Face Detection using Neural Network

Kalavdekar Prakash N.
SRES, COE, Kopargaon-423603.
Dist. Ahemadnagr, Maharashtra.

ABSTRACT
This paper describes a face detection framework that is capable of processing images extremely rapidly while achieving high detection rates. As continual research is being conducted in the area of computer vision, one of the most practical applications under vigorous development is in the construction of a robust real-time face detection system. Successfully constructing a real-time face detection system not only implies a system capable of analyzing video streams, but also naturally leads onto the solution to the problems of extremely constraint testing environments. Analyzing a video sequence is the current challenge since faces are constantly in dynamic motion, presenting many different possible rotational and illumination conditions. While solutions to the task of face detection have been presented, detection performances of many systems are heavily dependent upon a strictly constrained environment. The problem of detecting faces under gross variations remains largely uncovered. This paper gives a face detection system which uses an image based neural network to detect face images.

Key Words
face detection system, real time, neural network

1. INTRODUCTION
Face to Face communication is a real time process operating at a time scale. The level of uncertainty at this time scale is considerable, making it necessary for humans & machines to rely on sensory rich perceptual primitives rather than slow symbolic inference process. Because of real time bandwidth & environmental constraints, video processing has to deal with much lower resolution & image quality, when compared photograph processing. Video images can be easily acquired & they can capture the motion of person, so its make possible to track people until they are in a position convenient for recognition.
The face is the most distinctive and widely used key to a person's identity. The area of Face detection has attracted considerable attention in the advancement of human-machine interaction as it provides a natural and efficient way to communicate between humans and machines. The problem of detecting the faces and facial parts in image sequences has become a popular area of research due to emerging applications in intelligent human-computer interface, surveillance systems, content-based image retrieval, video conferencing, financial transaction, forensic applications, pedestrian detection, and image database management system and so on. Face detection is essentially localising and extracting a face region from the background. This may seem like an easy task but the human face is a dynamic object and has a high degree of variability in its appearance, which makes face detection a difficult problem in computer vision.

1.1. System overview
The face detection system designed as shown in Fig.1

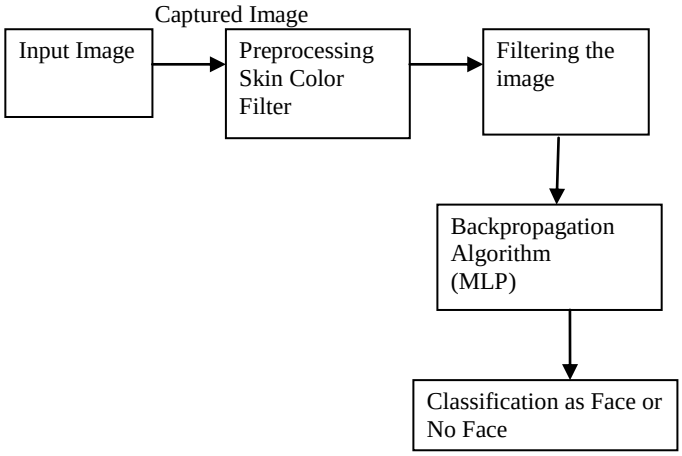

Fig 1: Face Detection

1.1.1 Skin color filter (Preprocessing)
The first step in preprocessing a color image consists of passing it through a skin color filter that detects the skin pixels. This is used to discard many of the pixels in the case of color images, thus reducing the amount of comparisons between the window and the image.

1.1.2 Filtering the image
This part consists of continuously applying a mask of 20*20 pixels to the preprocessed image. The mask is to some degree invariant to rotation and scale.

1.1.3 Multilayer Perceptron (MLP)
The prenetwork is a single multilayer perceptron (MLP). This is a neural network with input, hidden, and one output neurons (the output neuron is responsible for outputting either a face or a nonface). The prenetwork is trained using back-propagation. This filter eliminates many of the pixels to be considered in the comparison and is applied directly to grayscale images. For color images, the output of the skin filter is fed to the MLP.

1.1.4 Detection
The output of the neural network varies between 1 and -1 according to it whether a face has been detected or not, respectively

2 Implementation methods
2.1 Skin color filter
The first step in preprocessing a color image consists of passing it through a skin color filter that detects the skin pixels. This is used to discard many of the pixels in the case of color images, thus reducing the amount of comparisons between the window and the image. The first step in designing a skin color filter consists of changing the image from RGB to HSV, where 'H' stands for Hue, 'S' for Saturation and 'V' for

value. This reduces the effect of illumination. H, S, and V are continuous values varying between 0 and 1. Quantization is applied for both H and S to get discrete values. 10 points have been considered for each of the previously mentioned parameters.

Then, a color histogram is formed of H, S, and the pixel value. Thus, pairs of H and S are formed and for each of them, the corresponding number of pixels is determined. This allows us to get the first condition for a pixel to be a skin pixel. In fact, the color histogram (H, S) is compared to a skin threshold (determined empirically). If it is greater, the pixel can be classified as a skin pixel if it satisfies the second condition (discussed later). Otherwise, the pixel is rejected for being a nonskin pixel. The second condition consists of comparing the edge at each pixel with an edge threshold (determined empirically as well). The edge value is obtained by computing the gradient image using Sobel operator. This is useful for detecting edges in the image. If the computed edge at the pixel is less than the threshold, and the first condition has been satisfied, the pixel is classified as a skin pixel (and set to white). Otherwise, it is set to black.

The algorithm of the skin color filter can be summarized as follows:

1. Transform the image from RGB to HSV.
2. Compute the HSV values for each pixel and the color histogram (H, S).
3. Compute the gradient of RGB image using Sobel operator.
4. if (color histogram (H, S) > skin threshold and edge (x, y) < edge threshold)

Pixel (x, y) = 1 (white) ⇔ skin pixel
Pixel (x, y) = 0 (black) ⇔ nonskin pixel

2.2 Multilayer Perceptron (MLP)

The prenetwork is a single multilayer perceptron (MLP). This is a neural network with input, hidden, and one output neurons (the output neuron is responsible for outputting either a face or a nonface). The prenetwork is trained using back-propagation. This filter eliminates many of the pixels to be considered in the comparison and is applied directly to grayscale images. For color images, the output of the skin filter is fed to the MLP.

3 Detection

3.1 Filtering the image

This part consists of continuously applying a mask of 20*20 pixels to the preprocessed image. The mask is to some degree invariant to rotation and scale. The output of this operation (output of the neural network) varies between 1 and -1 according to whether a face has been detected or not, respectively. If the face in the original (preprocessed) image is larger than the window size, the image is sub-sampled (i.e., its size is reduced) and the filter is applied to the image at each size until the new face fits the mask.

First step:

At each step, another processing of the image is done to correct its illumination. This is done by first creating a function that varies linearly with the intensity inside the window. More precisely, the function varies linearly inside an oval in the window, and the outer contour is black to discard the background pixels. This transformed version of the image is then subtracted from the original one. Once this lighting correction has been done, histogram equalization is applied to the image to emphasize its contrast. As before, equalization is done in the oval part of the window. This is done to make sure that all images have the same properties regardless of the conditions under which they were taken and of the type of camera used.

Second step:

The extracted window is fed to the input layer of the neural network that determines whether the image contains a face or not. The hidden layers of the network consist of three types of units, with each type being specialized in one task. The first type is a set of four receptive fields (hidden units) that are responsible for detecting features such as the individual eyes, the nose, and the corners of the mouth. These units look at 10*10 pixel regions. The second category consists of 16 units that look at 5*5 pixel regions and have the same job as the ones described above. The third type is constituted of 6 units that look at 20*5 pixel regions and are responsible for detecting the mouth and the pair of eyes. This is possible since the units are horizontal.

Third step:

If the output of the network is 1, a face is detected. The opposite occurs for an output of -1. In order to train the system, a set of face and nonface images were used. Some features such as the eyes, the nose and the mouth were labeled, and the images were scaled and rotated using the following algorithm:

1. Initialize F, a vector that will be the average positions of each labeled feature over all the faces, with the feature locations in the first face F1.
2. The feature coordinates in F are rotated, translated, and scaled, so that the average locations of the eyes will appear at predetermined locations in a 20*20 pixel window.
3. For each face i, compute the best rotation, translation, and scaling to align the face's features Fi with the average feature locations F. Such transformations can be written as a linear function of their parameters. Thus, we can write a system of linear equations mapping the features from Fi to F. The least squares solution to this over-constrained system yields the parameters for the best alignment transformation. Call the aligned feature locations F'i.
4. Update F by averaging the aligned feature locations F'i for each face i.
5. Go to step 2.

The selection of nonface images during training is done as follows:

1. Create an initial set of nonface 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 nonface examples. The training algorithm is standard error backpropagation with momentum. 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.

3.2 The merge of overlapping detections and arbitration

First step: merging overlapping detections

Most faces are detected at many nearby locations, and therefore, the final detection of the face consists of taking all these detections and combining them to find the true position of the face in the image. For each location found, the number

of nearby detections is determined, and compared to a given threshold. If the number of detections is greater than the threshold, a face is correctly detected. Otherwise, this is a false detection. The location of the final detection is given by the centroid of all the nearby detections. This allows the different detections to be merged to give the final one. Once a face has been detected using the above approach, all the other detections are considered as errors and as such, are discarded. The only detected part we keep from the image is the one with a high enough number of detections within a small neighborhood.

Second step: arbitration among multiple networks

The above step is helpful in reducing the number of false detections (also called false positives). To reduce this number even further, a second step can be added, which consists of applying many networks and arbitrating between their outputs. Each detection at a particular position and scale is saved in an output pyramid and the outputs of different pyramids are ANDed together. When the outputs are ANDed, the detected part of an image will be correctly classified as a face if both networks agree upon it. Since it is rare that two networks will misclassify the faces, this strategy is helpful in decreasing the number of false detections. However, this strategy might reject a correctly identified face if only one of the two networks detects it.

4. RESULTS

The training sets are given of with frontal faces & are only roughly aligned. This was done by having a person place a bounding box around each face just above the eyebrows and about half-way between the mouth and the chin. This bounding box was then enlarged by 50% and then cropped and scaled to 20 by 20 pixels.

By observing the performance of this face detector on a number of test images. It was noticed a few different failure modes. The face detector was trained on frontal, upright faces. The faces were only very roughly aligned so there is some variation in rotation both in plane and out of plane. Informal observation suggests that the face detector can detect faces that are tilted up to about ±15 degrees in plane and about ±45 degrees out of plane (toward a profile view). The detector becomes unreliable with more rotation than this. Also noticed that harsh backlighting in which the faces are very dark while the background is relatively light sometimes causes failures. It is interesting to note that using a nonlinear variance normalization based on robust statistics to remove outliers improves the detection rate in this situation. Finally, this face detector fails on significantly occluded faces. If the eyes are occluded for example, the detector will usually fail. The mouth is not as important and so a face with a covered mouth will usually still be detected.

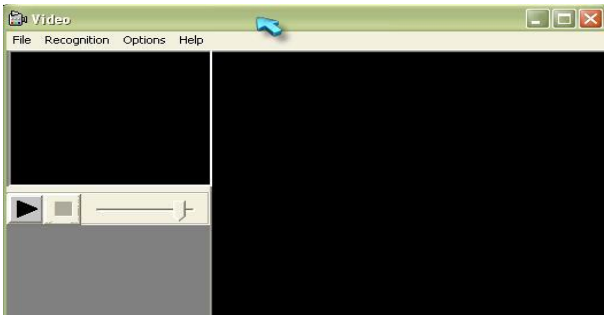

Fig. 2. Face detection in Video

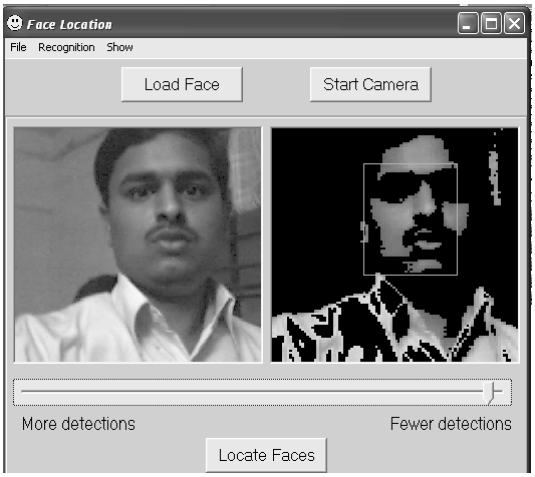

Fig 3. Result of locating single Face within picture

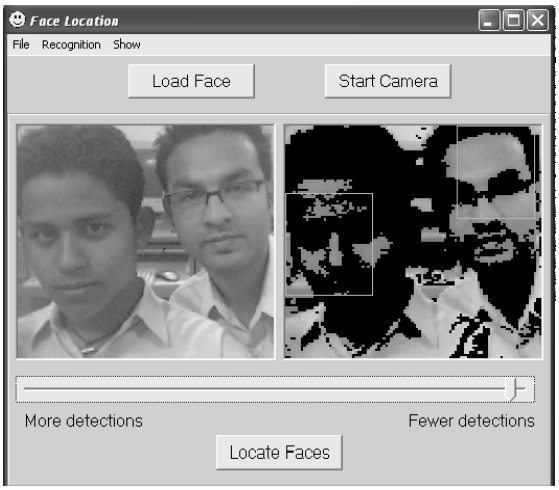

Fig 4 Result of Locating Multiple Faces within picture

Fig 5. Locating Non-Frontal Faces

Fig 6 Result of Locating Faces in Low Light Conditions

5. CONCLUSION

Here a presented an approach for face detection which minimizes computation time while achieving high detection accuracy. The approach was used to construct a face detection system which is approximately 5 times faster than any previous approach. The system is tested with video 15 FPS & till giving result. The system is also tested for multiple images & with some angle.

6. REFERENCES

[1] Milan Sonka, Vaclav Hlavac, Roger Boyle"Image Processing, analysis and Machine Vision", Tata McGraw-Hill ISBN 981-240-061-3

[2] Rowley, Baluja, and Kanade: "Neural Network-Based Face Detection" *IEEE Patt. Anal. Mach. Intell.*, 20:22–38.

[3] Viola & Jones: "Robust Real-Time Face Detection" International Journal of Computer Vision 57(2), 137-154, 2004.

[4] Maya Choueiri, Nassib El-Sayegh, and Wassim Said "Real-Time Face Detection and Recognition"

[5] Robi Polikar, Lalita Udpa, Satish S. Udpa and Vasant Honava, " Learn++: An Incremental Learning Algorithm for Supervised Neural Networks" IEEE Transactions on Systems, man, and cybernetics—part c: Applications and Reviews, vol. 31, no. 4, november 2001.

[6] S. M. Bileschi B. Heisele," Advances in Component Based Face Detection ",Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures (AMFG'03) 0-7695-2010-3/03 $ 17.00 © 2003 IEEE.

[7] Bernd Heisele**,"** Visual ObjectRecognition with Supervised Learning", 1094-7167/03/$17.00 © 2003 IEEE.