Rotation Invariant Neural Network-Based Face Detection*

Henry A. Rowley[1] har@cs.cmu.edu
Shumeet Baluja[2,1] baluja@jprc.com
Takeo Kanade[1] tk@cs.cmu.edu

[1] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213
[2] Justsystem Pittsburgh Research Center, 4616 Henry Street, Pittsburgh, PA 15213

Abstract

In this paper, we present a neural network-based face detection system. Unlike similar systems which are limited to detecting upright, frontal faces, this system detects faces at any degree of rotation in the image plane. The system employs multiple networks; a "router" network first processes each input window to determine its orientation and then uses this information to prepare the window for one or more "detector" networks. We present the training methods for both types of networks. We also perform sensitivity analysis on the networks, and present empirical results on a large test set. Finally, we present preliminary results for detecting faces rotated out of the image plane, such as profiles and semi-profiles.

1. Introduction

In our observations of face detector demonstrations, we have found that users expect faces to be detected at any angle, as shown in Figure 1. In this paper, we present a neural network-based algorithm to detect faces in gray-scale images. Unlike similar previous systems which could only detect upright, frontal faces [3, 4, 6–9, 12, 13, 15, 17, 18], this system efficiently detects frontal faces which can be arbitrarily rotated within the image plane. We also present preliminary results on detecting upright faces rotated out of the image plane, such as profiles and semi-profiles.

Many face detection systems are template-based; they encode facial images directly in terms of pixel intensities. These images can be characterized by probabilistic models of the set of face images [4, 7, 9], or implicitly by neural networks or other mechanisms [3, 6, 8, 12, 13, 15, 17]. Other researchers have taken the approach of extracting features and applying either manually or automatically generated rules for evaluating these features [5, 18]. By using a graph-matching algorithm on detected features, [5] also demonstrated rotation invariance. We present a general method to make template-based face detectors rotation invariant.

*This work was partially supported by Hewlett-Packard Corporation, Siemens Corporate Research, Inc., the Department of the Army, Army Research Office (grant number DAAH04-94-G-0006), and by the Office of Naval Research (grant number N00014-95-1-0591). The views and conclusions contained in this document are those of the authors, and do not necessarily represent the official policies of the sponsors.

Our system directly analyzes image intensities using neural networks, whose parameters are learned automatically from training examples. There are many ways to use neural networks for rotated-face detection. The simplest would be to employ one of the existing frontal, upright, face detection systems. Systems such as [12] use a neural-network based filter that receives as input a small window of the image, and generates an output signifying the presence or absence of a face. To detect faces anywhere in the image, the filter is applied at every location in the image. To detect faces larger than the window size, the input image is repeatedly subsampled to reduce its size, and the filter is applied at each scale. To extend this framework to capture rotated faces, the entire image can be repeatedly rotated by small increments and the detector can be applied to each rotated image. However, this would be an extremely computationally expensive procedure. For example, the system reported in [12] was invariant to approximately $10°$ of rotation from upright (both clockwise and counterclockwise). Therefore, the entire detection procedure would need to be applied *at least* 18 times to each image, with the image rotated in increments of $20°$.

An alternate, significantly faster procedure is described in this paper, extending some early results in [1]. This procedure uses a separate neural network, termed a "router", to analyze the input window before it is processed by the face detector. The router's input is the same region that the detector network will receive as input. If the input contains a face, the router returns the angle of the face. The window can then be "derotated" to make the face upright. Note that

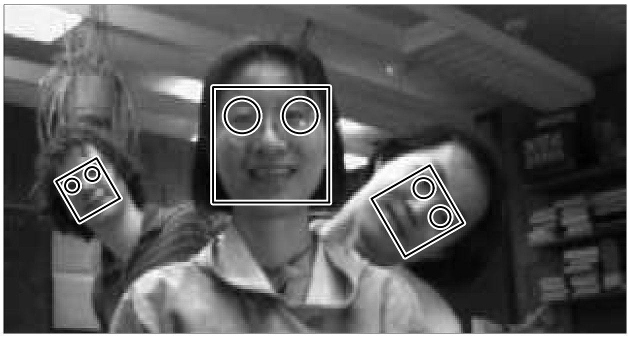

Figure 1. The output of our new system.

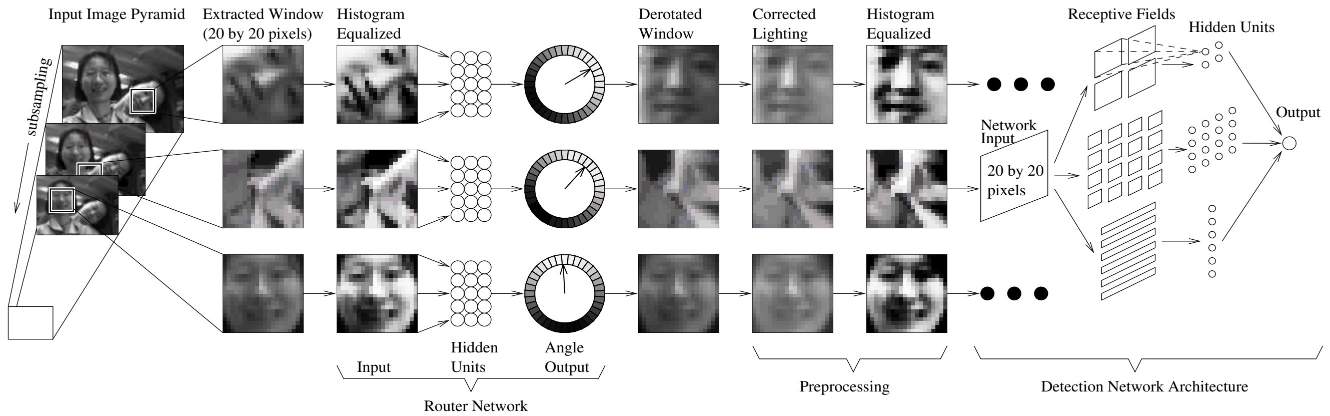

Figure 2. Overview of the algorithm.

the router network *does not* require a face as input. If a non-face is encountered, the router will return a meaningless rotation. However, since a rotation of a non-face will yield another non-face, the detector network will still not detect a face. On the other hand, a rotated face, which would not have been detected by the detector network alone, will be rotated to an upright position, and subsequently detected as a face. Because the detector network is only applied once at each image location, this approach is significantly faster than exhaustively trying all orientations.

Detailed descriptions of the example collection and training methods, network architectures, and arbitration methods are given in Section 2. We then analyze the performance of each part of the system separately in Section 3, and test the complete system on two large test sets in Section 4. We find that the system is able to detect 79.6% of the faces over a total of 180 complex images, with a very small number of false positives. Conclusions and directions for future research are presented in Section 5.

2. Algorithm

The overall algorithm for the detector is given in Figure 2. Initially, a pyramid of images is generated from the original image, using scaling steps of 1.2. Each 20x20 pixel window of each level of the pyramid then goes through several processing steps. First, the window is preprocessed using histogram equalization, and given to a *router network*. The rotation angle returned by the router is then used to rotate the window with the potential face to an upright position. Finally, the *derotated window* is preprocessed and passed to one or more detector networks [12], which decide whether or not the window contains a face.

The system as presented so far could easily signal that there are two faces of very different orientations at adjacent pixel locations in the image. To counter such anomalies, and to reinforce correct detections, some arbitration heuristics are employed. The design of the router and detector networks and the arbitration scheme are presented in the following subsections.

2.1. The Router Network

The first step in processing a window of the input image is to apply the router network. This network assumes that its input window contains a face, and is trained to estimate its orientation. The inputs to the network are the intensity values in a 20x20 pixel window of the image (which have been preprocessed by a standard histogram equalization algorithm). The output angle of rotation is represented by an array of 36 output units, in which each unit i represents an angle of $i*10^\circ$. To signal that a face is at an angle of θ, each output is trained to have a value of $\cos(\theta - i*10^\circ)$. This approach is closely related to the Gaussian weighted outputs used in the autonomous driving domain [11]. Examples of the training data are given in Figure 3.

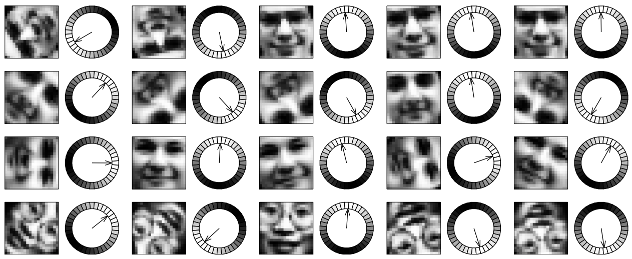

Figure 3. Example inputs and outputs for training the router network.

Previous algorithms using Gaussian weighted outputs inferred a single value from them by computing an average of the positions of the outputs, weighted by their activations. For angles, which have a periodic domain, a weighted sum of angles is insufficient. Instead, we interpret each output as a weight for a vector in the direction indicated by the output number i, and compute a weighted sum as follows:

$$\left(\sum_{i=0}^{35} \text{output}_i * \cos(i*10^\circ), \sum_{i=0}^{35} \text{output}_i * \sin(i*10^\circ)\right)$$

The direction of this average vector is interpreted as the angle of the face.

The training examples are generated from a set of manually labelled example images containing 1048 faces. In each face, the eyes, tip of the nose, and the corners and center of the mouth are labelled. The set of labelled faces are then aligned to one another using an iterative procedure [12]. We first compute the average location for each of the labelled features over the entire training set. Then, each face is aligned with the average feature locations, by computing the rotation, translation, and scaling that minimizes the distances between the corresponding features. Because such transformations can be written as linear functions of their parameters, we can solve for the best alignment using an over-constrained linear system. After iterating these steps a small number of times, the alignments converge.

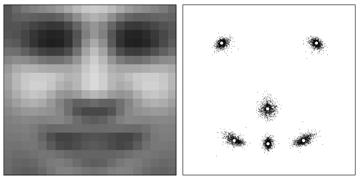

Figure 4. Left: Average of upright face examples. Right: Positions of average facial feature locations (white circles), and the distribution of the actual feature locations from all the examples (black dots).

The averages and distributions of the feature locations are shown in Figure 4. Once the faces are aligned to have a known size, position, and orientation, we can control the amount of variation introduced into the training set. To generate the training set, the faces are rotated to a random (known) orientation, which will be used as the target output for the router network. The faces are also scaled randomly (in the range from 1 to 1.2) and translated by up to half a pixel. For each of 1048 faces, we generate 15 training examples, yielding a total of 15720 examples.

The architecture for the router network consists of three layers, an input layer of 400 units, a hidden layer of 15 units, and an output layer of 36 units. Each layer is fully connected to the next. Each unit uses a hyperbolic tangent activation function, and the network is trained using the standard error backpropogation algorithm.

2.2. The Detector Network

After the router network has been applied to a window of the input, the window is derotated to make any face that may be present upright.

The remaining task is to decide whether or not the window contains an upright face. The algorithm used for detection is identical to the one presented in [12]. The resampled image, which is also 20x20 pixels, is preprocessed in two steps [13]. First, we fit a function which varies linearly across the window to the intensity values in an oval region inside the window. The linear function approximates the overall brightness of each part of the window, and can be subtracted to compensate for a variety of lighting conditions. Second, histogram equalization is performed, which expands the range of intensities in the window. The preprocessed window is then given to one or more *detector networks*. The detector networks are trained to produce an output of $+1.0$ if a face is present, and -1.0 otherwise.

The detectors have two sets of training examples: images which are faces, and images which are not. The positive examples are generated in a manner similar to that of the router; however, as suggested in [12], the amount of rotation of the training images is limited to the range $-10°$ to $10°$.

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 much harder to get a representative sample of those which do not. Instead of collecting the images before training is started, the images are collected during training in the following "bootstrap" manner, adapted from [13]:

1. Create an initial set of 1000 random non-face images.
2. Train the neural network to produce an output of $+1.0$ for the face examples, and -1.0 for the non-face examples. In the first iteration, the network's weights are initialized randomly. After the first iteration, we use the weights computed by training in the previous iteration as the starting point.
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 > 0.0).
4. Select up to 250 of these subimages at random, and add them into the training set as negative examples. Go to step 2.

Some examples of non-faces that are collected during training are shown in Figure 5. At runtime, the detector network will be applied to images which have been derotated, so it may be advantageous to collect negative training examples from the set of derotated non-face images, rather than only non-face images in their original orientations. In Section 4, both possibilities are explored.

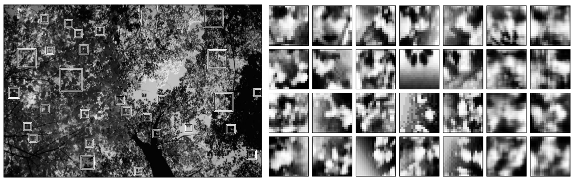

Figure 5. Left: A partially-trained system is applied to images which do not contain faces. Right: Any regions detected as faces are errors, which can be added into the set of negative training examples.

2.3. The Arbitration Scheme

As mentioned earlier, it is possible for the system described so far to signal faces of very different orientations at adjacent pixel locations. A simple postprocessing heuristic is employed to rectify such inconsistencies. Each detection is placed in a 4-dimensional space, where the dimensions are the x and y positions of the center of the face, the level in the image pyramid at which the face was detected, and the angle of the face, quantized to increments of $10°$. For each detection, we count the number of detections within 4 units along each dimension (4 pixels, 4 pyramid levels, or $40°$). This number can be interpreted as a confidence measure, and a threshold is applied. Once a face passes the threshold, any other detections in the 4-dimensional space which would overlap it are discarded.

Although this postprocessing heuristic was found to be quite effective at eliminating false detections, we have found that a single detection network still yields an unacceptably high false detection rate. To further reduce the number of false detections, and reinforce correct detections, we arbitrate between two independently trained detector networks, as in [12]. Each network is given the same set of positive examples, but starts with different randomly set initial weights. Therefore, each network learns different features, and makes different mistakes. To use the outputs of these two networks, the postprocessing heuristics of the previous paragraph are applied to the outputs of each individual network, and then the detections from the two networks are ANDed. The specific preprocessing thresholds used in the experiments will be given in Sections 4. These arbitration heuristics are very similar to, but computationally less expensive than, those presented in [12].

3. Analysis of the Networks

In order for the system described above to be accurate, the router and detector must perform robustly and compatibly. Because the output of the router network is used to derotate the input for the detector, the angular accuracy of the router must be compatible with the angular invariance of the detector. To measure the accuracy of the router, we generated test example images based on the training images, with angles between $-30°$ and $30°$ at $1°$ increments. These images were given to the router, and the resulting histogram of angular errors is given in Figure 6 (left). As can be seen, 92% of the errors are within $\pm 10°$.

The detector network was trained with example images having orientations between $-10°$ and $10°$. It is important to determine whether the detector is in fact invariant to rotations within this range. We applied the detector to the same set of test images as the router, and measured the fraction of faces which were correctly classified as a function of the angle of the face. Figure 6 (right) shows that the detector detects over 90% of the faces that are within $10°$ of upright, but the accuracy falls with larger angles. In summary, since the router's angular errors are usually within $10°$, and since the detector can detect most faces which are rotated up to $10°$, the two networks are compatible.

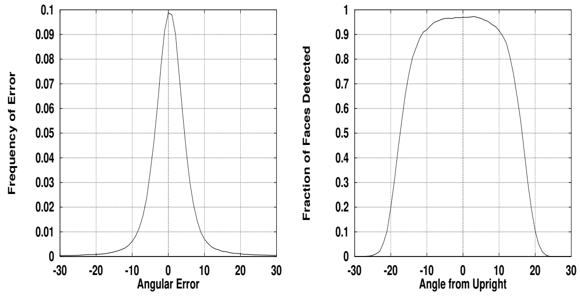

Figure 6. Left: Frequency of errors in the router network with respect to the angular error (in degrees). Right: Fraction of faces that are detected by the detector networks, as a function of the angle of the face from upright.

4. Empirical Results

In this section, we integrate the pieces of the system, and test it on two sets of images. The first set, which we will call the upright test set, is Test Set 1 from [12]. It contains many images with faces against complex backgrounds and many images without any faces. There are a total of 130 images, with 511 faces (of which 469 are within $10°$ of upright), and 83,099,211 windows to be processed. The second test set, referred to as the rotated test set, consists of 50 images (with 34,064,635 windows) containing 223 faces, of which 210 are at angles of more than $10°$ from upright.[1]

The upright test set is used as a baseline for comparison with an existing upright face detection system [12]. This will ensure that the modifications for rotated faces do not hamper the ability to detect upright faces. The rotated test set will demonstrate the new capabilities of our system.

4.1. Router Network with Upright Face Detectors

The first system we test employs the router network to determine the orientation of any potential face, and then applies two standard upright face detection networks from [12]. Table 1 shows the number of faces detected and the number of false alarms generated on the two test sets. We first give the results of the individual detection networks, and then give the results of the post-processing heuristics (using a threshold of one detection). The last row of the table reports the result of arbitrating the outputs of the two networks, using an AND heuristic. This is implemented by first post-processing the outputs of each individual network, followed by requiring that both networks signal a detection at the same location, scale, and orientation. As can be seen in the table, the post-processing heuristics significantly reduce the number of false detections, and arbitration helps further. Note that the detection rate for the rotated test set is higher than that for the upright test set, due to differences in the overall difficulty of the two test sets.

[1] These test sets are available over the World Wide Web at the URL `http://www.cs.cmu.edu/~har/faces.html`.

Table 1. Results of first applying the router network, then applying the standard detector networks [12] at the appropriate orientation.

	Upright Test Set		Rotated Test Set	
System	Detect %	# False	Detect %	# False
Network 1	89.6%	4835	91.5%	2174
Network 2	87.5%	4111	90.6%	1842
Net 1 → Postproc	85.7%	2024	89.2%	854
Net 2 → Postproc	84.1%	1728	87.0%	745
Postproc → AND	81.6%	293	85.7%	119

4.2. Proposed System

Table 1 shows a significant number of false detections. This is in part because the detector networks were applied to a different distribution of images than they were trained on. In particular, at runtime, the networks only saw images that were derotated by the router. We would like to match this distribution as closely as possible during training. The positive examples used in training are already in upright positions. During training, we can also run the scenery images from which negative examples are collected through the router. We trained two new detector networks using this scheme, and their performance is summarized in Table 2. As can be seen, the use of these new networks reduces the number of false detections by at least a factor of 4. Of the systems presented here, this one has the best trade-off between the detection rate and the number of false detections. Images with the detections resulting from arbitrating between the networks are given in Figure 7 .

Table 2. Results of our system, which first applies the router network, then applies detector networks trained with derotated negative examples.

	Upright Test Set		Rotated Test Set	
System	Detect %	# False	Detect %	# False
Network 1	81.0%	1012	90.1%	303
Network 2	83.2%	1093	89.2%	386
Net 1 → Postproc	80.2%	710	89.2%	221
Net 2 → Postproc	82.4%	747	88.8%	252
Postproc → AND	76.9%	34	85.7%	15

4.3. Exhaustive Search of Orientations

To demonstrate the effectiveness of the router for rotation invariant detection, we applied the two sets of detector networks described above without the router. The detectors were instead applied at 18 different orientations (in increments of 20°) for each image location. Table 3 shows the results using the standard upright face detection networks of [12], and Table 4 shows the results using the detection networks trained with derotated negative examples.

Recall that Table 1 showed a larger number of false positives compared with Table 2, due to differences in the training and testing distributions. In Table 1, the detection networks were trained with false-positives in their original orientations, but were tested on images that were rotated from their original orientations. Similarly, if we apply detector networks to images at all 18 orientations, we should expect an increase in the number of false positives because of the differences in the training and testing distributions (see Tables 3 and 4). The detection rates are higher than for systems using the router network. This is because any error by the router will lead to a face being missed, whereas an exhaustive search of all orientations may find it. Thus, the differences in accuracy can be viewed as a tradeoff between the detection and false detection rates, in which better detection rates come at the expense of much more computation.

Table 3. Results of applying the standard detector networks [12] at 18 different image orientations.

	Upright Test Set		Rotated Test Set	
System	Detect %	# False	Detect %	# False
Network 1	93.7%	17848	96.9%	7872
Network 2	94.7%	15828	95.1%	7328
Net 1 → Postproc	87.5%	4828	94.6%	1928
Net 2 → Postproc	89.8%	4207	91.5%	1719
Postproc → AND	85.5%	559	90.6%	259

Table 4. Networks trained with derotated examples, but applied at all 18 orientations.

	Upright Test Set		Rotated Test Set	
System	Detect %	# False	Detect %	# False
Network 1	90.6%	9140	97.3%	3252
Network 2	93.7%	7186	95.1%	2348
Net 1 → Postproc	86.9%	3998	96.0%	1345
Net 2 → Postproc	91.8%	3480	94.2%	1147
Postproc → AND	85.3%	195	92.4%	67

4.4. Upright Detection Accuracy

Finally, to check that adding the capability of detecting rotated faces has not come at the expense of accuracy in detecting upright faces, in Table 5 we present the result of applying the original detector networks and arbitration method from [12] to the two test sets used in this paper.[2] As expected, this system does well on the upright test set, but has a poor detection rate on the rotated test set.

Table 5. Results of applying the original algorithm and arbitration method from [12] to the two test sets.

	Upright Test Set		Rotated Test Set	
System	Detect %	# False	Detect %	# False
Network 1	90.6%	928	20.6%	380
Network 2	92.0%	853	19.3%	316
Net 1 → Postproc	89.4%	516	20.2%	259
Net 2 → Postproc	90.6%	453	17.9%	202
Threshold → AND	85.3%	31	13.0%	11

Table 6 shows a breakdown of the detection rates of the above systems on faces that are rotated less or more than 10° from upright. As expected, the original upright face detector trained exclusively on upright faces and negative examples in their original orientations gives a high detection

[2] The results for the upright test set are slightly different from those presented in [12] because we now check for the detection of 4 upside-down faces, which were present, but ignored, in the original test set. Also, there are slight differences in the way the image pyramid is generated.

Figure 7. Result of arbitrating between two networks trained with derotated negative examples. The label in the upper left corner of each image (D/T/F) gives the number of faces detected (D), the total number of faces in the image (T), and the number of false detections (F). The label in the lower right corner of each image gives its size in pixels.

rate on upright faces. Our new system has a slightly lower detection rate on upright faces for two reasons. First, the detector networks cannot recover from all the errors made by the router network. Second, the detector networks which are trained with derotated negative examples are more conservative in signalling detections; this is because the derotation process makes the negative examples look more like faces, which makes the classification problem harder.

Table 6. Breakdown of detection rates for upright and rotated faces from both test sets.

System	All Faces	Upright Faces ($\leq 10^\circ$)	Rotated Faces ($> 10^\circ$)
New system (Table 2)	79.6%	77.2%	84.1%
Upright detector [12]	63.4%	88.0%	16.3%

5. Summary and Extensions

This paper has demonstrated the effectiveness of detecting faces rotated in the image plane by using a router network in combination with an upright face detector. The system is able to detect 79.6% of faces over two large test sets, with a small number of false positives. The technique is applicable to other template-based object detection schemes.

We are investigating the use of the above scheme to handle out-of-plane rotations. There are two ways in which this could be approached. The first is directly analogous to handling in-plane rotations: using knowledge of the shape and symmetry of the face, it may be possible to convert a profile or semi-profile view of a face to a frontal view (for related work, see [2, 16]). A second approach, and the one we have explored, is to partition the views of the face, and to train separate detector networks for each view. We used five views: left profile, left semi-profile, frontal, right semi-profile, and right profile. The router is responsible for directing the input window to one of these view detectors [19].

Figure 8 shows some preliminary results. As can be seen, there are still a significant number of false detections and missed faces. We suspect that one reason for this is that our training data is not representative of the variations present in real images. Most of our profile training images are taken from the FERET database [10], which has very uniform lighting conditions.

Figure 8. Detection of faces rotated out-of-plane.

There are two immediate directions for future work. First, it would be interesting to merge the systems for in-plane and out-of-plane rotations. One approach is to build a single router which recognizes all views of the face, then rotates the image in-plane to a canonical orientation, and presents the image to the appropriate view detector network. The second area for future work is improvement to the speed of the system. Based on the work of [14], [12] presented a quick algorithm based on the use of a fast (but somewhat inaccurate) candidate detector network, whose results could then be checked by the detector networks. A similar technique may be applicable to the present work.

References

[1] S. Baluja. Face detection with in-plane rotation: Early concepts and preliminary results. Technical Report JPRC-1997-001-1, Justsystem Pittsburgh Research Center, 1997.

[2] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. Technical Report A.I. Memo 1431, MIT, November 1993.

[3] G. Burel and D. Carel. Detection and localization of faces on digital images. *Pattern Recognition Letters*, 15:963–967, October 1994.

[4] A. J. Colmenarez and T. S. Huang. Face detection with information-based maximum discrimination. In *Computer Vision and Pattern Recognition*, pages 782–787, 1997.

[5] T. K. Leung, M. C. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. In *Fifth International Conference on Computer Vision*, pages 637–644, June 1995.

[6] S. H. Lin, S. Y. Kung, and L. J. Lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Transactions on Neural Networks, Special Issue on Artificial Neural Networks and Pattern Recognition*, 8(1), January 1997.

[7] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. In *Fifth International Conference on Computer Vision*, pages 786–793, June 1995.

[8] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[9] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *Computer Vision and Pattern Recognition*, pages 84–91, 1994.

[10] P. J. Phillips, P. J. Rauss, and S. Z. Der. FERET (face recognition technology) recognition algorithm development and test results. Technical Report ARL-TR-995, Army Research Laboratory, October 1996.

[11] D. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. PhD thesis, Carnegie Mellon University, February 1992.

[12] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1), January 1998.

[13] K.-K. Sung. *Learning and Example Selection for Object and Pattern Detection*. PhD thesis, MIT AI Lab, January 1996.

[14] T. Umezaki. Personal communication, 1995.

[15] 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.

[16] T. Vetter, M. J. Jones, and T. Poggio. A bootstrapping algorithm for learning linear models of object classes. In *Computer Vision and Pattern Recognition*, pages 40–46, June 1997.

[17] G. Yang and T. S. Huang. Human face detection in a complex background. *Pattern Recognition*, 27(1):53–63, 1994.

[18] K. C. Yow and R. Cipolla. Feature-based human face detection. Technical Report CUED/F-INFENG/TR 249, Department of Engineering, University of Cambridge, England, 1996.

[19] M. Zhang and J. Fulcher. Face recognition using artificial neural network group-based adaptive tolerance (GAT) trees. *IEEE Transactions on Neural Networks*, 7(3):555–567, 1996.