Face Detection with In-Plane Rotation: Early Concepts and Preliminary Results

Technical Report JPRC-TR-97-001

Shumeet Baluja[1,2]

[1] Justsystem Pittsburgh Research Center
4616 Henry Street
Pittsburgh, PA 15213

[2] School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
baluja@jprc.com

Abstract. This paper presents a method for extending upright, frontal, template-based face detection systems to efficiently handle all in-plane rotations. Detecting rotated faces is a two step procedure. First a "De-Rotation" network is used to process each input window. If there is a face in the window, this network determines its angle of rotation. Based upon this estimated angle of rotation, the window is then rotated to an upright position. Second, a "Detection" network, or multiple detection networks, are used to determine whether the rotated window contained an upright face. The training methods for both networks are presented. Preliminary empirical results are also provided.

1 Introduction

This paper presents a general method to extend many template-based frontal, upright, face detection systems to handle in-plane rotations of the face. There have been many template-based face detection systems developed, for example, see [2, 3, 6–8, 11–13, 15]. Other systems, such as [5], can also achieve rotation invariance by extracting smaller features of the face and using graph-matching algorithms. In this paper, we concentrate on template-based methods, in particular the one presented by Rowley, Baluja & Kanade in [11].

The simplest method for creating a system which is invariant to rotations within the image-plane is to employ an existing frontal, upright, face detection system. Systems such as [11] use a neural-network based filter that receives as input a small, constant-sized window of the image, and generates an output signifying the presence or absence of a face. 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 sub-sampled to reduce its size, and the filter is applied at each scale. To extend this framework to capture faces which are rotated, the entire image can be repeatedly rotated by small increments and the detection system can be applied to each rotated image. However, this is an extremely computationally expensive procedure. For example, the system reported in [11] 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°.

The procedure presented in this paper is significantly faster than the one described above. This procedure uses two networks to perform the detection. The first network, termed a "De-Rotation" network, analyzes the input window before it is given to the "Detection" network. The De-Rotation network's input is the same region that the detection network will receive as input. If the input contains a face, the De-Rotation network returns the angle of the face. The window can then be rotated by the negative of that angle to make the face upright. Note that the De-Rotation network *does not* require a face as input. If a non-face image is encountered, the router will return a meaningless rotation. However, since a rotation of a non-face image will yield another non-face image, the detection network will still not detect a face. On the other hand, a rotated face, which would not have been detected by the detection network alone, will be rotated to an upright position, and subsequently detected as a face. Before giving the details of this system, the next section gives a review of the face detection system described in [10, 11].

2 Frontal, Upright, Face Detection using the Rowley, Baluja & Kanade System [11]

This section was largely taken from [11]; it is provided to give the reader the necessary background for the next section.

The "Detection" networks used in this study are taken from [11]. The detection network operates as 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 reduced in size (by sub-sampling), and the filter is applied at each size. For the work presented here, the filter is applied at every pixel position in the image, and the image is scaled down by a factor of 1.2 for each step in the pyramid.

The filtering algorithm is shown in Figure 1. First, a preprocessing step, adapted from [12], 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 in across the window. We fit a function which varies linearly across the window to the intensity values in an oval region inside the window, see [11]. 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.

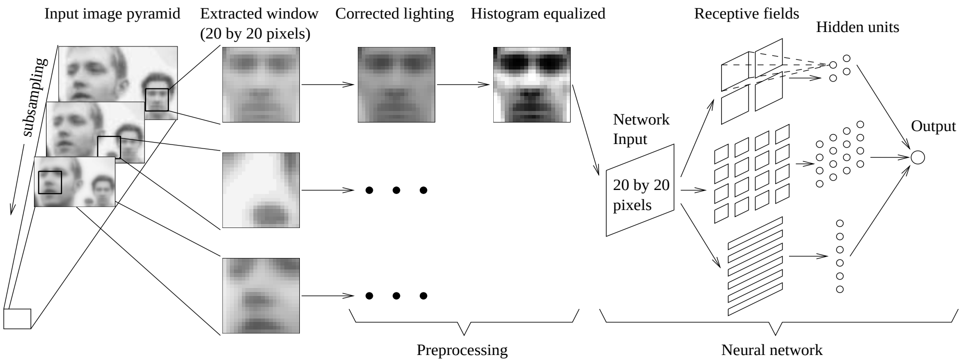

Fig. 1. Overview of algorithm for frontal, upright, face detection. Taken from [10].

To train the neural network used for detection, a large number of face and non-face images are needed. Approximately 1050 face examples were used. For each of these faces, fifteen face examples are generated 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 20%.[3] 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.

As pointed out in [10], training the detector to recognize non-face images is a difficult problem because the space of non-face images is much larger than the space of face images. Collecting 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 [12]:

[3] The reader may question why it is not possible to simply train a network with images which have been rotated by larger amounts, and avoid this rotation problem altogether. Unfortunately, in the experiments conducted, training with larger amounts of rotation adversely affected performance. This is because the larger the amount of variability in the training set, the harder it is to train a network which is able to maintain an acceptably low number of false-positives. However, this may be an interesting avenue for future exploration.

1. Create an initial set of non-face images by generating 1000 random images. Apply the preprocessing steps to each of these images.
2. Train a 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 backpropagation with momentum [4]. On the first iteration of this loop, 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 sub-images in which the network incorrectly identifies a face (an output activation > 0).
4. Select up to 250 of these sub-images at random, apply the preprocessing steps, and add them into the training set as negative examples. Go to step 2.

[11] used 120 images of scenery for collecting negative examples in the bootstrap manner described above. Further, several networks were trained in this manner. Various arbitration heuristics were used to combine the results of the multiple networks. More details about the training procedure can be found in [11].

3 The De-Rotation Network

The De-Rotation network is trained to output the rotation of a face given in the 20x20 pixel inputs. The training examples consist of a set of images which contain faces each rotated a random amount within the image-plane. These examples are generated from the positive training examples used for training the detection network. 25 examples are generated from each of the original positive examples; each of these examples is rotated to a random amount between 0° - 360°. The types of examples used for training are shown in Figure 2.

The network must output a value which can be interpreted as a value between 0° - 360°. There are many possibilities for representing the output of the network. Pomerleau has explored many of these for the design of a neural-network controller for the autonomous navigation of a vehicle [9]. In Pomerleau's application, the output of the network was interpreted to be the steering angle of the vehicle to ensure that the vehicle stayed on the road. Several output representations were explored:

1. *Single Unit*: The activation amount of a single output unit (usually either between 0-1 or -1 and +1) is mapped linearly between the range of 0° - 360° to determine the angle of rotation.
2. *1-of-N Encoding*: **N** units are used to represent the output. Each unit represents $360/N°$. For example, if there were 180 units, and if unit 30 had the highest activation, this would indicate a rotation of 60°.
3. *Gaussian Output Encoding*: Similar to the 1-of-N encoding; **N** units are used to represent the output. However, instead of only training the network to turn on a single output, outputs close to the desired output are also turned on

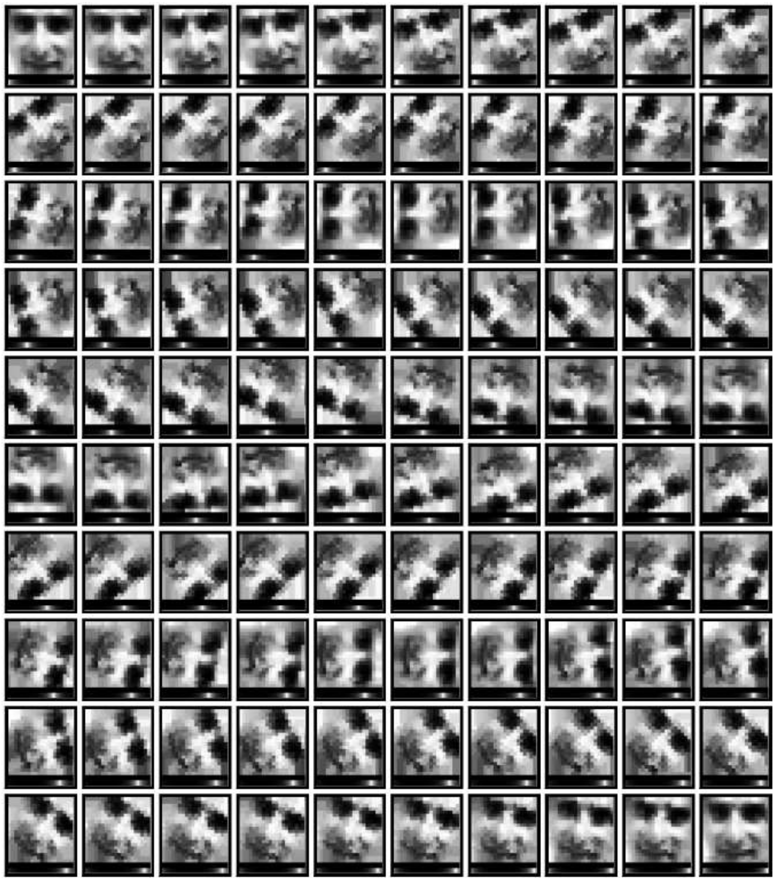

Fig. 2. 100 input/output training examples shown. The thin line below each face represents the target outputs for the network (72 output units total). In this figure, the same face is rotated in increments of 3.6°. Note that the Gaussian output vector wraps around the boundaries, ensuring that rotations close to 0°, such as 359° and 1°, are treated similarly.

in proportion to the distance from the desired output. For example, in training to indicate a rotation of 60° in 180 units, unit 30 would have maximum activation, units 29 and 31 less activation, 28 and 32 even less, etc. This representation avoids the imposed discontinuities of the strict 1-of-N encoding for images which are similar, but which have slight differences in rotations. Pomerleau found this representation to work the best of the three [9].

For this study, 72 output units were used with the Gaussian output encoding. At runtime, after the forward-propagation pass is done through the network, a Gaussian is fit to the outputs, and the estimated peak of the Gaussian is used to determine the network's estimate of the rotation of the face (similar to the method proposed by [9]). This allows a finer granularity than would be possible if only the maximum activation of the output units was used (*ie.*, it is possible to get values other than multiples of 5° with 72 output units); see [9] for details. Sample input and output pairs are shown in Figure 2.

The "De-Rotation" network has a single hidden layer consisting of a total of 100 units. In this hidden layer, there are 4 sets of 25 units. Each unit in the hidden layer has retinal connections to the input layer. The receptive fields of the hidden units are shown in Figure 3. Each unit connects to a 4x4 region of the input. Each set of 25 units covers the entire input without overlap.

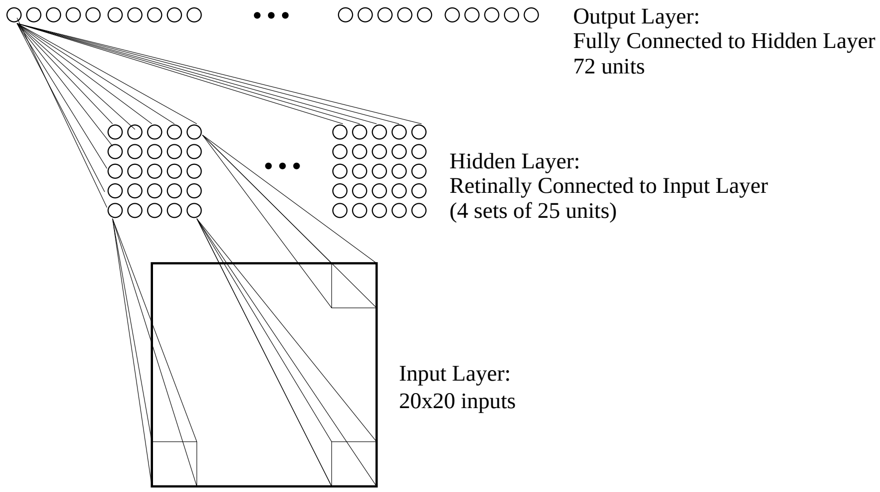

Fig. 3. Network architecture for the "De-Rotation" network. The network has 100 hidden units, each of which is connected to a 4x4 patch of the input.

The network is trained by the standard error backpropagation training method, see [4] for details.

3.1 Performance of the De-Rotation Network

As described in the previous section, the face detector is trained to handle small variations in rotation of approximately $-10°$ - $10°$. Therefore, the De-Rotation network must be able to determine the angle of faces to within this tolerance. Figures 4 & 5 show the distribution of errors in the estimation of rotation, as a function of degrees. These histograms were generated by testing the network on an image set which consisted of 12000 face images which were randomly rotated between 0° - 360° (there were approximately 1000 original face images from which 12 randomly rotated images were created). As can be seen, approximately 73% of errors were less than 5°, and approximately 93% of the errors were less than 10°.

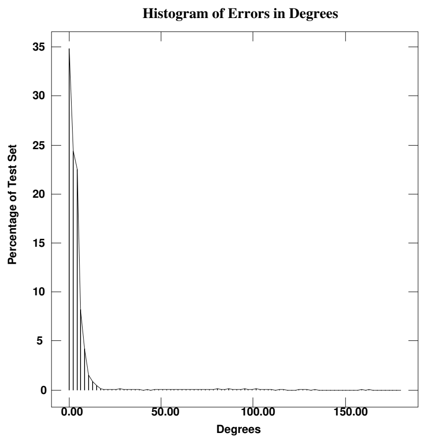

Fig. 4. Histogram of the errors in the estimation of rotation.

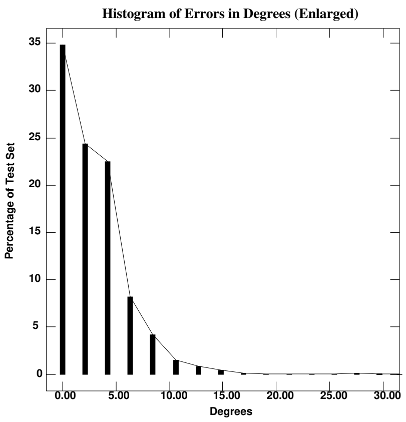

Fig. 5. Histogram of the errors in the estimation of rotation - enlarged.

4 Integrating the De-Rotation and Detection Networks

In this section, a small set of experiments are conducted to ascertain whether the the De-Rotation network and the Detection network will work together. The following steps are repeated for each 20x20 image which is given to the system:

1. The window is preprocessed to account for lighting variation. This step was described earlier in Section 2, and is adapted from [12].
2. Histogram Equalization is performed on an oval region inside the window [11,12].
3. The window is passed to the De-Rotation network. A Gaussian is fit to the network's outputs, and the peak is interpreted as an angle between $0°$ - $360°$.
4. The window is rotated by the angle returned in the previous step. Because the rotation may cause unfilled pixels to enter the image, all unfilled pixels are estimated from nearby pixels. Instead of performing this estimation procedure, this process can be improved by re-sampling the image from which the 20x20 window was extracted.
5. The rotated-window is passed to the detection network. If the detection network outputs a positive value, the window is labeled as a face, otherwise the window is labeled as a non-face.

As suggested in [11], this process can be repeated for multiple networks. To be used for face detection in arbitrary images, it can be repeated for every pixel location in the original image, as well as in sub-sampled images, to account for translation and scale variations.

4.1 Empirical Comparison

The first tests conducted are used to set a base-line for the performance that we can expect to achieve with the De-Rotation network. First, we examine the performance of one of the networks from the face detector (from [11]) on a set of 4000 upright faces images of size 20x20. These images were synthesized by randomly rotating a set of 1000 face images (about their center points) up to $10°$, scaling between 90% and 110%, translating up to half a pixel, and mirroring. The face detector was trained to handle exactly this amount of variation. As can be seen in Table 1, the face detector is able to detect approximately 86.5% of these faces. For the second experiment, each of these 4000 images was rotated 10 times by a random amount, chosen uniformly from $0°$ - $360°$ (creating a total of 40000 images). As expected, since the network is not trained to handle rotations greater than $10°$, the network's performance drops dramatically, to only being able to detect approximately 8% of the images as faces.

Table 1. Results of using the Original System Described in [11] without any adjustment for rotations.

	Number Detected	Detection Rate
Original Face Images	3460/4000	86.5%
Rotated Face Images	3181/40000	8.0%

Next, we examine the ability of the full system; this system used the De-Rotation network to determine the correct orientation of each input image. First, we ensure that the network has not lost the ability to detect frontal, upright faces. This could happen if the rotation network does not correctly estimate rotations, and mistakenly suggests large rotations for face images which were initially upright. However, this was not the case. As can be seen in the first line of Table 2, there was only a very small decline in the ability to detect upright faces. Next, we check the ability of the network to make rotated faces upright. As can be seen in the second line of Table 2, the faces from the second image set, which contained rotated faces, were correctly labeled almost as often as the faces which were upright. Therefore, the De-Rotation network performed well in determining the rotation of the face.

Table 2. Results of using the De-Rotation network to pre-process all of the examples.

	Number Detected	Detection Rate
Original Face Images	3429/4000	85.7%
Rotated Face Images	33971/40000	84.9%

Finally, we check the performance of the full system to correctly label non-faces. Recall that during training the face-detector, false-positive images are gathered by collecting the mistakes the network makes by labeling faces in images that are known to contain no faces. These false-positives are added to the network's training set, and training is continued. At the end of training, this false-positive set is fairly large, and is comprised of a set of images that were difficult for the network to correctly classify as face or non-face. In the face-detector presented in [11], several networks were trained for face detection; each was individually trained by the process described above. Here, we use the false-positive training set of two of the networks trained in [11] to test the accuracy of the detector (note that the network tested here *was not* trained on this set of images). This set of images comprises a difficult set of images to correctly classify, and is *not* representative of the distribution of actual 20x20 images seen in natural images, since it is chosen to be a set of images that were difficult to classify by a network.

Table 3. Comparative Results on Non-Faces

	Number Mistakenly Detected	False Positive Rate
Original System from [11]	2252/11150	19.6%
System with De-Rotation	2570/11150	22.3%

Note that the original system falsely detects approximately 2252 out of this set of 11150 faces (a false detection rate of approximately 19.6%). When the system with the De-Rotation network is used, the false detection rate increases, see Table 3. This increase in error occurs because even images which do not appear similar to upright faces in their original orientation are rotated so that they look as much as possible like upright faces. This is because the De-Rotation network does not have knowledge of whether the input image is a face or not; it rotates the image to make it appear as an upright face. Therefore, the final, rotated, images which are passed to the detector can be much more difficult to classify correctly . Some sample non-face images which were not detected as faces by the original system, but were detected after rotation, are shown in Figure 6. This problem is returned to in the next section.

Fig. 6. Misclassification due to using rotation pre-processing. 20 pairs shown. Top of each pair is unrotated version, which was *not* mislabeled as face. Bottom: rotated version, which *was* mislabeled as a face.

In summary, the De-Rotation and Detection networks work well together. There is little loss of accuracy in detecting upright faces, and a large increase in accuracy in detecting rotated faces. There is a potential problem of mistakenly classifying more non-faces as faces than if rotation had not been used. A method for overcoming this problem is briefly mentioned in the next section.

5 Summary and Future Directions

We have proposed a simple and effective technique for extending existing template-based upright face-detection methods to efficiently detect faces which are rotated within the image-plane. Preliminary empirical results are promising.

There are three immediate directions for future work. The first is retraining the detection networks. At run-time, the distribution of images which the detection network is given is one which has been preprocessed by the De-Rotation network. However, in training, this preprocessing was not done. As was shown in Table 3, this can have an adverse affect on the ability to correctly classify non-faces. To avoid this problem in the future, the detection network should be trained with only examples which have been preprocessed with the De-Rotation network.

This system is much faster than the alternative of rotating the image multiple times, and applying the detector to each rotated image. However, the speed of this system is not yet up to the same performance as the "fast" version of the system presented in [11]. It is suspected that similar techniques which made that system fast can be applied here. The gain in speed was achieved by using a fast, but inaccurate, "candidate" detector network to eliminate regions which had no faces, and using the accurate, but slower, detection networks only on the regions which were not eliminated.

Finally, it would be interesting to extend this system to out-of-plane rotations. By using knowledge of the shape and/or symmetry of the face, it should be possible to convert semi-profile views of a face to frontal views. Related work has been explored in [1, 14].

Acknowledgements

I would like to thank Henry Rowley and Takeo Kanade for collaborating on the original face detection system. I would also like to thank Henry Rowley for pointing out the existence of several face detectors of which I was not aware. This work was done at Justsystem Pittsburgh Research Center (JPRC). This report was originally released internally, in preliminary form, at JPRC in July, 1997. The views and conclusions contained in this document are those of the author, and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of JPRC, Justsystem, or Carnegie-Mellon University.

References

1. David Beymer, Amnon Shashua, and Tomaso Poggio. Example based image analysis and synthesis. Technical Report A.I. Memo 1431, MIT, November 1993.
2. Gilles Burel and Dominique Carel. Detection and localization of faces on digital images. *Pattern Recognition Letters*, 15:963–967, October 1994.

3. Antonio J. Colmenarez and Thomas S. Huang. Face detection with information-based maximum discrimination. In *Computer Vision and Pattern Recognition*, pages 782–787, 1997.
4. John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1991.
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, Cambridge, Massachusetts, June 1995. IEEE Computer Society Press.
6. Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object detection. In *Fifth International Conference on Computer Vision*, pages 786–793, Cambridge, Massachusetts, June 1995. IEEE Computer Society Press.
7. Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In *Computer Vision and Pattern Recognition*, pages 130–136, 1997.
8. Alex Pentland, Baback Moghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. In *Computer Vision and Pattern Recognition*, pages 84–91, 1994.
9. Dean Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. PhD thesis, Carnegie Mellon University, Feburary 1992. Available as CS Technical Report CMU-CS-92-115.
10. Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visual scenes. Technical Report CMU-CS-95-158R, Carnegie Mellon University, November 1995. Also available at http://www.cs.cmu.edu/~har/faces.html.
11. Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.
12. Kah-Kay Sung. *Learning and Example Selection for Object nad Pattern Detection*. PhD thesis, MIT AI Lab, January 1996. Available as AI Technical Report 1572.
13. 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.
14. Thomas Vetter, Michael J. Jones, and Tomaso Poggio. A bootstrapping algorithm for learning linear models of object classes. In *Computer Vision and Pattern Recogition*, pages 40–46, San Juan, Puerto Rico, June 1997.
15. Gaungzheng Yang and Thomas S. Huang. Human face detection in a complex background. *Pattern Recognition*, 27(1):53–63, 1994.

This article was processed using the LaTeX macro package with LLNCS style