of the gaze, as well as the stimulus position, type and orientation, are dropped. One of the examples is shown in Fig. 2(d-f) with the only difference in the resolution of the rendered images (for the simulation we used much lower resolution).

For each experiment case we have prepared several vergence databases with the number of synthetic scenes ranging from 100 to 1000 and the number of samples from 200 to 4000.

The input vector for the linear VC-net was constructed as a concatenation of the pooled population response (56 values), the gaze direction (2 values) and the actual vergence (1 value), so its dimensionality is 59. The output is a prediction of the vergence angle, which is a scalar value. Due to the linearity of the network, there was no reason to introduce any hidden layers, so the linear VC-net consisted of only one linear unit. This simplest possible vergence control network has only 60 parameters (including bias), which can be learned either directly (using robust linear regression) or iteratively (using gradient descent) from the training database. Not surprisingly, both training approaches produced almost identical solutions on the same training data in the simplified case.

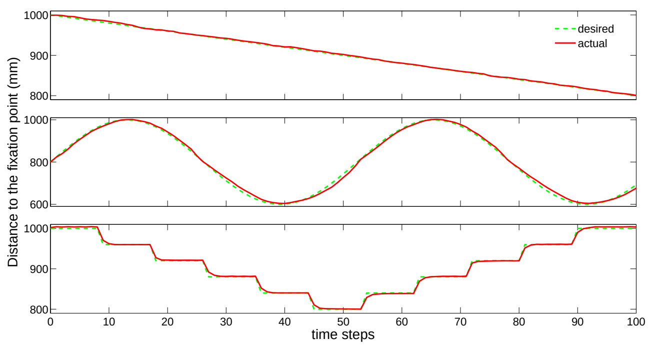
## C. *Evaluation and results*

For the evaluation of the VC-net, we have adopted the methodology described in [5] with the next differences:

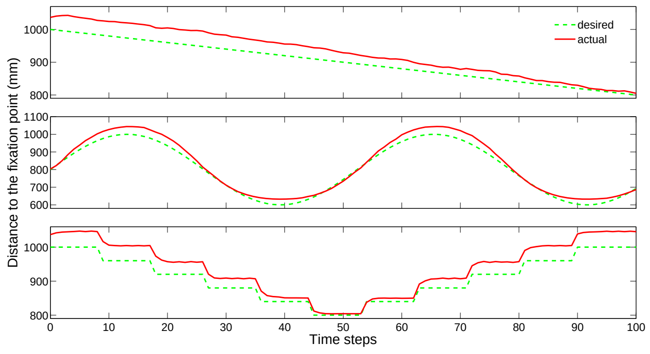
- the stimuli are allowed to move in the direction of the gaze (not only in Z direction),
- the rendered stimuli cover 60-100% of the image area (allowing for depth discontinuities),
- in the general case the stimuli can be not only 2D plane rectangles but also 3D primitives (cubes or tetrahedrons),
- in the general case the stimuli can have arbitrary position and orientation inside the workspace.

The first item is very important for the general case, when the gaze direction is not necessarily parallel to the Z-axis.

Three standard tests (ramp, sinusoid and staircase) were carried out for the simplified as well as the general case. The typical results of the performance, measured in terms of distance to the fixation point, are shown in Fig. 4.



(a) Simplified scenario



(b) General case scenario

Fig. 4. A typical examples of the depth-based performance plots for a linear vergence control network in the simplified (a) and general case (b) scenarios.

The results of the evaluation of the linear VC-network show, that in the simplified setup, it can produce an accurate and robust vergence control using spatially pooled population responses. The relative error (distance-based as well as angular measure) was always less then 1% in all tests of the simplified scenario (see Fig. 4a).

Though, in general, this approach has unpredictable systematic error, which in our tests was up to 7% (see Fig. 4b). The large magnitude of the vergence error of the linear network in the general case can be explained, from our point of view, by the presence of the vertical disparity asymmetric patterns (due to the arbitrary orientation and position of the stimuli) and by the disparity discontinuities (caused by the limited size of the stimuli). In the simplified scenario, the vertical disparity information is symmetrically spread over the spatial dimensions of the population response, and is discarded in the preprocessing stage by spatial pooling. This does not happen in the general case, so the pooled population response is biased by the residual vertical disparity, and linear network, in turn produces a biased vergence control signal.

This situation motivated us to investigate a more complex paradigm for the vergence control, which should be able to recognize particular patterns in the population responses in the general case, and produce a proper vergence control signal. To this purpose, we have chosen a convolutional network [9], [10], [11], as it has proved to be one of the best paradigms for pattern recognition.

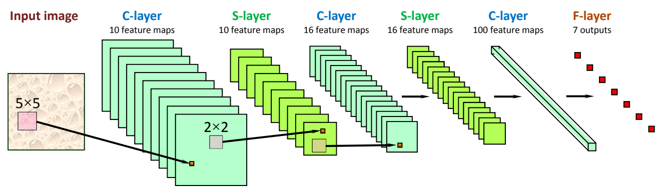


Fig. 5. An example of typical convolutional network.

## V. Convolutional network

The first *convolutional network* (CN) appeared in the work of Fukushima in [9] and was called Neocognitron. The basic architectural ideas behind the CN (*local receptive fields*, *shared weights*, and spatial or temporal *subsampling*) allow such networks to achieve some degree of shift and deformation invariance and, at the same time, reduce the number of training parameters.

Since 1989, Yann LeCun and co-workers have introduced in [12] a series of convolutional networks with the general name *LeNet*, which contrary to the Neocognitron use supervised training. In this case, the major advantage is that the whole network is optimized for the given task, making this approach useable for real-world applications. LeNet have been successfully applied to character recognition nonlinear-dimensionality reduction of image-sets [13] and even to obstacle avoidance in an autonomous robot [14].

A typical convolutional network is a feed-forward network of layers of three types: *convolutional* (C-layer), *subsampling* (S-layer) and *fully-connected* (F-layer). The C-layers and S-layers usually come in pairs and are interleaved, and F-layers come at the end (see Fig. 5). The output of a C-layer is organized as a set of *feature maps*. Each feature map contains the output of a set of neurons with local receptive fields. All neurons in the feature map share the same weights, so the feature map is responsible for a particular local visual feature which is encoded in the weights of these neurons. The computation of a feature map starts with a 2D convolution of the input with a fixed kernel defined by the neuron's weights. A feature map can have inputs from several feature maps of the previous layer. In order to condense the extracted features and make them more invariant with respect to spatial deformations, the C-layer is typically followed by an S-layer which does a local averaging and subsampling. Each neuron in a F-layer just does summation of bias with all weighted inputs and then propagates the sum through a nonlinear transfer function (RBF or sigmoid).

The network is trained in a supervised manner using backpropagation. For the efficient training of large CNs, LeCun and colleagues proposed a number of tricks and a modification of the Levenberg-Marquardt algorithm [15].

### A. Extended convolutional network

For the modeling of CN-based vergence control we have developed our own version of the convolution network in

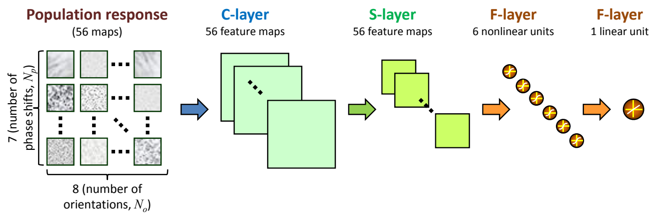


Fig. 6. Convolutional network (and its input) used for the vergence control.

MATLAB. This network can be considered as an extension of LeCun's LeNet because it has the next features:

- any directed acyclic graph can be used for connecting the layers of the network;
- the network can have any number of arbitrarily sized input and output layers;
- the neuron's receptive field (RF) can have an arbitrary stride (step of local RF tiling), which means that in the S-layer, RFs can overlap and in the C-layer the stride can differ from 1;
- any layer or feature map of the network can be switched from trainable to nontrainable (and vice versa) mode even during training;
- new layer type: M-layer.

The M-layer works similarly to the C-layer with the only difference in the subsampling operation $s(\mathbf{x}, a) = a \sum_i x_i$ is replaced by a softmax-like M-operation:

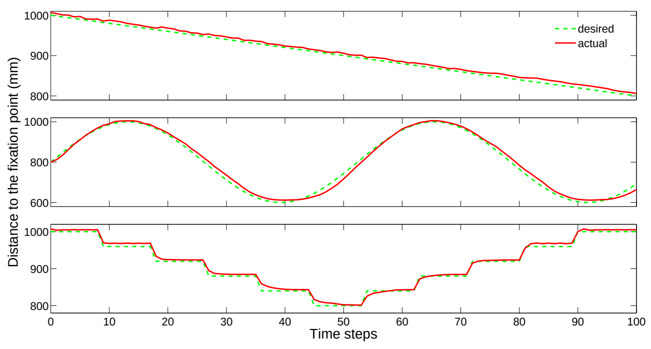
$$m(\mathbf{x}, a) = \frac{\sum_i x_i e^{a x_i}}{\sum_i e^{a x_i}}, \tag{7}$$

where the receptive field is denoted by $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. This function $m(\mathbf{x}, a)$ has been chosen because its properties:

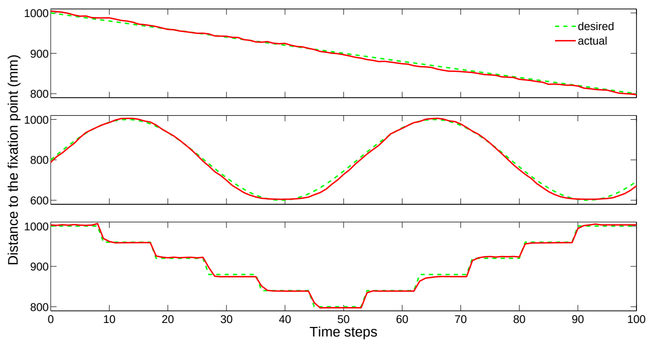
- $m(\mathbf{x}, a) \approx \max\{x_i\}_i$, if $a \gg 1$ (*e.g.* $a = 100$);
- $m(\mathbf{x}, a) \approx \min\{x_i\}_i$, if $a \ll -1$ (*e.g.* $a = -100$);
- $m(\mathbf{x}, a) = \sum_{i=1}^{n} x_i / n$, if $a = 0$.

### B. Convolution network design

The idea behind the use of the convolutional network as a vergence controller consist in an assumption that this powerful network, after proper training, will be able to recognize disparity patterns directly from the population responses, and convert them into a proper vergence signal. The architecture of the convolutional network, used for our experiments, is CSFF and is depicted on Fig. 6. The main challenge in this approach was the amount of data: the population response consists of 56 $(7 \times 8)$ maps of resolution $41 \times 41$ (rendered image resolution), so the input of the network has 94136 $(41 \times 41 \times 8 \times 7)$ components. In order to be able to train the network with such high dimensional input data, we had to reduce the number of the training parameters. The first (convolutional) layer was set as fixed (nontrainable) with Gaussian kernels of size $19 \times 19$ with standard deviation 6. The second (subsampling) layer has also 56 feature maps size of which was set to $3 \times 3$.



(a) Simplified scenario



(b) General case scenario

Fig. 7. A typical examples of the depth-based performance plots for a convolutional vergence control network in the simplified (a) and general case (b) scenarios.

### C. Evaluation and results

For the evaluation of the convolutional network we have used exactly the same tests as for the linear network (see Section IV). The performance was very similar in both scenarios (see Fig. 7). The average relative error (in distance-based measure) for both scenarios is less than 1%. Comparing the performances of two the networks, it is possible to conclude that the convolutional one has a more pronounced inertia with respect to the linear one, but it still is able to handle the general case tasks with an acceptable accuracy and robustness. But, on the other hand, vergence control based on the convolutional network is much more computationally expensive than linear-based.

## VI. Conclusion and future steps

Most of the conventional vergence control models [16], [3], [4], [6], [17], are based on the minimization of the horizontal disparity.

Following an approach similar to [5], we propose to avoid explicit computation of the disparity map and extract the vergence control signal directly from the population response, over the "foveal" region, of a cortical-like network organized as hierarchical arrays of binocular complex cells [7]. A neural network paradigm has been chosen for this type of conversion/extraction procedure. Specifically, an increasing complexity strategy in the learning process is adopted: starting from the simplest one-unit architecture we increase the number of units/layers until an acceptable level of generalization error is reached.

Although the model only resort to a population of neurons in a single scale, we demonstrate that, using a convolutional network, accurate and fast vergence control can be achieved in a closed loop, for different orientations of the gaze.

In the direction of the development of the vergence control networks, our next steps of investigation are the following:

- we can allow the first (C-)layer of the convolutional network to be trained (in a supervised or unsupervised manner);
- we can replace the disparity detector population by additional non-trainable layers of the convolutional network;
- we can specialize the disparity detectors at different levels in the hierarchical network architecture, in order to explore the effect of learning specific coding and decoding strategies for active vergence control and depth vision.

## ACKNOWLEDGMENT

This work has been partially supported by the EU Project FP7-ICT-217077 EYESHOTS[1].

## REFERENCES

[1] I. Ohzawa, R. Freeman, and G. DeAngelis, "Stereoscopic depth discrimination in the visual cortex: Neurons ideally suited as disparity detectors," *Science*, vol. 249, pp. 1037–1041, 1990.

[2] G. Masson, C. Busettini, and F. Miles, "Vergence eye movements in response to binocular disparity without depth perception," *Nature*, vol. 389, pp. 283–286, 1997.

[3] W. Theimer and H. Mallot, "Phase-based vergence control and depth reconstruction using active vision," *CVGIP, Image understanding*, vol. 60, no. 3, pp. 343–358, 1994.

[4] S. Patel, H. Ogmen, and B. Jiang, "Neural network model of short-term horizontal disparity vergence dynamics," *Vision Research*, vol. 37, no. 10, pp. 1383–1399, 1996.

[5] A. Gibaldi, M. Chessa, A. Canessa, S. Sabatini, and F. Solari, "A neural model for binocular vergence control without explicit calculation of disparity," in *Proc. European Symposium on Artificial Neural Networks (ESANN'09)*, Bruges, Belgium, April 2009.

[6] G. Hung, J. Semmlow, and K. Ciuffreda, "A dual-mode dynamic model of the vergence eye movement system," *Trans. on Biomedical Engineering*, vol. 36, no. 11, pp. 1021–1028, 1986.

[7] M. Chessa, S. Sabatini, and F. Solari, "A fast joint bioinspired algorithm for optic flow and two-dimensional disparity estimation," in *Proc. International Conference on Computer Vision Systems (ICVS'09)*, Liege, Belgium, October 2009.

[8] N. Qian, "Computing stereo disparity and motion with known binocular cell properties," *Neural Computation*, vol. 6, no. 3, pp. 390–404, 1994.

[9] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[10] R. Linsker, "From basic network principles to neural architecture: Emergence of orientation columns," *Proceedings of the National Academy of Sciences*, vol. 83, no. 22, pp. 8779–8783, 1986.

[11] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[12] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the trade*, G. Orr and M. K., Eds. Springer, 1998.

[13] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press, 2006.

[14] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," *Advances in neural information processing systems*, vol. 18, 2006.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," vol. 86, no. 11, 1998, pp. 2278–2324.

[16] J. Horng, J. Semmlow, G. Hung, and K. Ciuffreda, "Initial component in disparity vergence: A model-based study," *IEEE Trans. Biomed. Eng.*, vol. 45, pp. 249–257, 1998.

[17] V. Krishnan and L. Stark, "A heuristic model for the human vergence eye movement system," *IEEE Trans. Biomed. Eng.*, vol. 24, pp. 44–49, 1977.

[1]http://www.eyeshots.it