Face Detection with Neural Networks

Jacob H. Ström
Department of Electrical and Computer Engineering
University of California, San Diego
San Diego, California

Abstract

A method for finding faces in images is presented. The image is first wavelet transformed and then each pixel in the lowest subband is classified as being "part-of-the-face" or "not-part-of-the-face". A neural network is trained to do the classification with information from the lowest 4 subbands from five training images. The output of the network is then postprocessed with morphological filters. The algorithm is evaluated with a verification set of images that were not part of the training set. A block misclassification rate of 24% is achieved on an average.

Keywords: face detection, pattern recognition, neural networks, wavelet, subband decomposition, region of interest, image analysis, face identification

1 Introduction

Many applications would benefit from rapid and robust face detection. In video compression for video telephony for instance, face detection can be used to find the important regions in the image and assign more bits to encode them, yielding a higher perceived quality. Face detection can also be used as a preprocessor for face recognition algorithms. Several different approaches have been made. In [Bedini95], Bedini et al. use snakes to find the convex hull of the head. In [Rowley95], Rowley et al. use a neural network that inputs a neighborhood of a pixel and outputs whether or not this is the center of a face. Different kinds of pre- and postprocessing are made to boost network performance and remove false targets respectively. The method proposed in this paper is also based on a neural network, but in a different way. Instead of outputting whether or not a pixel is the center of a face, the output states whether or not a block of pixels is part of a face. The difference is apparent in figure 1. With an algorithm that finds the center of faces, noise in the output appear as multiple centers. A postprocessing stage is needed to makes sure that a single head only gets detected once. In a block based method the noise appears as flipped blocks. Here morphological filters may be used to remove isolated blocks. The paper is

Figure 1: Left: Output of an algorithm that tries to find the center of the face (the ellipse). Right: Our algorithm lets each block of pixels determine whether it is part of the face or not.

organized as follows: A detailed description of the system is found in section 2. In section 3 the implementation is examined, the results are presented in section 4 and conclusions and future work are discussed in section 5.

2 Description of the System

The system operates in three stages. First, a wavelet transform is applied. This lowers the resolution so that the number of inputs to the network is reduced. It also has the effect of removing mean inensity in all but the lowest subband. The 128 by 128 image is decomposed three times, so that each pixel in the lowest subband corresponds to a block of 8 by 8 pixels in the original image. For each pixel in the lowest subband, apart from the border pixels, a feature vector $\bar{x}_k$ is now created. The vector consists of the 3x3 neighborhood of the pixel[1], the pixels corresponing to the same area from the following 3 subbands, and the x- and y- coordinates of the pixel. This is depicted in figure 2. The

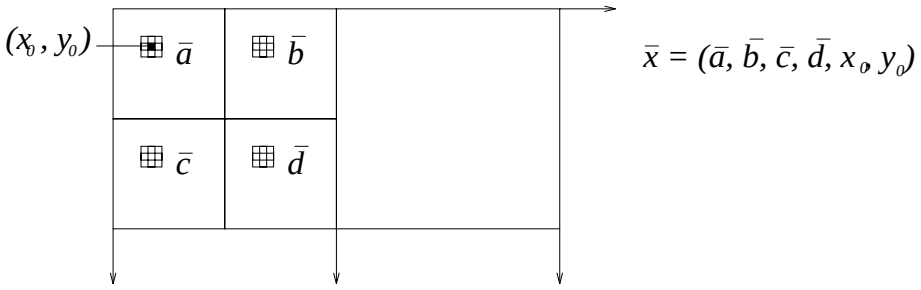

Figure 2: The feature vector $\bar{x}$ consists of 9 pixels from the lowest subband ($\bar{a}$), and 9 each from the following 3 subbands ($\bar{b}, \bar{c}, \bar{d}$) and the pixel coordinates x_0, y_0.

[1] The pixel itself and it's 8-connected neighbors

feature vector is fed into a feed forward neural network, that decides whether or not the pixel is part of a face. The third stage is processing the output image with morphological filters. Before this stage all processing have been solely local. The network has no information whether adjacent blocks are set or not. In reality there is a strong correlation between the blocks. An isolated block indicating face, for instance, would almost never occur in a real image. Therefore, isolated blocks are removed and blocks are also grouped together with a closing operations.

3 Implementation

The system was implemented in Matlab. The training set consisted of 5 images, each 128 by 128 pixels in 256 grayscales. For each image, a binary face map image was constructed by hand, with a 1 where the image contained a face and 0 elsewhere. The mask image was then tiled into 8x8 blocks, each block corresponding to a single pixel in the lowest subband. Each pixel in the lowest subband generated a feature vector $\bar{x}_k$ and a training output vector $\bar{y}_k$. The output vector $\bar{y}_k$ was set to (1,0) if any pixel in the 8x8 block in the face map was 1, and to (0,1) otherwise. A training set $T = (\bar{x}_k, \bar{y}_k)_{k=0}^{N}$ was then constructed by scanning through all five images and randomizing the vectors afterwards. When using the neural network, an output vector of $\bar{y}' = (y_1', y_2')$ classifies the block as part of the face if $y_1' > y_2'$ and as not part of the face otherwise.

3.1 Training Set Problems

Since the input data was biased towards the non-face category (most of the image was not covered by a face), the network tended to always choose the "non-face" category. To avoid this problem the feature vectors containing a face were duplicated, until a the training set contained 50% "face" feature vectors. Another problem with the original training set (obtained via ftp from an MIT database) was that the background was almost the same on all images. Thus the network was trained on finding "background"/"not-background" instead of "face"/"not-face". This problem was solved by producing our own set of images (of people from the Visual Computer lab) where the background was allowed to vary a lot. Still there was a problem that the background tended to be brighter than the face, which triggered the network to choose dark pixels as face-pixels. This was problem was diminished (but not entirely coped with) by extracting bright background images from the training set.

3.2 Training the Neural Net

The first training method used was backpropagation. The problem here was to find a proper learning rate. If set too big, the network never learned, if

too small, the training took too long. Since backpropagation with momentum automatically adjusts the learning rate, this training method was used instead with far more satisfying results. Next, a proper number of hidden units was to be chosen. Different sizes were tried, ranging from 25 to 200. A number of 75 turned out to give optimum performance on the verification set. Another question was how many epochs the network should be trained. The performance was evaluated using ten images with known face-maps. The total error for each image, $e(i)$ was the percentage of misclassifications. The total error rate for the system was the root mean squared error, RMSE

$$RMSE = \sqrt{\frac{1}{10}\sum_{i=1}^{10} e^2(i)}$$

which is similar to the mean of $e(i)$ but penalizes large errors more. Figure 3 shows a typical fluctuation of the RMSE over different number of traning epochs.

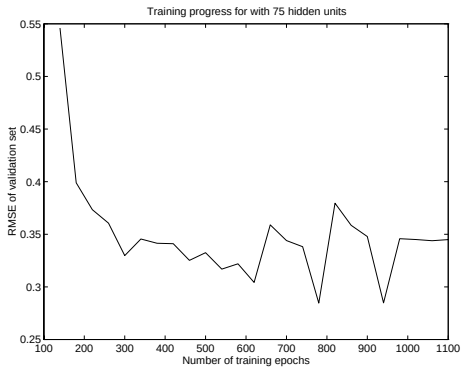

Figure 3: The RMSE over the validation set as a function of number of epochs used in training. Note that after the original drop the curve levels out fluctuating around 30%.

A value around 1000 seems reasonable, but since the curve is fluctuating so much it can be advantagous to save the weight matrices every time the RMSE hits an all time low.

4 Results

Figure 4 shows the behaviour of the system for an image that is not part of the training set. The left-most image is a photo of the author. The true segmentation (in 8x8 blocks) is shown directly next to it. The third image is the output of the neural network. After cleaning (removing isolated pixels and performing

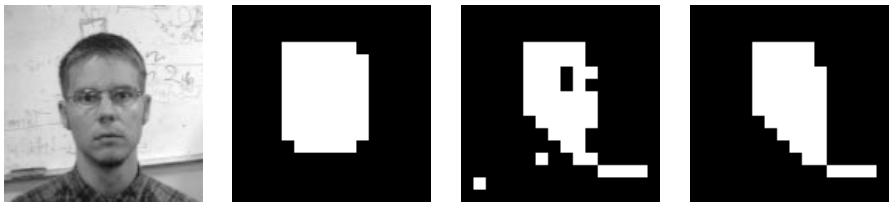

Figure 4: From left: Original image (not in training sequence), true face-map, network output and final result after cleaning. Error rate = 11%.

a closing operation), we get the resulting output from the system in the rightmost image. 22 blocks are misclassified and they make up a total of 11% of the blocks. Figure 5 shows how important the postprocessing stage is. The cleaning

Figure 5: From left: Original image (not in training sequence), true face-map, network output and final result after cleaning. Note the big impact of the cleaning filter, from 35% errors to 20%.

operation removes the two isolated blocks to the left and the closing operator fills in the parts of the face that are hollow. The error count drops from 68 to 39 blocks, i.e. from 35% to 20%. In figure 6 the gain of the postprocessing stage is shown. The dashed line is with postprocessing, and the solid line is without. The constant lines are the average error of all 10 images, it goes down from 29% to 24%.

5 Conclusions and Future Work

A system finding faces in images was designed and implemented. The image is first decomposed with wavelet filters, then small blocks of pixels are fed into a neural net, from which the output is postprocessed with morpholigcal filters. An error rate of 24% on an average is obtained. Possible improvements might be to increase the number of training examples (5 is fairly small), including coefficients form higher subbands in the feature vector, and taking advantage of color information if applied on color images. If video is considered information where

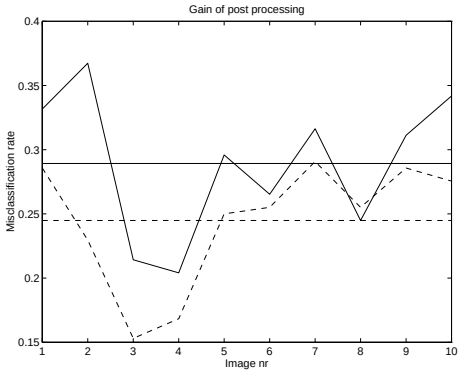

Figure 6: The error rate plotted for the ten different validation images. Dashed line is the error with postprocessing, solid line is the error without. The constant lines are the mean errors.

the head was in the last frame should be exploited to improve performance. Further and more intelligent postprocessing might also be useful. Variations of the system can be built that tracks only the eyes or the mouth. Performance might then go up since these features are more local. Finally, performing the operation on multiple scales and averaging might be useful.

References

[Bedini95] Bedini, G., Favalli, L., Mecocci, A., and others. (1995) Intelligent image interpretation for high-compression high-quality sequence coding. European Transactions on Telecommunications, May-June 1995, vol. 6, (no.3):255-265.

[Rowley95] Rowley, H.A., Baluja, S., Kanade, T. (1995) Human Face Detection in Visual Scenes School of Computer Science, Carnegie Mellon University, Pitsburg, PA 15213, November 1995, CMU-CS-95-158R.