Face recognition

Axel Beauvisage, Carole Belloni, Adrian Kostkowski, Gareth Hughes, Domonkos Huszar

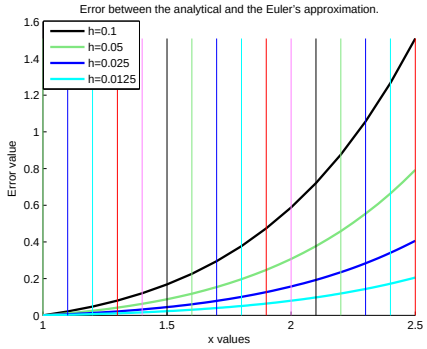


Fig. 1. Simulation Results

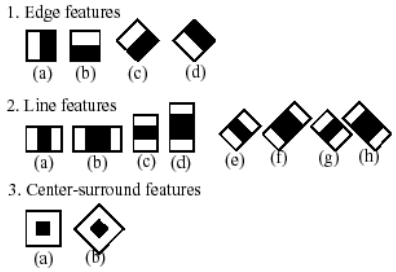


Fig. 2. Haar features used in openCV (including 45 features)

***Abstract*—**

I. Introduction

THIS demo file is intended to serve as a "starter file" for IEEE journal papers produced under LaTeX using IEEEtran.cls version 1.7 and later. I wish you the best of success.

mds
January 11, 2007

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Methods

A. Normalization and Haar feature-based cascade classifier

Preprocessing of the image is needed in order to have a good recognition. We'll use for this purpose equalized gray images to a better detection of faces during the normalization. We use a Haar features-based cascade to detect the face. The output will be the difference between the sum of the intensities on the whole picture and the intensities on the black part of the feature. The feature are scaled and move across the initial image. These are quickly calculated by using the corresponding intensity image. We use indeed two intensity images : one adding

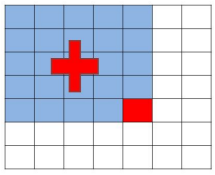


Fig. 3. Integral image : sum of the intensities of the value of left and up pixels of the current pixel

the left and up pixels of the analyzed pixel and an other intensity image adding the pixels in the corner of the current pixel made by two 45 lines. The cascade is then trained. Adaboost is used for this purpose on positive images (a face, an eye, a nose, a mouth). The result is a sequences of criteria of recognition following a cascade checking of the object. However, Haar will only recognize not rotated face. If we do not recognize a face, the face will be rotated and the same Haar process will be used. Once a face is detected we'll use again Haar features to detect the eyes or the mouth and nose. The angle remaining to have the eyes or nose and mouth aligned is then used to have these features aligned. The face is then cropped to the dimensions of the face detected by Haar features and resize to a specified size, so that every picture has the same

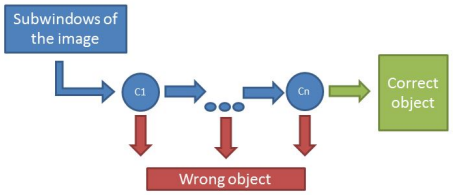


Fig. 4. Cascade

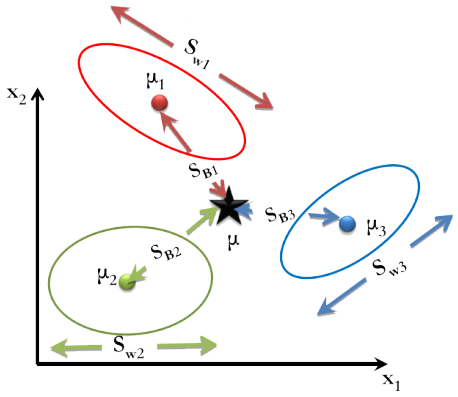


Fig. 5. Az LDA algoritmus ltal megvizsglt szrsok. The source of the image: [1]

size.

B. Fisher method

Ha m dimenzis vektorokat szeretnk lekpezni egy C-dimenzis trbe akkor azt a kvetkez mdon tehetem meg.

$$y = w^T x, \text{ where} \tag{1}$$

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} w = \begin{pmatrix} w_{1,1} \dots w_{1,C} \\ \vdots \ddots \vdots \\ w_{m,1} \dots w_{m,C} \end{pmatrix} y = \begin{pmatrix} y_1 \\ \vdots \\ y_C \end{pmatrix} \tag{2}$$

C. Eigenfaces method

The Eigenfaces method is based on a principal components analysis from a training set of images. To get those, each normalized image is translated into a vector. The principal components will be then determined : They have the highest eigenvalue in the diagonalized correlation matrix (made out of the image and the mean values of the training set).

After selecting the principals components (first eigenvectors called eigenfaces in this case), each image can be then be approximated with those weighed eigenfaces :

$$x \approx \hat{x} = \bar{x} + E_x * W, \text{ where} \tag{3}$$

x = vector representing the image,
$\hat{x}$ = approximated vector,
$\bar{x}$ = mean values of the vector during training,
E_x = Matrix of eigenfaces (principal components),
W = Matrix of weights of eigenfaces

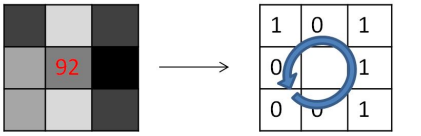
W is particular to each image and is calculated to minimize the distance between x and $\bar{x}$:

$$W = E_x^T (x - \bar{x}) \tag{4}$$

The face recognized will be then the one in the training having its weight matrix W the closest to the weight matrix of the current face analyzed.

D. Local Binary Pattern Histogram

The local binary pattern histogram method uses the local feature of the face. First, the image is sliced into squares. To avoid problems of scale, The neighbours of a pixel analysed will be the ones on a circle with a varying radius and this pixel as center. The intensity of the central pixel is then compared to the ones of its neighbours. (higher values become 1, lower ones become 0). We then compute the new value of the central pixel, being the sum of powers of two combined with the value of neighbor taken clockwise. The point is to spot particular patterns of neighbor such as edges, lines, corner, flat areas.



0x1+0x2+1x4+1x8+1x16+0*32+1*64+0*128 =92

Fig. 6. Analysing the local structure

The histograms are then added (and not merged) for each part of the image. The resulting vector is then compared to the vectors obtained during the training. A fixed number of the closest vectors

to the resulting vector is chosen and the person assigned to the greater number of these vector will be associated with the initial image.

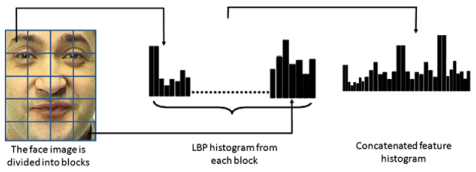


Fig. 7. Adding histograms (source : http://what-when-how.com/face-recognition/)

E. RBF and Convolutional NeuralNetwork

NeuralNetwork stuff here

III. Implementation

A. The Graphical User Interface

To make our program simple and easy to use, we have created a Graphical User Interface(GUI). To do so we have used the Qt library and because it can generate problems on some computers, the program can also be run without GUI by specifying the training and testing folders in the command line. The GUI is divided into 3 main parts: the main window, the result dialogBox and the ImageCapture program.

1) The main window: It is the first window to appear when the program is executed. It contains the following functionalities:

- A menu to load and save classifiers in a file (the classifier selected in the list of methods will be used) and to run the ImageCapture program.
- A path selection tool to specify where is are the training and validation folders. The paths can be modified directly from the text inputs or by browsing it using the "..." button.
- A widget displaying the image retrieved from the webcam. A Rectangle has been added to help the user to center his face on the picture.
- An output console embedded in the window.
- A normalization checkbox to apply the normalization on the images or not
- A progressBar to make sure that the program is still running while loading the images (unfortunately it was not possible to do the same for the training).

2) The result dialogBox: When a new picture is taken by the user, a new result window is created. This dialogBox call the predict function from the selected classifier and display the results to the user(image, predicted label and confidence level).

3) The ImageCapture program: - Drag'n Drop square - Big cross for alignment - Space Bar to record

4) The Webcam object:

B. Dataset seperation

We have broken down our initial dataset into 2 folders: a training folder and a testing one.

IV. Results

The result goes here.

V. Conclusion

Before making a conclusion I will first wait for you to finish this paper.

some ideas : Face recognition is a tough problem But we manage to do something

Appendix A
Important code parts

Appendix two text goes here.

References

[1] A. A. Farag and S. Y. Elhabian, "A tutorial on data reduction linear discriminant analysis," 2008.