In Proceedings of Seventh International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES'03), Oxford, UK,
LNAI 2774, vol. 2, pp. 139-145, September 2003.

Face Localization in the Neural Abstraction Pyramid

Sven Behnke

International Computer Science Institute
1947 Center St., Berkeley, CA, 94704, USA
behnke@icsi.berkeley.edu

Abstract. One of the major parts in human-computer interface applications, such as face recognition and video-telephony, consists in the localization of a face in an image.
I propose to use hierarchical neural networks with local recurrent connectivity to solve this task, even in presence of complex backgrounds, difficult lighting, and noise. The network is trained using a database of gray-scale still images and manually determined eye coordinates. It is able to produce reliable and accurate eye coordinates for unknown images by iteratively refining an initial solution.
Since the network processes an entire image, no time consuming scanning across positions and scales is needed. Its fast update allows for real-time face tracking.

1 Introduction

To make humans-computer interactions more pleasant, computers must adapt to the users. For adaptation it is necessary to perceive the user. One important step for many adaptive applications, like face recognition, lip reading, and video-telephony is the localization of the user's face in a captured image.

A recent survey on face detection can be found in [6]. Many localization techniques rely on image motion or skin color which are not always available. In [12] multiresolution window scanning in combination with a neural network is used to detect faces in gray-scale static images. Such sequential search techniques are computationally expensive. Many methods preprocess the data intensively to extract facial features and match them with predefined models [7, 9].

Here, I present a method that uses a hierarchical neural network with local recurrent connectivity to localize a face in gray-scale still images. It operates by iteratively refining an initial solution. The network is trained with supervision to perform the task.

The paper is organized as follows. The next section describes the database used for the experiments and the preprocessing procedure. Section 3 introduces the network architecture and details its training. Experimental results are presented in Section 4. The paper concludes with a discussion.

Fig. 1. Some images from the BioID face localization data set.

2 Face Database and Preprocessing

To validate the proposed approach to face localization, the BioID data base [7] is used. It consists of 1521 gray scale images showing 23 individuals in complex office environments (see Fig. 1). The persons differ in gender, age, and skin color. Some of them wear glasses or a beard. Since the face size, position, and view, as well as the facial expression and lighting vary considerably, the dataset is considered challenging. Such real world conditions are the ones that show the limits of current localization techniques. For instance, while the hybrid system described in [7] correctly localizes 98.4% of the XM2VTS database [10] that has been produced under controlled conditions, the same system localizes only 91.8% of the BioID faces.

The BioID images have a size of 384×288 pixels. To reduce border effects, the contrast is lowered towards the sides of the image. To limit the amount of data, the image is subsampled to 48×36, 24×18, and 12×9 pixels as shown in Fig. 2. The data set also contains manually labeled eye positions. A multi-resolutional Gaussian blob is produced for each eye as a target output for training the network.

I divided the BioID data set randomly into 1000 training images and 521 test examples. For the training set, eight image variants are generated by translating them by ±3 pixels in each direction before subsampling.

3 Network Architecture and Training

The preprocessed images are presented to a Neural Abstraction Pyramid [3, 2]. This hierarchical architecture has been motivated by the structure of the hu-

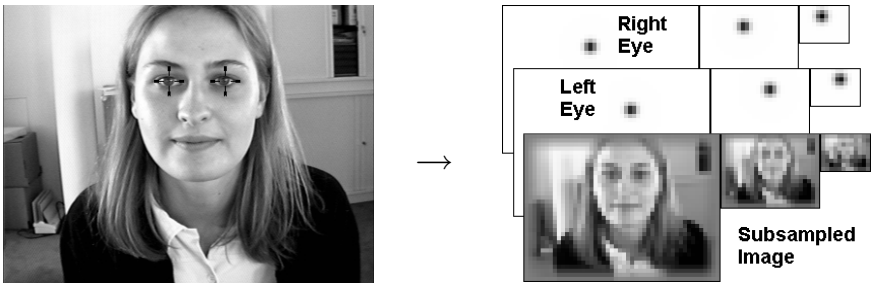

Fig. 2. Preprocessing: Framing and subsampling. Blobs indicate eye positions.

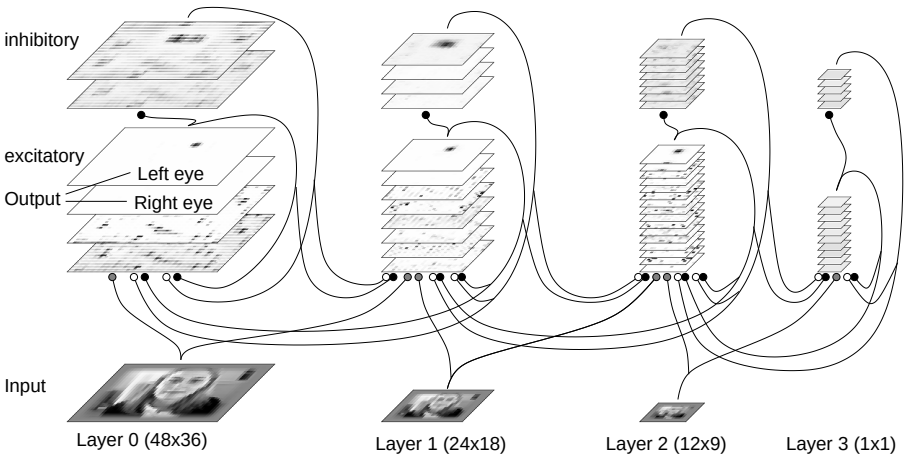

Fig. 3. Sketch of the Neural Abstraction Pyramid network used for face localization.

man visual system. It represents images at different levels of abstraction. High-resolution signal-like representations are present at the bottom of the pyramid, while more abstract representations can be found in higher layers, where resolution decreases and the number of different features increases. These representations are produced by simple processing elements that interact locally. Since the connections form horizontal and vertical feedback loops, the network's activity develops over time. This allows for refinement of an initial solution by incorporating contextual information. This is an efficient way to resolve ambiguities.

As can be seen in Figure 3, the network used here consists of four layers. Each layer contains excitatory and inhibitory feature arrays. Each feature is computed at a 2D-grid of locations by $\sum$-units that share a common weight template. The resolution of the layers decreases from L_0 (48×36) to L_2 (12×9) by a factor of 2 in both dimensions. L_3 has only one cell per feature. The number of feature arrays per layer increases when going from L_0 (4+2) to L_2 (16+8). L_3 contains 10 excitatory and 5 inhibitory features.

The network's connectivity is recurrent and local. The weights of all cells of a feature array are described by a common template. Such weight sharing has been used in the Neocognitron [5] and in convolutional neural networks [8] to reduce the number of free parameters. Each feature cell receives input from only a small window of cells that correspond to similar locations in the layer below (forward weighs), in the same layer (lateral weights) and from the layer above (backward weights). Weights from excitatory cells are non-negative. Weights from inhibitory cells are non-positive. Input weights can have any sign.

The excitatory feature cells of L_0-L_2 receive input from 4×4 windows of the feature arrays one layer below. They look at 5×5 input cells and at a 3×3 neighborhood at the same layer. The backward weights have a window size of 2×2. Connections between L_2 and the topmost L_3 are different. Both, forward- and backward weights have a 12×9 window size. Inhibitory feature cells look only

at 5×5 windows of all excitatory feature arrays at the same layer. Of course, in L_3 this reduces to 1×1.

The update step $(t+1)$ of value $v_{x,y,z,q}$ at position (x,y) in L_z for feature array q is done as follows:

$$v_{x,y,z,q}^{t+1} = \sigma\left[\sum_{j\in\mathcal{L}(i)} \mathcal{W}(j)\, v^{t}_{\mathcal{X}(j,x),\mathcal{Y}(j,y),\mathcal{Z}(j,z),\mathcal{Q}(j)} + \mathcal{B}(i)\right]. \tag{1}$$

The template $i = \mathcal{T}(z,q)$ is associated with feature array q at layer z. $\mathcal{L}(i)$ is its set of links and $\mathcal{B}(i)$ is the template bias. $(\mathcal{X}(j,x), \mathcal{Y}(j,y), \mathcal{Z}(j,z), \mathcal{Q}(j))$ describe location and feature array of the input for link j, and $\mathcal{W}(j)$ is its weight. The output function $\sigma(x) = \mathsf{ln}(1+e^{\beta x})/\beta$ used here is a smooth approximation to the rectifying function $\mathsf{max}(0,x)$. In addition, a start value $\mathcal{V}^0(i)$ for uniform initialization at $t=0$ is needed for each template.

Training recurrent networks is difficult due to the non-linear dynamics of the system. The backpropagation through time algorithm (BPTT) [13] unfolds the network in time and applies backpropagation to compute the gradient of an error function. For face localization, I present a static input $\mathbf{x}_k$ to the network and train it to quickly produce the desired output $\mathbf{y}_k$, where $k = 1 \ldots K$ is the example index. The network is updated for a fixed number $T = 10$ of iterations. The output error δ_k^t, the difference between the activity of the output cells $\mathbf{v}_k^t$ and the desired output $\mathbf{y}_k$, is not only computed at the end of the sequence, but after every update step. In the error function the squared differences are weighted progressively, as the number of iterations t increases:

$$E = \sum_{k=1}^{K}\sum_{t=1}^{T} t\ \|\mathbf{y}_k - \mathbf{v}_k^t\|^2. \tag{2}$$

Minimizing the error function with gradient descent faces the problem that the gradient in recurrent networks either vanishes or grows exponentially in time depending on the magnitude of gains in loops [4]. Hence, it is very difficult to determine a learning constant that allows for both stability and fast convergence. For that reason, I decided to employ the RPROP algorithm [11], that maintains a learning rate for each weight and uses only the sign of the gradient to determine the weight change. Not only the weights are modified in this way, but the biases and start values are adapted as well. To accelerate the training, I initially worked with randomly chosen subsets of the training set, as described in [1].

4 Experimental Results

Figure 4 shows the development of the trained network's output when the test image from Fig. 2 is presented as input. One can see that the blobs signaling the eye locations develop in a top-down fashion. After the first iteration they appear only in the lowest resolution. This coarse localization is used to bias the development of blobs in lower layers. After five iterations, the network's

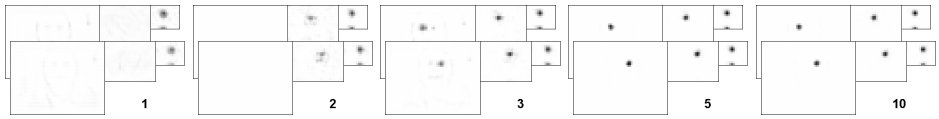

Fig. 4. Recall. Shown are the activities of the network's output over time.

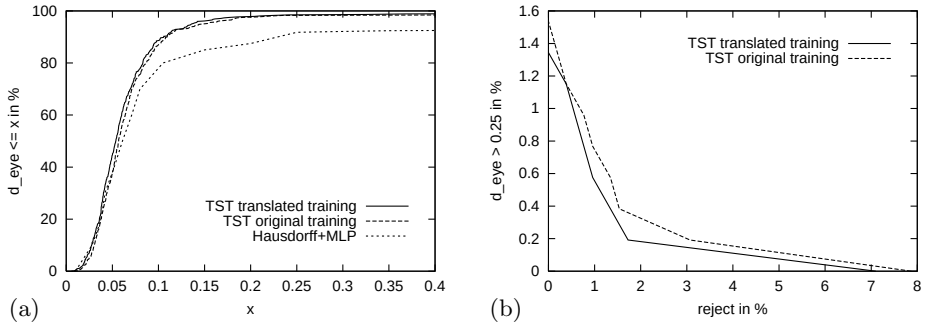

Fig. 5. Localization performance: (a) test examples with small d_{eye} for pyramid trained with the original/translated images and for a hybrid system (Hausdorff+MLP)[7]; (b) rejecting the least confident examples lowers the number of mislocalizations.

output is close to the desired one. It does not change significantly during the next iterations that take 22ms each on a P4 1.7GHz PC.

The generation of stable blobs is the typical behavior of the network. To evaluate its performance, one has to estimate eye coordinates from the blobs and to compute a quality measure by comparison with the given coordinates. Each eye position is estimated separately by finding the feature cell with the highest activity in the corresponding high resolution output array. In a 7×7 window around it, the cells belonging to the blob are segmented by comparing their activity with a threshold that increases with greater distance from the center. The weighted mean of the segmented cells is the estimated eye position. After transforming these eye positions into the original coordinate system, a scale independent relative error measure is computed as suggested in [7]:

$$d_{eye} = \mathsf{max}(d_l, d_r)/||C_l - C_r||, \tag{3}$$

where d_l and d_r are the distances of the estimated eye positions to the given coordinates C_l and C_r. A relative distance $d_{eye} < 0.25$ is considered a successful localization, since $d_{eye} = 0.25$ corresponds approximately to the half width of an eye.

All training examples have been localized successfully. The performance on the test set is similar. Figure 5(a) shows the network's test set localization performance when trained with the original/translated images in comparison to the data taken from [7] (Hausdorff+MLP). Only 1.5%/1.3% of the test examples have not been localized accurately enough. Compare this to the 8.2% mislocalizations of the reference system.

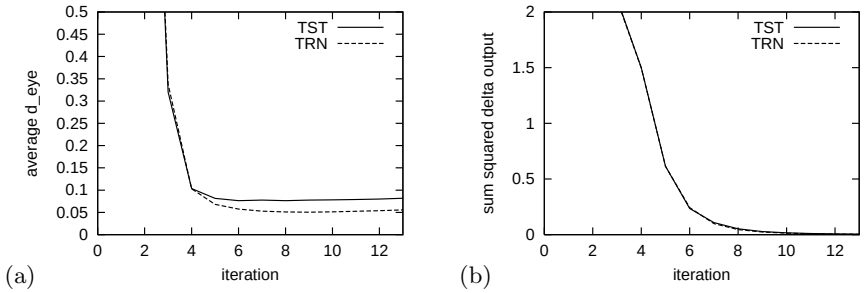

Fig. 6. Performance over time: (a) average distance d_{eye}; (b) output changes.

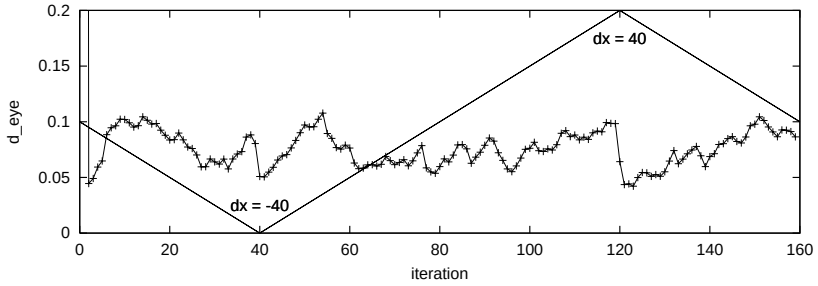

Fig. 7. Tracking of a horizontally moving input (see Fig. 2 for original).

A detailed analysis of the network's output for the mislocalizations showed that in these cases the output deviates from the one-blob-per-eye pattern. It can happen that no blob or that several blobs are present for an eye. By comparing the activity of a segmented blob to a threshold and to the total activity a confidence measure is computed for each eye. Both are combined to a single confidence that is compared to a reject threshold.

In Figure 5(b) one can see that rejecting the least confident test examples lowers the number of mislocalizations rapidly. When rejecting 3.1%/1.7% of the images, only one mislocalization is left. The average localization error of the accepted examples is $d_{eye} = 0.06$. That is well within the area of the iris and corresponds to the accuracy of the given coordinates.

Figure 6 illustrates the network's performance over time when trained with the original images. The average error d_{eye} drops rapidly within the first five iterations and stays low afterwards. The changes in the network's output are large during the first iterations and decrease even when updated longer than the ten steps it has been trained for.

The network is also able to track a moving face. Fig. 7 shows the localization error as the example from Fig. 2 is translated horizontally by ± 40 pixels. After the first iterations, the localization distance stays low at a value of about 0.075. The blobs indicating the eye positions follow the movement of the image with a small delay. They catch up at iterations 40 and 120, when the direction of movement is reversed.

5 Conclusions

In this paper, I presented an approach to face localization that is based on a hierarchical neural network with local recurrent connectivity. The network is trained to solve this task even in the presence of complex backgrounds, difficult lighting, and noise through iterative refinement. Since it processes an entire image in parallel, it does not need any time-consuming scanning over positions and scales.

The network's performance was evaluated on the BioID data set. It compares favorably to a hybrid reference system that uses a Hausdorff shape matching approach in combination to a multi layer perceptron. Adding slightly translated variants to the training set helped generalization. One could extend this approach by using a larger variety of transformations, including rotation, scaling, changes in lighting, and noise.

The proposed method is not limited to gray-scale images. The extension to color is straight forward. Since the network works iteratively and one iteration takes only a few milliseconds, it is also possible to use it for real-time face tracking.

References

1. S. Behnke. Learning iterative image reconstruction in the Neural Abstraction Pyramid. *I.J. Computational Intelligence and Applications*, 1(4):427–438, 2001.
2. Sven Behnke. *Hierarchical Neural Networks for Image Interpretation.* LNCS/LNAI. Springer, to appear 2003.
3. Sven Behnke and Raúl Rojas. Neural Abstraction Pyramid: A hierarchical image understanding architecture. In *Proceedings IJCNN'98*, pages 820–825, 1998.
4. Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Tr. on Neural Networks*, 5(2):157–166, 1994.
5. K. Fukushima, S. Miyake, and T. Ito. Neocognitron: A neural network model for visual pattern recognition. *IEEE Trans. SMC*, 13:826–834, 1983.
6. Erik Hjelmas and Boon Kee Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83:236–274, 2001.
7. O. Jesorsky, K.J. Kirchberg, and R.W. Frischholz. Robust face detection using the Hausdorff distance. In *Int. Conf. on Audio- and Video-based Biometric Person Authentication,* p. 90-95, 2001.
8. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of IEEE*, 86(11):2278–2324, 1998.
9. Dario Maio and Davide Maltoni. Real-time face localization on gray-scale static images. *Pattern Recognition*, 33:1525–1539, 2000.
10. K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre. XM2VTSDB: The extended M2VTS database. In *Int. Conf. on Audio and Video-based Biometric Person Authentication,* p. 72-77, 1999.
11. M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of ICNN'93*, pages 586–591, 1993.
12. Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network based face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20:23–38, 1998.
13. R.J. Williams and J. Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):491–501, 1990.