Appears in *Computer Vision and Pattern Recognition*, 1996.

Neural Network-Based Face Detection*

Henry A. Rowley har@cs.cmu.edu
Shumeet Baluja baluja@cs.cmu.edu
Takeo Kanade tk@cs.cmu.edu
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

We present a neural network-based face detection system. A retinally connected neural network examines small windows of an image, and decides whether each window contains a face. The system arbitrates between multiple networks to improve performance over a single network. We use a bootstrap algorithm for training the networks, which adds false detections into the training set as training progresses. This eliminates the difficult task of manually selecting non-face training examples, which must be chosen to span the entire space of non-face images. Comparisons with other state-of-the-art face detection systems are presented; our system has better performance in terms of detection and false-positive rates.

1 Introduction

In this paper, we present a neural network-based algorithm to detect frontal views of faces in gray-scale images[1]. The algorithms and training methods are general, and can be applied to other views of faces, as well as to similar object and pattern recognition problems.

Training a neural network for the face detection task is challenging because of the difficulty in characterizing prototypical "non-face" images. Unlike face *recognition*, in which the classes to be discriminated are different faces, the two classes to be discriminated in face *detection* are "images containing faces" and "images not containing faces". It is easy to get a representative sample of images which contain faces, but it is much harder to get a representative sample of those which do not. The size of the training set for the second class can grow very quickly.

We avoid the problem of using a huge training set for non-faces by selectively adding images to the training set as training progresses [Sung and Poggio, 1994]. Detailed descriptions of this training method, along with the network architecture are given in Section 2. In Section 3, the performance of the system is examined. We find that the system is able to detect 90.5% of the faces over a test set of 130 images, with an acceptable number of false positives. Section 4 compares this system with similar systems. Conclusions and directions for future research are presented in Section 5.

2 Description of the system

Our system operates in two stages: it first applies a set of neural network-based filters to an image, and then arbitrates the filter outputs. The filters examine each location in the image at several scales, looking for locations that might contain a face. The arbitrator then merges detections from individual filters and eliminates overlapping detections.

2.1 Stage one: A neural network-based filter

The first component of our system is a filter that receives as input a 20x20 pixel region of the image, and generates an output ranging from 1 to -1, signifying the presence or absence of a face, respectively. To detect faces anywhere in the input, the filter is applied at every location in the image. To detect faces larger than the window size, the input image is repeatedly subsampled by a factor of 1.2, and the filter is applied at each scale.

The filtering algorithm is shown in Figure 1. First, a preprocessing step, adapted from [Sung and Poggio, 1994], is applied to a window of the image. The window is then passed through a neural network, which decides whether the window contains a face. The preprocessing first attempts to equalize the intensity values across the window. We fit a function which varies linearly across the window to the intensity values in an oval region inside the window. Pixels outside the oval may represent the background, so those intensity values are ignored in computing the lighting

*This work was partially supported by a grant from Siemens Corporate Research, Inc., and by the Department of the Army, Army Research Office under grant number DAAH04-94-G-0006. This work was started while Shumeet Baluja was supported by a National Science Foundation Graduate Fellowship. He is currently supported by a graduate student fellowship from the National Aeronautics and Space Administration, administered by the Lyndon B. Johnson Space Center. The conclusions in this document are those of the authors, and do not necessarily represent the policies of the sponsoring agencies.

[1] A demonstration at http://www.cs.cmu.edu/~har/faces.html allows anyone to submit images for processing by the face detector, and displays the detection results for pictures submitted by others.

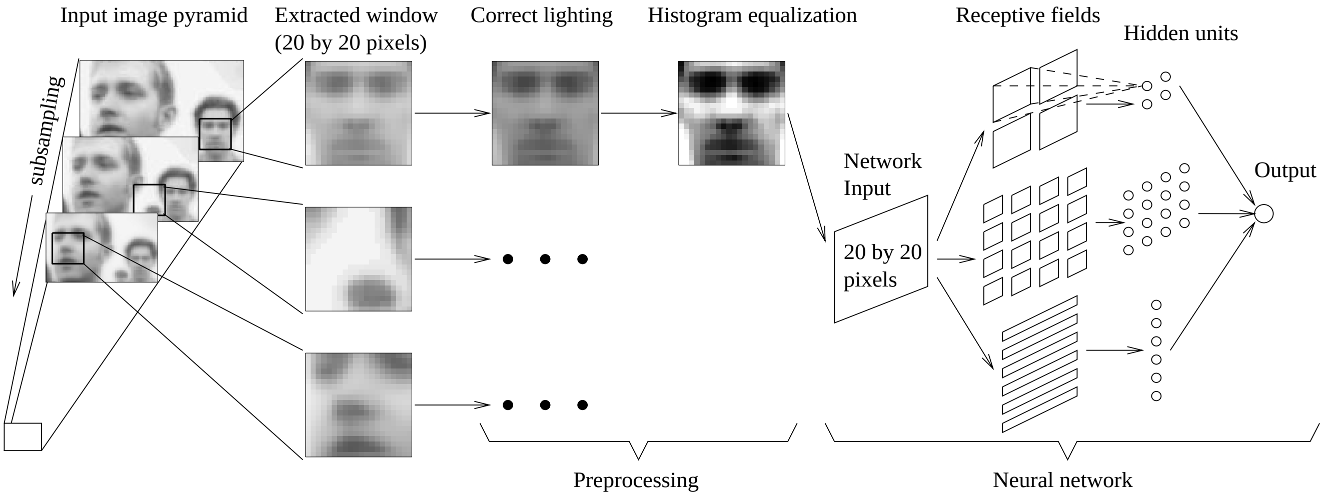

Figure 1: The basic algorithm used for face detection.

variation across the face. The linear function will approximate the overall brightness of each part of the window, and can be subtracted from the window to compensate for a variety of lighting conditions. Then histogram equalization is performed, which non-linearly maps the intensity values to expand the range of intensities in the window. The histogram is computed for pixels inside an oval region in the window. This compensates for differences in camera input gains, and improves the contrast in some cases.

The preprocessed window is then passed through a neural network. The network has retinal connections to its input layer; the receptive fields of hidden units are shown in Figure 1. There are three types of hidden units: 4 which look at 10x10 pixel subregions, 16 which look at 5x5 pixel subregions, and 6 which look at overlapping 20x5 pixel horizontal stripes of pixels. Each of these types was chosen to allow the hidden units to represent localized features that might be important for face detection. Although the figure shows a single hidden unit for each subregion of the input, these units can be replicated. For the experiments which are described later, we use networks with two and three sets of these hidden units. Similar input connection patterns are commonly used in speech and character recognition tasks [Waibel *et al.*, 1989, Le Cun *et al.*, 1989]. The network has a single, real-valued output, which indicates whether or not the window contains a face.

To train the neural network used in stage one to serve as an accurate filter, a large number of face and non-face images are needed. Nearly 1050 face examples were gathered from face databases at CMU and Harvard[2]. The images contained faces of various sizes, orientations, positions, and intensities. The eyes and the center of the upper lip of each face were located manually, and these points were used to normalize each face to the same scale, orientation, and position, as follows:

1. Rotate image so both eyes appear on a horizontal line.
2. Scale image so the distance from the point between the eyes to the upper lip is 12 pixels.
3. Extract a 20x20 pixel region, centered 1 pixel above the point between the eyes and the upper lip.

In the training set, 15 face examples are generated from each original image, by randomly rotating the images (about their center points) up to 10°, scaling between 90% and 110%, translating up to half a pixel, and mirroring. Each 20x20 window in the set is then preprocessed (by applying lighting correction and histogram equalization). The randomization gives the filter invariance to translations of less than a pixel and scalings of ±10%. Larger changes in translation and scale are dealt with by applying the filter at every pixel position in an image pyramid, in which the images are scaled by factors of 1.2.

Practically any image can serve as a non-face example because the space of non-face images is much larger than the space of face images. However, collecting small yet a "representative" set of non-faces is difficult. Instead of collecting the images before training is started, the images are collected during training in the following manner, adapted from [Sung and Poggio, 1994]:

1. Create an initial set of non-face images by generating 1000 images with random pixel intensities. Apply the preprocessing steps to each of these images.
2. Train the neural network to produce an output of 1 for the face examples, and -1 for the non-face examples. The training algorithm is standard error backpropogation. On the first iteration of this loop, the network's weights are initially random. After the first iteration, we use the weights computed by training in the previous iteration as the starting point for training.

[2] Dr. Woodward Yang at Harvard provided over 400 mug-shot images.

3. Run the system on an image of scenery *which contains no faces*. Collect subimages in which the network incorrectly identifies a face (an output activation > 0).
4. Select up to 250 of these subimages at random, apply the preprocessing steps, and add them into the training set as negative examples. Go to step 2.

We used 120 images of scenery for collecting negative examples in this bootstrap manner. A typical training run selects approximately 8000 non-face images from the 146,212,178 subimages that are available at all locations and scales in the training scenery images.

2.2 Stage two: Merging overlapping detections and arbitration

The system described so far, using a single neural network, will have some false detections. Below we mention some techniques to reduce these errors; for more details the reader is referred to [Rowley *et al.*, 1995].

Because of a small amount of position and scale invariance in the filter, real faces are often detected at multiple nearby positions and scales, while false detections only appear at a single position. By setting a minimum threshold on the number of detections, many false detections can be eliminated. A second heuristic arises from the fact that faces rarely overlap in images. If one detection overlaps with another, the detection with lower confidence can be removed.

During training, identical networks with different random initial weights will select different sets of negative examples, develop different biases and hence make different mistakes. We can exploit this by arbitrating among the outputs of multiple networks, for instance signalling a detection only when two networks agree that there is a face.

3 Experimental results

The system was tested on three large sets of images, which are completely distinct from the training sets. Test Set A was collected at CMU, and consists of 42 scanned photographs, newspaper pictures, images collected from the World Wide Web, and digitized television pictures. These images contain 169 frontal views of faces, and require the networks to examine 22,053,124 20x20 pixel windows. Test Set B consists of 23 images containing 155 faces (9,678,084 windows); it was used in [Sung and Poggio, 1994] to measure the accuracy of their system. Test Set C is similar to Test Set A, but contains some images with more complex backgrounds and without any faces, to more accurately measure the false detection rate. It contains 65 images, 183 faces, and 51,368,003 windows.[3]

Rather than providing a binary output, the neural network filters produce real values between 1 and -1, indicating whether or not the input contains a face, respectively. A threshold value of zero is used during *training* to select the negative examples (if the network outputs a value of greater than zero for any input from a scenery image, it is considered a mistake). Although this value is intuitively reasonable, by changing this value during *testing*, we can vary how conservative the system is. We measured the detection and false positive rates as the threshold was varied from 1 to -1. At a threshold of 1, the false detection rate is zero, but no faces are detected. As the threshold is decreased, the number of correct detections will increase, but so will the number of false detections. This tradeoff is illustrated in Figure 2, which shows the detection rate plotted against the number of false positives as the threshold is varied, for two independently trained networks. Since the zero threshold locations are close to the "knees" of the curves, as can be seen from the figure, we used a zero threshold value throughout testing.

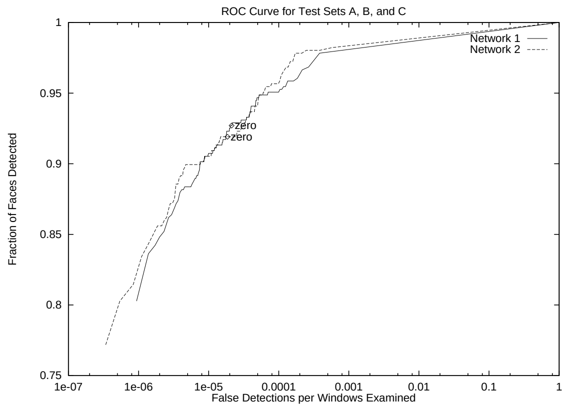

Figure 2: The detection rate plotted against false positives as the detection threshold is varied from -1 to 1, for two networks. The performance was measured over all images in Test Sets A, B, and C. Network 1 uses two sets of the hidden units illustrated in Figure 1, while Network 2 uses three sets. The points labelled "zero" are the zero threshold points used for all other experiments.

Table 1 shows the performance for four networks working alone, the effect of overlap elimination and collapsing multiple detections, and the results of using ANDing, ORing, voting, and neural network arbitration. Networks 3 and 4 are identical to Networks 1 and 2, respectively, except that the negative example images were presented in a different order during training. The results for ANDing and ORing networks were based on Networks 1 and 2, while voting was based on Networks 1, 2, and 3. The table shows the percentage of faces correctly detected, and the number of false detections over the combination of Test Sets A, B, and C. [Rowley *et al.*, 1995] gives a breakdown of the performance of each of these system for each of the three test

[3] The test sets are available at http://www.cs.cmu.edu/~har/faces.html.

Table 1: Combined detection and error rates for Test Sets A, B, and C

Type	System	Missed faces	Detect rate	False detects	False detect rate
	0) Ideal System	0/507	100.0%	0	0/83099211
Single network, no heuristics	1) Network 1 (52 hidden units, 2905 connections)	37	92.7%	1768	1/47002
	2) Network 2 (78 hidden units, 4357 connections)	41	91.9%	1546	1/53751
	3) Network 3 (52 hidden units, 2905 connections)	44	91.3%	2176	1/38189
	4) Network 4 (78 hidden units, 4357 connections)	37	92.7%	2508	1/33134
Single network, with heuristics	5) Network 1 → threshold(2,1) → overlap elimination	46	90.9%	844	1/98459
	6) Network 2 → threshold(2,1) → overlap elimination	53	89.5%	719	1/115576
	7) Network 3 → threshold(2,1) → overlap elimination	53	89.5%	975	1/85230
	8) Network 4 → threshold(2,1) → overlap elimination	47	90.7%	1052	1/78992
Arbitrating among two networks	9) Networks 1 and 2 → AND(0)	66	87.0%	209	1/397604
	10) Networks 1 and 2 → AND(0) → threshold(2,3) → overlap elimination	107	78.9%	8	1/10387401
	11) Networks 1 and 2 → threshold(2,2) → overlap elimination → AND(2)	74	85.4%	63	1/1319035
	12) Networks 1 and 2 → thresh(2,2) → overlap → OR(2) → thresh(2,1) → overlap	48	90.5%	362	1/229556
Three nets	13) Networks 1, 2, 3 → voting(0) → overlap elimination	53	89.5%	195	1/426150

threshold(distance,threshold): Only accept a detection if there are at least *threshold* detections within a cube (extending along x, y, and scale) in the detection pyramid surrounding the detection. The size of the cube is determined by *distance*, which is the number of a pixels from the center of the cube to its edge (in either position or scale).

overlap elimination: A set of detections may erroneously indicate that some faces overlap with one another. This heuristic examines detections in order (from those having the most votes within a small neighborhood to those having the least), and removing conflicting overlaps as it goes.

voting(distance), AND(distance), OR(distance): These heuristics are used for arbitrating among multiple networks. They take a *distance* parameter, similar to that used by the threshold heuristic, which indicates how close detections from individual networks must be to one another to be counted as occuring at the same location and scale. A *distance* of zero indicates that the detections must occur at precisely the same location and scale. Voting requires two out of three networks to detect a face, AND requires two out of two, and OR requires one out of two to signal a detection.

sets, as well as the performance of systems using neural networks to arbitration among multiple detection networks. The parameters required for each arbitration method are described below the table.

Systems 1 through 4 show the raw performance of the networks. Systems 5 through 8 use the same networks, but include the thresholding and overlap elimination steps which decrease the number of false detections significantly, at the expense of a small decrease in the detection rate. The remaining systems all use arbitration among multiple networks. Arbitration further reduces the false positive rate, and in some cases increases the detection rate slightly. Note that for systems using arbitration, the ratio of false detections to windows examined is extremely low, ranging from 1 false detection per 229,556 windows to down to 1 in 10,387,401, depending on the type of arbitration used. Systems 10, 11, and 12 show that the detector can be tuned to make it more or less conservative. System 10, which uses ANDing, gives an extremely small number of false positives, and has a detection rate of about 78.9%. On the other hand, System 12, which is based on ORing, has a higher detection rate of 90.5% but also has a larger number of false detections. System 11 provides a compromise between the two. The differences in performance of these systems can be understood by considering the arbitration strategy. When using ANDing, a false detection made by only one network is suppressed, leading to a lower false positive rate. On the other hand, when ORing is used, faces detected correctly by only one network will be preserved, improving the detection rate. System 13, which votes among three networks, yields about the same detection rate and lower false positive rate than System 12, which using ORing with two networks.

Based on the results in Table 1, we concluded that System 11 makes an reasonable tradeoff between the number of false detections and the detection rate. System 11 detects on average 85.4% of the faces, with an average of one false detection per 1,319,035 20x20 pixel windows examined. Figure 3 shows examples output images from System 11.

4 Comparison to other systems

[Sung and Poggio, 1994] reports a face detection system based on clustering techniques. Their system, like ours, passes a small window over all portions of the image, and determines whether a face exists in each window. Their system uses a supervised clustering method with six “face” and six “non-face” clusters. Two distance metrics measure the distance of an input image to the prototype clusters. The first metric measures the “partial” distance between the test pattern and the cluster’s 75 most significant eigenvectors. The second distance metric is the Euclidean distance between the test pattern and its projection in the 75 dimensional subspace. These distance measures have close ties with Principal Components Analysis (PCA), as described in [Sung and Poggio, 1994]. The last step in their system is to use either a perceptron or a neural network with a hidden

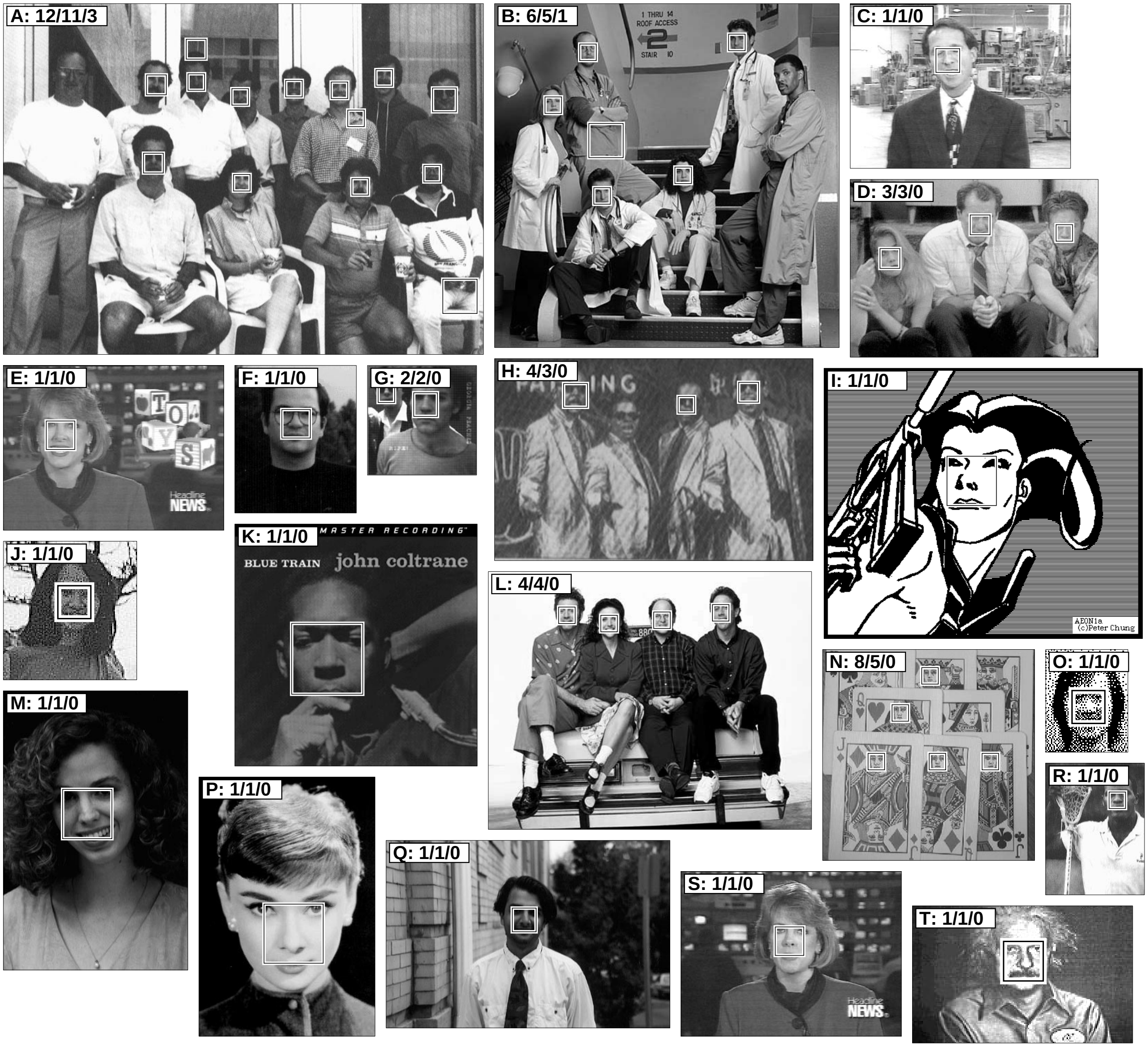

Figure 3: Output obtained from System 11 in Table 1. For each image, three numbers are shown: the number of faces in the image, the number of faces detected correctly, and the number of false detections. Faces are missed in A and H (for unknown reasons), B (large angle), and N (the stylized faces are not reliably detected at the same locations and scales by the two networks, and so are lost by the AND heuristic). False detections are present in A and B. Although the system was trained only on real faces, hand drawn faces are detected in I and N. Images A, H, K, and R were scanned from printed photographs, B, D, G, I, L, and P were obtained from the World Wide Web, C, E, and S are digitized television images, F, J, M, and Q were scanned from photographs, N and T were provided by Sung and Poggio at MIT, and O is a dithered CCD image.

layer, trained to classify points using the two distances to each of the clusters (a total of 24 inputs). The main computational cost in [Sung and Poggio, 1994] is in computing the two distance measures from each new window to 12 clusters. We estimate that this computation requires fifty times as many floating point operations as are needed to classify a window in our system, in which the main costs are in preprocessing and applying neural networks to the window. Table 2 shows the accuracy of their system on Test Set B, along with the our results using Systems 10, 11, and 12 in Table 1, and shows that for equal numbers of false detections, we can achieve higher detection rates.

Table 2: Comparison of [Sung and Poggio, 1994] and our system on Test Set B

System	Missed faces	Detect rate	False detects	False detect rate
10) Networks 1 and 2 → AND(0) → threshold(2,3) → overlap elimination	34	78.1%	3	1/3226028
11) Networks 1 and 2 → threshold(2,2) → overlap elimination → AND(2)	20	87.1%	15	1/645206
12) Networks 1 and 2 → threshold(2,2) → overlap → OR(2) → threshold(2,1) → overlap	11	92.9%	64	1/151220
[Sung and Poggio, 1994] (Multi-layer network)	36	76.8%	5	1/1929655
[Sung and Poggio, 1994] (Perceptron)	28	81.9%	13	1/742175

Although our system is less computationally expensive than [Sung and Poggio, 1994], the system described so far is not real-time because of the number of windows which must be classified. In the related task of license plate detection, [Umezaki, 1995] decreased the number of windows that must be processed. The key idea was to have the neural network be invariant to translations of about 25% of the size of a license plate. Instead of a single number indicating the existence of a face in the window, the output of Umezaki's network is an image with a peak indicating where the network believes a license plate is located. These outputs are accumulated over the entire image, and peaks are extracted to give candidate locations for license plates. In [Rowley *et al.*, 1995], we show that a face detection network can also be made translation invariant. However, this translation invariant face detector makes many more false detections than one that detects only centered faces. We use the centered face detector to verify candidates found by the translation invariant network. With this approach, we can process a 320x240 pixel image in less than 5 seconds on an SGI Indy workstation. This technique is related, at a high level, to the technique presented in [Vaillant *et al.*, 1994].

5 Conclusions and future research

Our algorithm can detect between 78.9% and 90.5% of faces in a set of 130 test images, with an acceptable number of false detections. Depending on the application, the system can be made more or less conservative by varying the arbitration heuristics or thresholds used. The system has been tested on a wide variety of images, with many faces and unconstrained backgrounds.

There are a number of directions for future work. The main limitation of the current system is that it only detects upright faces looking at the camera. Separate versions of the system could be trained for different head orientations, and the results could be combined using arbitration methods similar to those presented here. Other methods of improving system performance include obtaining more positive examples for training, or applying more sophisticated image preprocessing and normalization techniques. For instance, the color segmentation method used in [Hunke, 1994] for color-based face tracking could be used to filter images. The face detector would then be applied only to portions of the image which contain skin color, which would speed up the algorithm as well as eliminating some false detections.

One application of this work is in the area of media technology. Every year, improved technology provides cheaper and more efficient ways of storing information. However, automatic high-level classification of the information content is very limited; this is a bottleneck preventing media technology from reaching its full potential. The work described above allows a user to make queries of the form "Which scenes in this video contain human faces?" and to have the query answered automatically.

Acknowledgements

The authors thank Kah-Kay Sung and Dr. Tomaso Poggio (at MIT) and Dr. Woodward Yang (at Harvard) for providing a series of test images and a mug-shot database, respectively. Michael Smith (at CMU) provided some digitized television images for testing purposes. We also thank Eugene Fink, Xue-Mei Wang, Hao-Chi Wong, Tim Rowley, and Kaari Flagstad for comments on drafts of this paper.

References

[Hunke, 1994] H. Martin Hunke. Locating and tracking of human faces with neural networks. Master's thesis, University of Karlsruhe, 1994.

[Le Cun *et al.*, 1989] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropogation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[Rowley *et al.*, 1995] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visual scenes. CMU-CS-95-158R, Carnegie Mellon University, November 1995. Also available at http://www.cs.cmu.edu/~har/faces.html.

[Sung and Poggio, 1994] Kah-Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. A.I. Memo 1521, CBCL Paper 112, MIT, December 1994.

[Umezaki, 1995] Tazio Umezaki. Personal communication, 1995.

[Vaillant *et al.*, 1994] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings on Vision, Image, and Signal Processing*, 141(4), August 1994.

[Waibel *et al.*, 1989] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *Readings in Speech Recognition*, pages 393–404, 1989.