A Practical Implementation of Face Detection by Using Matlab Cascade Object Detector

Elena Alionte
Department of Automatic Control and Applied Informatics
Technical University "Gheorghe Asachi"of Iasi
Iasi, Romania
alionte.elena@ac.tuiasi.ro

Corneliu Lazar, *Member, IEEE*
Department of Automatic Control and Applied Informatics
Technical University "Gheorghe Asachi"of Iasi
Iasi, Romania
clazar@ac.tuasi.ro

Abstract: **The detection of faces in an image is a subject often studied in computer vision literature. The algorithm which allowed face detection, imposing new standards in this area, was the Viola – Jones algorithm. In this paper, a practical implementation of a face detector based on Viola-Jones algorithm using Matlab cascade object detector is presented. Employing the system type object *vision.CascadeObjectDetector*, eight face detectors were developed using the *trainCascadeObjectDetector* function and tuning the number of cascade layer and the False Alarm Rate. For different tuning parameters, the performances of the face detectors were analyzed.**

Keywords: Viola – Jones algorithm, face detection, AdaBoost, integral image, cascade object detector

I. INTRODUCTION

In many applications, such as driver face monitoring, face recognition, video surveillance, human computer interface or image database management, human face detection is an important and complex process. The complexity of the face detection algorithms is due to the variations in illumination, background, visual angle and facial expressions and the implementation is not easy [1].

Face detection algorithms are usually divided into two general categories [2]: (i) feature-based and (ii) learning-based methods. The algorithms from the first category are based on the assumption that face in the image can be detected based on some simple features, independent of ambient light, face rotation and pose. Thus, a simple method uses image projection to detect faces under the assumption that the background is uniform and with the vertical projection of the gray level image is determined the face position [3]. Another feature-based face detection approach is based on a skin color model determined by using the probability distribution in a color space. The face is detected in image by applying a threshold on the modeled distribution as in [4].The algorithms from the second category are more robust but they need a greater computational effort. Learning-based methods use a number of training samples and benefit from statistical models and machine learning algorithms. From this category, Viola-Jones face detector [5] is one of the most extensively used. The detector can be extended to other kinds of objects. The convergence of the training phase of this algorithm depends a lot on the training data.

The Viola-Jones face detector can run in real time because it is based on the following main ideas [6]:

- rapid computation of Haar-like features using the integral image;
- classifier learning with AdaBoost to select the best features;
- the attentional cascade structure which rejects the majority of the sub-windows in early layers of the detector, making the detection process extremely efficient.

Due to the simplicity of extracted features process and selection of the best features, Viola-Jones face detector is fast and robust, being reported many and various implementations for different applications. An implementation used successfully is the one in OpenCV. A complete algorithmic description of Viola-Jones face detection method, with a learning code and a learned face detector is presented in [7]. Another implementation incorporates six different types of feature images into the Viola and Jones' algorithm [8] to improve its performance. Computer Vision System Toolbox™ supports several approaches to object detection in an image or video, including the Viola-Jones algorithm [9]. The Viola-Jones algorithm uses Haar-like features and a cascade of classifiers to identify pretrained objects, including faces, noses, eyes, and other body parts. It is also possible to train a custom classifier.

In this paper is presented a practical implementation of a frontal view face detection algorithm based on Viola-Jones approach using Matlab cascade object detector. Employing the Matlab system object *vision.CascadeObjectDetector*, a face detector was developed configurated to use the user classification model specified in the XMLFILE input file. The file is created with the help of the *trainCascadeObjectDetector* function. The attentional cascade training is done using a set of positive samples (windows with faces) and a set of negative images. For obtaining a more accurate detector, the number of cascade layers and the function parameters were tuned. Finally, for different tuning parameters the performances of the face detector were analyzed.

978-1-4799-8481-7/15/$31.00 ©2015 IEEE

This paper is organized as follows. In Section 2, is presented Viola-Jones face detection algorithm. Section 3 illustrates the implementation of the Viola-Jones algorithm using Matlab cascade object detector. In section 4, we tested our proposal face detection system. Section 5 describes the conclusion and the future work.

II. VIOLA – JONES ALGORITHM

The Viola – Jones algorithm is intended for real – time detection of faces from an image. Its real – time performance is obtained by using Haar type features, computed rapidly by using integral images, feature selection using the AdaBoost algorithm (Adaptive Boost) and face detection with attentional cascade.

A. Feature calculation

Starting from the common characteristics of the faces, such as the region around the eyes is darker than the cheeks or the region of the nose is brighter than those of the eyes, five Haar masks (Fig. 1) were chosen for determining the features, calculated at different positions and sizes. Haar features are calculated as the difference between the sum of the pixels from the white region and the sum of the pixels from the black region. In this way, it is possible to detect contrast differences.

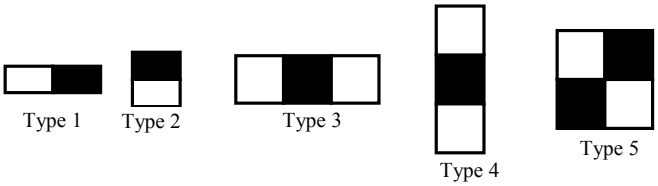

Fig. 1. Haar masks used

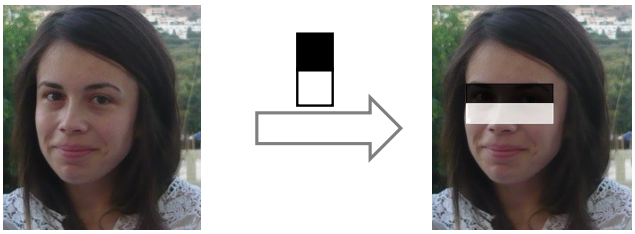

Fig. 2. Type 2 Haar feature from which the intensity difference between the pixels from eyes region and the cheek region can be observed

If we consider the mask M from Fig. 2, the Haar feature associated with the image I behind the mask is defined by:

$$\sum_{1\le i\le N}\sum_{1\le j\le N} I(i,j)_{white} - I(i,j)_{black} \qquad (1)$$

The features are extracted for windows with the dimensions of 24x24 pixels, which are moved on the image where we want to detect faces. For such a window, Haar masks are scaled and moved, resulting 162,336 of features. To reduce the computation time of the Haar features, which vary depending on the size and type of the feature, the integral image was used. In Fig. 3 is illustrated how from an original image is obtained the integral one and how is computed the sum of pixels within a rectangle region using integral image.

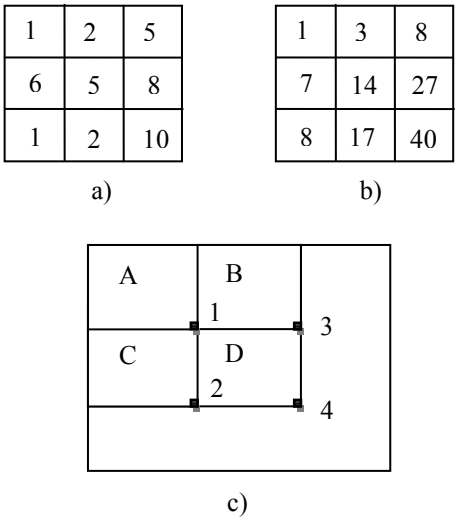

Fig. 3. Integral image: a) original image *I*; b) integral image *II*; c) pixel computation from the region D using integral image

For the location (*i, j*), the integral image *II* contains the sum of the pixels above and to the left of (*i, j*), inclusive:

$$II(i,j) = \sum_{1\le s\le i}\sum_{1\le t\le j} I(s,t), \quad 1\le i\le N, 1\le j\le N \qquad (2)$$

The sum of the pixels within rectangle can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A . The value at location 2 is A + C, at location 3 is A + B, and at location 4 is A + B + C + D . The sum within D can be computed as 4 + 1 – (2 + 3).

B. Feature selection using AdaBoost algorithm

As the number of Haar features for an image with 24 x 24 pixels is d = 162 336, and many of them are redundant, AdaBoost algorithm was used to select a smaller number of features. The basic idea is to build a complex classifier (decision rule) using a weighted linear combination of weak classifiers. Every feature f is considered a weak classifier, defined by:

$$h(x,f,p,\theta) = \begin{cases} 1, \ if \ \ pf(x) < p\theta \\ 0, \ otherwise \end{cases} \qquad (3)$$

where x is a 24 x 24 pixel image, θ is a threshold and p is a parity.

AdaBoost algorithm [10] is based on a training set which contains n pairs (x_i, y_i), where x_i is a positive or a negative image, and y_i is a label assigned to each image and is equal to 1 for a positive image and to -1 for a negative image. Each image is weighted with $w_i \in R_+$ and AdaBoost algorithm aims

to decrease losses defined by $\sum_{i=1}^{n} w_i 1_{yi \neq f(xi)}$ by adjusting the values of the weights w_i. AdaBoost algorithm consists in T iteration, from which T weak classifiers, meaning T features, will be selected. At each stage t of *boosting* is performed:

Step 1: with data $x_i(t-1)$ weighted from the previous phase, train all weak classifiers (d = 162 336) and choose the most efficient weak classifier h_t that will become a component of the strong classifier.

Step 2: Combine the weak classifier with other weak classifiers declared the most efficient in the previous phases.

Step 3: calculate the weighted error $e_t = \sum_{i=1}^{n} w_i 1_{ht(xi) \neq yi}$ and update the weights for iteration (t + 1) with the formula:

$$\forall i, \quad w_i(t+1) = \frac{w_i(t)}{2}\left(\frac{1}{e_t} 1_{ht(xi) \neq yi} + \frac{1}{1-e_t} 1_{ht(xi)=yi}\right) \qquad (5)$$

In this way, the next weak classifier will focus on harder examples from the training set.

Finally, the strong classifier will be a linear combination of T weak classifiers whose decision rule will be:

$$h(x) = \begin{cases} 1 \quad if \ \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, \ otherwise \end{cases} \qquad (6)$$

where the weights are $\alpha_t = \frac{1}{2} \ln\left(\frac{1-e_t}{e_t}\right)$. These weights will be larger for a weak classifier that has a small error and will be smaller for a weak classifier with a high classification error.

C. Attentional Cascade

After AdaBoost algorithm, a strong classifier will result that classifies the windows of *N*x*N* size well enough. Since, on average, only 0.01% of the windows are positive images, meaning faces, only potentially positive windows must be examined.

Instead, to achieve a higher detection rate and a smaller misclassified images detection rate, we should use another strong classifier that classifies correctly the before misclassified images. This creates the attentional cascade, as showed in Fig. 4. At the first layer of the attentional cascade, a strong classifier with few features is used, which will filter/reject most negative windows. A cascade of classifiers that are becoming more and more complex (with more features) will follow and they will allow to achieve a better detection rate. At each layer of the cascade, the negative images classified correctly will be eliminated and the new strong classifier will have a more difficult task than the previous step classifier.

Finally, the cascade of classifiers will operate as follows:

- the image will be split into multiple windows;
- every window is an input in the attentional cascade;
- at every layer, the window is checked if it contains a face or not – according to the strong classifier;
- if it is negative, the window is rejected and the steps will be repeated for another window;
- if it is positive, it means that the window is a possible face and will move to the next layer of the cascade;
- the window contains a face if it passes all layers of the attentional cascade.

III. Face Detector Implementation using Matlab

The Computer Vision Toolbox from the Matlab environment contains a cascade object detector (*vision.CascadeObjectDetector*) which creates a system object (detector) capable to detect objects using the Viola – Jones algorithm. By default, the detector is set to detect faces in an image, but it can also detect the nose, mouth, eyes or the upper part of the body defined by the input string MODEL (*ClassificationModel*). This paper presents a detector based on Haar type features, that builds a system type object, configurated to use the user classification model specified in the XMLFILE input file. The file is created with the help of the *trainCascadeObjectDetector* function

.

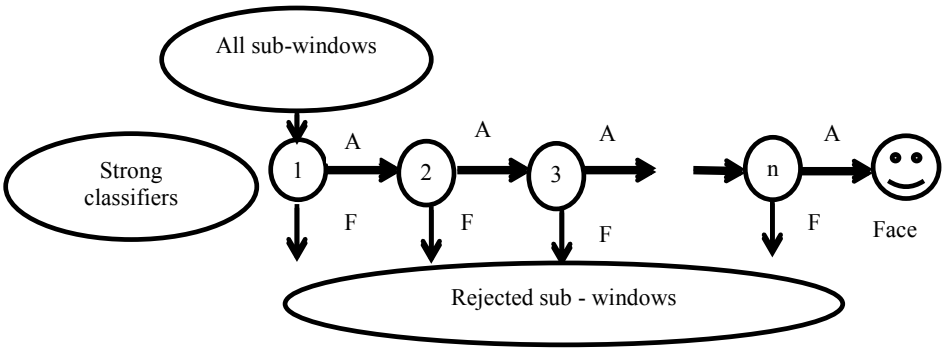

Fig.4 Attentional cascade

A. Training the detector using the trainCascadeObjectDetector function.

Because the system object from Computer Vision Toolbox *vision.CascadeObjectDetector* contains only a few pretrained classifiers, insufficient for a face detection application, the attentional cascade requires training for each user, using the *trainCascadeObjectDetector* function. The attentional cascade training is done using a set of positive samples (windows with faces) and a set of negative images. The negative samples are automatically generated from the set of negative images. For obtaining a more accurate detector the number of cascade layers, the feature type (Haar in our case) and the function parameters must be specified. The attentional cascade training (Fig.5) is done layer by layer, as follows:

- Train Layer One using:
  - Calculated number of positive samples, which is less than the total number of user-provided samples.
  - Generated negative samples from user-provided negative images.
- Train layer Two:
  - Use the results from layer one.
  - Classify all positive samples and discard samples misclassified as negatives.
  - Use the same calculated number of positive samples of the remaining in positive samples.
  - Generate negative samples by processing negative images with sliding window and using false-positive classified samples.
- Train layer N:
  - Use the results from the previous layer.
  - Classify all positive samples and discard samples misclassified as negatives.

The optional function parameters are:

- *ObjectTrainingSize:* the size of the objects used for training – is in the form of a vector with two elements. Before training the attentional cascade all negative and positive images will be resized to the values specified by the user.
- To have better results, the value of the 'ObjectTrainingSize' parameter should be as close as possible to the size of the object detected. To reduce the training duration the value of the 'ObjectTrainingSize' parameter must be smaller than the size of the object. The default value of the parameter is *auto* – the median width-length ratio will be calculated.
- *NegativeSamplesFactor:* the multiplication factor of the number of positive samples for obtaining the number of negative samples used in the training of the attentional cascade. The default value is 2.
- *FalseAlarmRate:* the false positive result rate (FalsePositiveRate) accepted for each layer of the attentional cascade. A false positive result represents a negative sample classified as a face. The parameter value can be a number in the interval (0, 1] and the default value is 0.5.
- *TruePositiveRate:* the positive result rate correctly detected for each layer. The parameter value can be a number in the interval (0, 1] and the default value is 0.995.
- *FeatureType:* the feature type used for training. The accepted types are "Haar", "LBP" and "HOG" and for this paper, the feature type used was "Haar".

The function parameters must be specified for achieving an optimal functioning detector. Setting the parameters is based on the following compromises:

- For a small training set - Decrease the number of layers and set a lower false positive rate for each layer.
- For a large training set (in the thousands) - Increase the number of layers and set a higher false positive rate for each layer.
- For reducing the probability of missing an object – it is recommended to increase the true positive rate.
- For reducing the number of false detections – it is recommended to increase the number of layers or decrease the rate of false positive results.

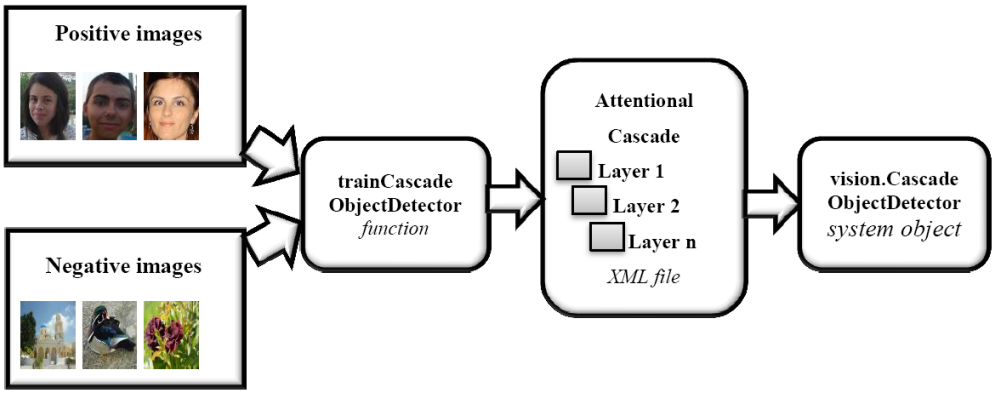

Fig. 5 The attentional cascade training

For training the detector, positive images with size of 24 x 24, which contain only the cropped face, are used. The positive samples will consist of an array of structures where the address of the positive sample and the region where the face is located – in this case, the whole image, will be retained.

The pseudocode used to obtain the positive samples is:

For each positive image

Put in openFileName <- the address of the positive image

Put in positiveInstances(number). ObjectBoundingBoxes <- [1 1 24 24]

Put in positiveInstances(number).imageFilename <- openFileName

The negative samples will be automatically generated from the negative images provided by the user. The negative images are specified by the path of their folder:

negativeFolder = 'D:\matlab2015\imagininegative\';

The execution of the trainCascadeObjectDetector function which trains a detector with the False Positive Rate equal to 0.3 and the number of layers equal to 16, will be as follows:

trainCascadeObjectDetector('Haar500V2.xml',positiveInstances,negativeFolder,'NumCascadeLayers',16,'FalseAlarmRate',0.3,'FeatureType','Haar');

B. Face detector based on Viola – Jones algorithm

For creating the system object (detector) that detects faces from an image, using the Viola – Jones algorithm, the following command is used:

detector = vision.CascadeObjectDetector('attentionalCascade.xml'),

where the only parameter is represented by the name of the xml file in which the attentional cascade was saved.

After the creation of the detector, the step method is called by the following syntax:

BBOX = step(detector, I)

which returns BBOX, an *M* – by – 4 matrix *defining* *M* bounding boxes containing the detected objects. Each row contains 4 elements [*x y width height*], that specify in pixels, the upper-left corner and size of a bounding box.

In order to use a detector obtained from training, the following steps are done:i) open the desired image; ii) create the detector object; iii) identify faces from the image; iv) annotate the faces; v) show image with annotated faces.

IV. EXPERIMENTAL RESULTS

The proposed face detection algorithm based on Viola-Jones was implemented using Matlab cascade object detector with different setting parameters of the Matlab function *trainCascadeObjectDetector,* resulting eight face detectors. The performances of these detectors were analyzed using 3 different images: I1 with 6 faces (2128x1416px), I2 with 11 faces (1920x1080px) and I3 with 6 faces (1752x1360px).

We have trained the detectors, providing 1000 positive images and 1000 negative images. We started with the default parameters given in Section III, resulting the detector D1. After that, we decreased/increased the number of layers (NL) and/or the False Alarm Rate (FAR), obtaining detectors D2-D8.

The results obtained with the eigth trained detectors for the 3 images are illustrated in Table 1, where the number of detected faces (NDF) is given for every detector.

Table 1. Results of face detection

Detector	NL	FAR	I1-NDF	I2-NDF	I3-NDF
D1	20	0.5	8	13	11
D2	16	0.5	48	97	117
D3	22	0.5	6	10	8
D4	26/23	0.5	7	12	6
D5	22	0.7	282	175	236
D6	26	0.7	64	94	117
D7	22/14	0.3	7	14	6
D8	26/14	0.3	6	12	6

There were some detectors that have stopped from training because more negative images were needed. The resulting detectors (D4, D7, D8) had less stages than the wanted number (table 1).

Starting from the default patrameters, better results were obtained by increasing the number of layers and decreasing the False Alarm Rate. The best results were obtained with detector D8 and are shown in Fig. 6-8 for images I1-I3.

Fig.6 Face detection with D8 for I1

From Table 1, it is observed that keeping constant FAR = 0.5, the results are improved by increasing the number of levels from NL = 16 (D2) to NL = 26/23 (D4). A high value of the False Alarm Rate (FAR = 0.7) leads to poor results worsen along with reducing the number of levels (see results for D5

and D6 in Table 1). Notable results were obtained with a low value of the the False Alarm Rate (FAR = 0.3) and the best for a high value of NL (see results for D7 and D8 in Table 1).

Fig.7 Face detection with D8 for I2

Face detection is a difficult task due to the many variations in scale, location, orientation, pose, facial expression, lighting conditions, occlusions etc. For example, in the image I2 people are not placed in the foreground and therefore the results obtained with the 8 detectors are not the best, never obtaining the detection of only 11 faces.

Fig.7 Face detection with D8 for I3

Having in view that the goal of face detection is to determine whether or not there are any faces in an image and, if present, return the image location and extent of each face in real time, the response time of the detector is an important performance.

Table 2 Time response (sec) of face detectors

	D1	D2	D3	D4	D5	D6	D7	D8
I1	2.2	2.54	1.79	1.68	3.68	2.27	1.92	1.54
I2	1.55	1.83	1.23	1.28	2.61	1.76	1.27	1.09
I3	1.07	2.56	1.75	1.47	4.79	2.71	1.63	0.83

As seen in Table 2, the response time of the 8 face detectors D1-D8 with different tuning parameters is satisfactory to an application of real-time face detection.

V. Conclusions

Using *vision.CascadeObjectDetector*, a Matlab object system, a face detector based on Viola-Jones algorithm has been developed. Starting from several pretrained classifiers for detecting frontal faces, we used the *trainCascadeObjectDetector* function to train our face detector classifier, employing a set of positive samples and a set of negative images. Selecting the function parameters, we optimized the number of layers, the false positive rate and the true positive rate, resulting more detectors. Using three different images with 6, 11 and 6 faces, all detectors were tested. To get better results with a higher positive rate, more images are required in the training process.

REFERENCES

[1] Cha Zhang, Zhengyou Zhang, "Boosting-based face detection and adaptation", Synthesis Lectures on Computer Vision, Vol. 2, No. 1 , Pages 1-140, Morgan & Claypool Publishers, 2010.

[2] M. H. Yang, D. J. Kriegman and N. Ahuja, "Detecting faces in images: a survey", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 1, 2002.

[3] F. Wang and H. Qin, "A FPGA based driver drowsiness detecting system", Proceedings of IEEE International Conference on Vehicular Electronics and Safety, Xian, October, 2005.

[4] J. Batista, "A drowsiness and point of attention monitoring system for driver vigilance", Proceeding of IEEE Intelligent Transportation Systems Conference, , Seattle, USA, October, 2007.

[5] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", Proceeding of International Conference on Computer Vision and Pattern Recognition (CVPR), Kauai, HI, USA, 2001.

[6] P.Viola and M. Jones, "Robust real-time face detection", International Journal of Computer Vision, 57(2), 2004, pp 137–154.

[7] Y. Wang, "An alasysis of the Viola – Jones face detection algorithm", Image Processing On Line, 4, 2014, pp. 128 – 148

[8] Qian Li, Niaz, U., Merialdo, B., "An improved algorithm on Viola-Jones ooject detector", 10th International Workshop on Content-Based Multimedia Indexing (CBMI), Annecy, 27-29 June, 2012, pp.1-6 .

[9] http://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html

[10] R. E. Schapire, "A brief introduction to boosting" Proc. of the Sixteenth International Joint Conference on Artificial Intelligence, 1999.