Towards Enhancing the Face Detectors Based on Measuring the Effectiveness of Haar Features and Threshold Methods

Nidal F. Shilbayeh[*], Khadija M. Al-Noori[**], Asim Alshiekh[*]

[*]University of Tabuk, Faculty of Computers and Information Technology, Tabuk, Saudia, Arabia
nshilbayeh@ut.edu.sa, aalshiekh@ut.edu.sa
[**]Middle East University, Faculty of Information Technology, Amman, Jordan
Kmk-84@yahoo.com

Abstract: - Face detection has been regarded as the most complex and challenging problem in the field of computer vision, due to the large intra-class variations caused by the changes in facial appearance, lighting, and expression. Face detection is the essential first step towards many advanced computer vision, biometrics recognition and multimedia applications, such as face tracking, face recognition, and video surveillance. One of the most famous approaches that is successful is the Viola & Jones algorithm. In this paper, systems were designed based on this approach to measure the effectiveness of the different Haar feature types, and to compare two types of threshold computing methods. The two methods used for computing thresholds are the average of means and the optimal threshold methods. There are 8 different Haar features has been used in building these systems. The implemented systems have been trained using a handpicked database. The database contains 350 face and nonface images. Adaboost algorithm has been used to build our detectors. Each detector consists of 3 cascade stages. In each stage, we randomly use a number of weak classifiers to build the strong classifier. Each weak classifier is computed based on threshold before entering the Adaboost algorithm. If the image can pass through all stages of the detector, then the face will be detected. The detectors have been tested using the CMU+MIT database. Some recommendations have been suggested according to the Haar features and the computed threshold to improve the face detection of Viola Jones approach.

Key-Words: - Face Detection, Haar-Like Features, Pattern Recognition, Weak Classifier, Integral Image, Strong Classifier, Adaboost Algorithm.

1 Introduction

Face detection is a computer technology that determines if there are any faces in arbitrary images and identifies: location, size, and content of each human face. It also detects the facial features and ignores anything else, such as: buildings, trees, animals and bodies.

Human face detection is an active area of research covering several disciplines such as: image processing, pattern recognition and computer vision. Face detection is the first step in any automated system, which solves Face recognition or face identification, face authentication, face tracking, facial expression recognition, and face localization. It's also the first step of any fully automatic system that analyzes the information contained in faces (e.g., identity, gender, expression, age, race and pose). Face detection is used in a lot of applications, such as a part of a facial recognition system, video surveillance, human computer interface, image database management, and newer digital cameras use face detection for autofocus and bodies [10-12].

Face detection is considered a part of object detection as in [1-3]; Object detection and classification holds the key to many other high level applications such as: face recognition, human computer interaction, security and tracking among others.

2 Literature Review

Face detection is a computer technology that has received a lot of interest in the last few years. In the last ten years, face detection and facial expression recognition have attracted much more attention, even though they had been studied for more than 30 years by psychophysicists, neuroscientists, and engineers. Face detection is one of the most active areas in computer science, so there are is a lot of effort and researches in this area.

It is known that isolated pixel values cannot give any information except the luminance and/or the color of the radiation received by the camera at a given point. Therefore, there are two motivations for using features instead of the pixel intensities directly. First, features encode domain knowledge is better than pixels, so the features help to encode some information about the class to be detected. The second reason is that a Feature-Based System can be much faster than a Pixel Based System [1-2].

One of these features is the Haar feature, which encodes the existence of oriented contrasts between regions in the image. A set of these features can be used to encode the contrasts exhibited by a human face and their special relationships.

2.1 Haar-Like Features

Haar features, are represented by a template (shape of the feature). Each feature is composed of a number of "black" and "white" rectangles joined together. After the approach of Viola & Jones succeeded, an extended set of Haar-like features are added to the basic feature set. There are more than 15 kinds (or prototypes) of Haar feature types. Fig.1 shows the basic Haar features , and the fifteen extended Haar features respectively .

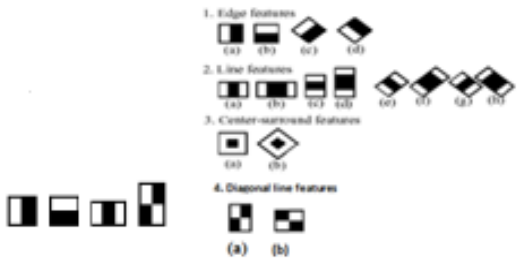


Fig.1 A set of basic Haar features and Extended Haar features

To obtain the value of a Haar-like feature, it is computed as the difference between the sums of the pixel gray level values within the black and white rectangular regions. This is done by subtracting the pixels covered by white rectangles from the sum of the pixels covered by black rectangles as in eq.1.

$$fx = Pixels\ in\ white\ area - Pixels\ in\ black\ area \quad (1)$$

2.2 Integral Image

It is a new image representation, "Integral Image" is similar to the "Summed Area Table" (SAT) idea which is used in computer graphics for texture mapping. It can be defined as 2-dimensional "look up table" in the form of a matrix with the same size of the original image.

The integral image's value at each pixel (x,y) could be computed by summing the values of the pixels above and to the left of (x,y). However, it can quickly be computed in one pass through the image and can be calculated by using eq.2 and fig.2:

$$P(x,y) = \sum_{x' \le x, y' \le y} i(x', y') \quad (2)$$

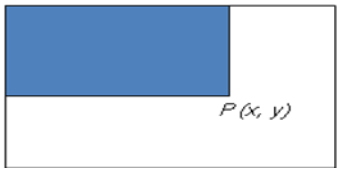


Fig.2 Integral Image for point (x,y)

2.3 Adaboost Algorithm

Boosting is an efficient classifier, which converts a weak classifier to a strong one by combining a collection of weak classifier to form a strong one.

The adaptive boosting (Adaboost) algorithm exists in various varieties[3]. In addition, there are three modifications of the original algorithm that were proposed: Gentle-, Logit-, and Real Adaboost [4].

The aim of boosting is to improve the classification performance of any given simple learning algorithm. It is used to select rectangle features and combine them into an ensemble classifier in a cascade node, which is used to reduce the training time.

2.4 Weak Classifier and Threshold

Weak classifiers are constructed using one or a few Haar features with trained threshold values. In most papers, one feature for every weak classifier is used. To determine the weak classifier, first we must compute the threshold value. The threshold is a value used to separate the value of face and non-face into building the weak classifier. It is important because it is the base of building the weak classifier.

There are more than one way to compute the threshold value. In this paper, we will compare between two of these methods to see which of them is better to use in face detection system. Based on these two methods, two algorithms will be used to compute the threshold value as follows:

- Taking the average of their mean's using eq.3:

$$Threshold = \frac{\mu_i^+ + \mu_i^-}{2} \quad (3)$$

Where μ_i^+ is the mean of positive samples and μ_i^- is means of the negative samples.

- Finding the optimal threshold that use an algorithm that chose the value that best separates the faces from the non-faces.

2.5 Adaboost and Strong Classifier

The strong classifier is a combination of several weak classifiers. Each of the weak classifiers is given weights depending on its detection accuracy. When classifying a detection window with a strong classifier, all of the weak classifiers are evaluated. The pertained weights of the weak classifiers that classify the window as a face are added together. In the end, the sum of the weights is compared with a predefined threshold to determine if the strong classifier classifies the detection window as a face or not.

There are several types of the Adaboost algorithms used for boosting, Gentle-, Logit-, and Real Adaboost. In [6] they compared the three different boosting algorithms: Discrete Adaboost, Real Adaboost, and Gentle Adaboost. Three 20-stage cascade classifiers were trained with the respective boosting algorithm using the basic feature set.

In [8] they used a cascaded classifier trained by gentle Adaboost algorithm, one of the appearance-based pattern learning method. In [5] they addressed joint Haar-like features using Adaboost, In [7, 9] they addressed a fast and effective multi-view face tracking algorithm based on Adaboost algorithm.

2.6 Cascade classifier

The main idea of building the Cascade Classifier is to reduce computation time, by giving different treatments to different kinds of input, depending on their complexity.

In general works, they use a cascade structure as a detector to detect a face. Cascade detectors have demonstrated impressive detection speeds and high detection rates, using the cascade structure, in order to ensure high testing speed. Where detection rate is the ratio of the true faces to the number of the database .

The cascade training process involves two types of tradeoffs. In most cases, the classifiers with the most features will achieve higher detection rates and lower false positive rates. At the same time, classifiers with more features require more time to compute. In general, one could define optimization framework by the number of classifier stages, the number of features in each stage, and the threshold of each stage, are traded off in order to minimize the expected number of evaluated features.

Each stage in the cascade reduces the false positive rate as well as the detection rate as in eq.4 and eq.5. False positive rate is the probability of falsely rejecting the null hypothesis for a particular test among all the tests performed. (also known as type 1 errors). Where Fp is False positive. An image is called false positive if the image is not a face, but the detector labels it as positive, Tn is True negative which a negative image is correctly labeled as negative. As

$$\text{False positive rate } (\alpha) = Fp/(Fp + Tn) \quad (4)$$

Or

$$\text{False Positive Rate}(A) = 1 - \text{Specificity} \quad (5)$$

Where Specificity = number of Tn/(number of Tn + number of Fp)

3 Face Detection System Architecture

In our proposed architecture, we used a statistical style for measuring the effectiveness of some of the prototypes of the Haar features on the detector, and compares them using two methods that compute the threshold value. This will be done by building a system based on the ideas of The Viola & Jones approach, but different in some ways like changing the number of stages, and changing the number of features in each stage.

Each of the systems is designed to locate multiple faces with a minimum size of 24×24 pixel. The detector will go through a thorough search, at all positions, all scales for faces under all light conditions. The systems will be grouped in pairs ; each pair will have 4 basic features, 4 features, 5 features, 6 features, 7 features, or 8 features, to a total of 12 systems. Each pair's threshold will be computed based on one of two methods.

3.1 Systems description

Before talking about the main parts of the systems, the following will be discussed:

Subwindow size (window size): is the image size that is used in parts of the feature generation and in the units of training and testing. There are different window sizes that are used in other systems like 19×19, 20×20 and 24×24. These sizes affect the number of the features that could be generated for

each image. The 24×24 Subwindow size is used in this paper because it was used in the viola & Jones paper, furthermore it is popular in a lot of other papers that use the Haar like feature generation. Therefore this size will be used in all of our implemented systems.

Fig.3 describes the architecture of the system that will be used as a face detection. It will be based on the Haar features and the Adaboost algorithm in general and each stage will have a detailed description:

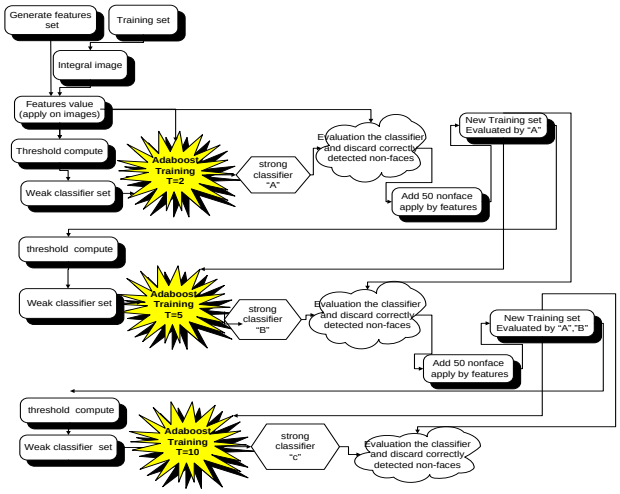


Fig.3 The Cascade Training Process for three stages

3.1.1 Generate Features Set

The features which are generated will be in different sizes and locations as shown in fig.4.

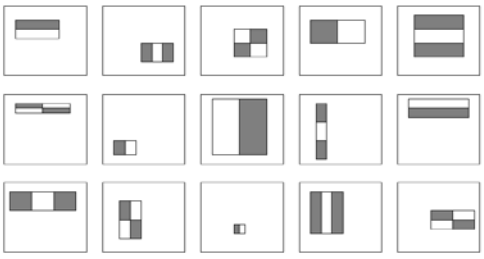


Fig.4 Examples of Haar-like features in different sizes and different locations.

To find the number of the features that could be obtained for any subwindow in any size. The number of features derived from each prototype is quite large and differs from prototype to another . Let X= $\lfloor W/w \rfloor$ and Y= $\lfloor H/h \rfloor$ be the maximum scaling factors in x and y direction where W is the width of window size H is the high of window size and w is the width of feature rectangle h is the high of feature rectangle. An upright feature of size wxh then generates the number of raw features as in eq.6:

$$XY \cdot \left(W + 1 - w\frac{X+1}{2}\right) \cdot \left(H + 1 - h\frac{Y+1}{2}\right) \qquad (6)$$

Eq.7 could be used to calculate this number for every feature, and the researchers could change it as needed. As an example, the number of features generated from the first and third features respectively are 43200, 27600. This equation is used to compute the Haar features.

There are several types of features used in our systems. Fig.5 shows the feature types that will be used in all systems. Table 1 describes the number of features that will be used in each system, which of them use in each one, and how many features will they generate.

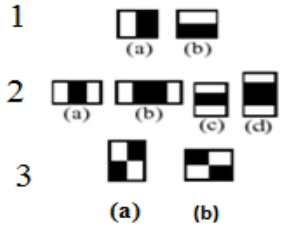


Fig.5 The Haar features used

Table 1. Lists the feature numbers and the type used in each system

System Number	Features number	Feature types used	Total
System 1	4feature	1a, 1b, 2a, 2c	141600
System 2	4feature	1a, 1b, 2a, 3a	134736
System 3	5feature	1a, 1b, 2a, 2c, 3a	162336
System 4	6feature	1a, 1b, 2a, 2c, 3a, 3b	183072
System 5	7feature	1a, 1b, 2a, 2c, 3a, 3b, 2b	202872
System 6	8feature	1a, 1b, 2a, 2c, 3a, 3b, 2b, 2d	222672

3.1.2 Training Set

Viola & Jones approach deals with gray scale images. In this approach every image must be provided in gray scale. Therefore all the images in it that are not in gray scale must be converted. The reason for that is because this approach deals with only gray intensities, so the system needs to preprocess the images in it to build the database.

The first preprocessing will be in the dataset step which consists of the following steps:

Step 1. Determining the two groups

The images that need to be stored in our dataset will be in one of the two groups, which are either:

a. face (positive examples)
b. nonface (negative examples)

The data set consists of two labeled parts. The face dataset consist of numbers of different human face images for different ages, poses, and different luminances and some images with glasses. The rest of the images are taken from The IMM (Informatics and Mathematical Modelling) Face Database which is a Face Database without glasses consist of six different Image types which are:

1. Full frontal face, neutral expression, diffuse light.
2. Full frontal face, "happy" expression, and diffuse light.
3. Face rotated approx. 30 degrees to the person's right, neutral expression, diffuse light.
4. Face rotated approx. 30 degrees to the person's left, neutral expression, diffuse light.
5. Full frontal face, neutral expression, spot light added at the person's left side.
6. Full frontal face, "joker image" (arbitrary expression), diffuse light.

The nonface dataset part can consist of set of different images for anything like trees, flowers, except for human faces. Those images are picked in an arbitrary manner.

Step 2. Preprocessing the face images
After determining the two groups, the first group needs to be processed. The parts in the first group which contain a face image are determined in the images. Once that is done, the faces are cut and saved as a new image in dataset 1 manually.

Step 3. Resizing the images in the data set
After the first step, and determining the two groups of images, the images need to be resized into the subwindows size (24×24).

Step 4. Gray scale
The data set must be in grayscale. All the images will be converted to grayscale if they're not already in grayscale.
To achieve the goal of the preprocessing and obtain dataset 2, we will need to build a function, or a small program, which will consist of two loops that will read the files (images), from the two folders; face and non-face. The images will then be saved in a new folder called dataset 2, this dataset will consist of 250 images for both face and nonface groups. The dataset built contains (250) images, 150 images as face image, and 100 images as nonface images. Fig.6 shows the steps used in building the training datase 2.

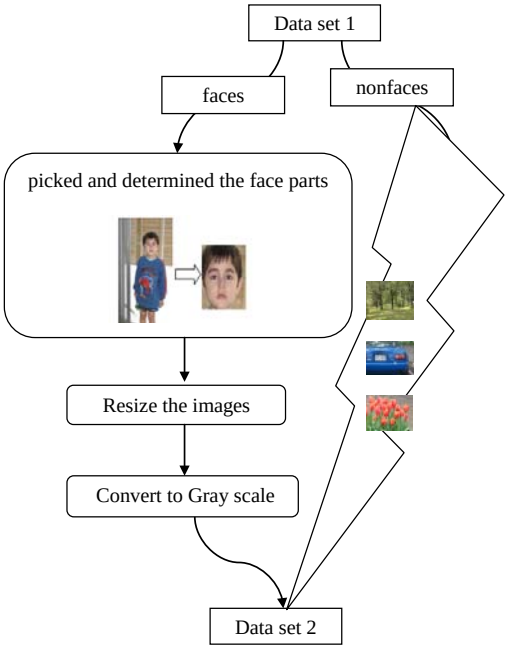


Fig.6 The training set

3.1.3 Integral Image (Fast Feature Evaluation)

The value of the integral image will be computed for all the images using the "Summed Area Table (SAT)" idea and eq.2 . To guarantee that the integral image function does its job correctly, another function will be used which is called the pad function. This function is used to pad two lines of zeros, one at the top of the image, and one to the left of the image, to guarantee a correct result as shown in the fig.7 and fig.8.

To explain the idea of the integral image let's say that there are 2 rectangles 3*3 in fig.7, one represents part of original image and the other represent parts of the integral image.

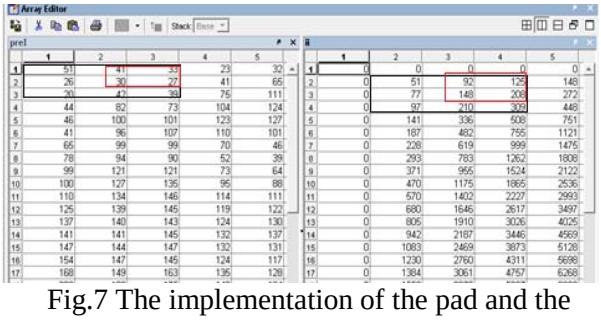


Fig.7 The implementation of the pad and the integral image function as an array

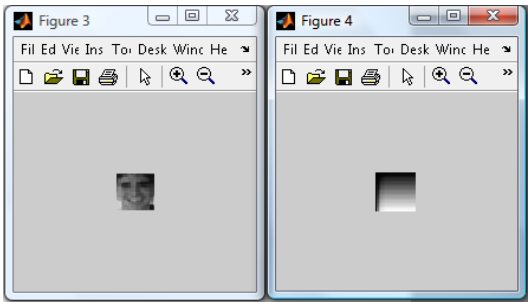


Fig.8 The implementation of the pad and the integral image function in an image

As shown in the second rectangle each pixel will represent the summation value of pixels above and to the left of it. This rectangle is padded, therefore the index starts at p(2,2). If we want to calculate the integral image, we'll need to start at the point +1 (e.g. point x,y, would be start at (x+1,y+1). For example, the integral image for pixel p(2,3) is 208 where the value of summation of all pixels left and up is: (51+41+33+26+30+27), this is represented in p(3,4).

When using this implementation, it is easy to compute the value of the rectangular sum at any scale or position. For example if we want to compute the value of the pixel s of [p(1,2), p(1,3), p(2,2), p(2,3)] we can easily obtain the sum of them by summing the values 208+0-0-77=131 and so on.

3.1.4 Features Extraction

Feature extraction is a special form of dimensionality reduction in pattern recognition and image processing. When the input data to an algorithm is too large to be processed, and is suspected to be notoriously redundant (much data, but not much information), then the input data will be transformed into a reduced representation set of features (also named features' vector). Transforming the input data into the set of features is called feature extraction.

If the features extracted are carefully chosen, it is expected that the features' set will extract the relevant information from the input data in order to perform the desired task, using this reduced representation instead of the full size input. Feature extraction will be used to extract the features of every image in the training set.

This part is used to generate a large number of features very quickly, by computing the integral image for a given set of training images. Then use the feature extraction method to reduce the represented set of features, and afterwards extract a small number of these features by using for the Adaboost algorithm. As the hypothesis of Viola & Jones supposes that a very small number of these features can be combined to form an effective classifier.

3.1.5 Threshold Computing

The threshold is important because it is the base of computing the weak classifier. It is the value that separates the face from the non-face images. The weak classifier is the input to the Adaboost algorithm.

There are several steps for this approach for each feature extraction:

1. Start with the lowest possible threshold.
2. Evaluate the weak classifier with the current threshold on every face example, and store the sum of correctly classified faces in a histogram (Hfaces) at the current threshold.
3. Evaluate the weak classifier with the current threshold on every non-face example, and store the sum of incorrectly classified non-faces in another histogram (Hnonfaces) at the current threshold.
4. Increase the threshold to the next discrete value and start again at step 2 until all thresholds have been evaluated.
5. Compare Hfaces with Hnonfaces and find the threshold t that maximizes the difference function in eq.7.

$$threshold(t)=Hfaces(t)-Hnonfaces(t) \quad (7)$$

Based on this information about threshold, we built 12 systems. Each of the 6 systems are built based on one of the threshold algorithms, but different in the number of the features as shown in table 1. The first 6 systems that are built are based on the average of means and saved as thrshold1.mat for each one of these 6 systems. The other 6 systems that are built based on the algorithm of the optimal threshold are saved as threshold2.mat for each one of them.

3.1.6 Weak Classifier Set (retrain)

The Adaboost learning algorithm needs to build simple classifiers by using the Haar like features. Each single feature will be associated with a threshold value to build a weak classifier that is used as a simple classifier which is an input to the Adaboost algorithm. A practical method for completing the analogy between weak classifiers and features can be explained as follows:

1. Restrict the weak learner to the set of classification functions, each of which depend on a single feature. The weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples.
2. For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples is misclassified.
3. A weak classifier (h_i) thus consists of Feature (f_i), Threshold (θ_i) Parity (p_i), indicating the direction of the inequality sign.

An easy way to link the weak learner and Haar features is to assign one weak classifier to one feature. The value of a given single feature vector fi is evaluated at x, and the output of the weak classifier hi(x) is either -1 or 1. The output depends on whether the feature value is less than a given threshold θi in eq.8.

$$h_i(x) = \begin{cases} 1 & if\ p_i f_i(x) < p_i \theta_i \\ -1 & otherwise \end{cases} \quad (8)$$

Where p_i is the parity and x is the image-box to be classified. Thus our set of features defines a set of weak classifiers. From the evaluation of each feature type on training data, it is possible to estimate the value of each classifier's threshold and its parity variable.

The weak classifier that is generated will be an array of two dimensions (features, database) of 1,-1. To retrain the weak classifier, use the new value of the threshold to compute the weak classifier.

3.2 Adaboost training

The task of the Adaboost algorithm is to pick a few hundred features and assign weights to each feature. A set of training images is reduced to compute the weighted sum of the chosen rectangle-features and apply a threshold. The algorithm builds a strong classifier from the weak classifier by choosing the lowest error in the weak classifier groups.

3.2.1 The Training Algorithm

The following explains how training algorithm works:

- give example images(x_1 ,y_1)……(x_{250}, y_{250}),where y_i=1,-1 for negative and positive respectively, where X is the images in 24 ×24 size and it consist of the 150 images are face and the rest 100 images are non-face, where the Y is label of 1,-1 for face , non-face.

- Initialize weights $w_{1,i}$ = , for $y_{i=1,-1}$ respectively, where m =100_and =150 are the numbers of negatives (non-face) and positives (face) respectively.
- For t=1……..T, in this paper T=3 in each system

1. Normalize the weights, by using eq.9.

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \quad (9)$$

So that w_t is a probability distribution.

2. For each feature j, train a classifier h_j which is restricted to using a single feature. (weak classifier compute) The error ϵ is evaluated with respect to w_t computed by eq.10.

$$\epsilon_j = \sum_i w_i |h_j(x_i) - y_i| \quad (10)$$

3. Choose the classifier h_t, with the lowest error ϵ_t
4. Update the weights, using eq.11.

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \quad (11)$$

Where e_i= 0 if example x_i is classified correctly, e_i =1 otherwise, and β_t $\frac{\epsilon_t}{1-\epsilon_t}$

=

- The final strong classifier can be calculated using eq.12.

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases} \quad (12)$$

where α_t =log

To explain how the Adaboost algorithm works, fig.9 describes the process of the Adaboost training.

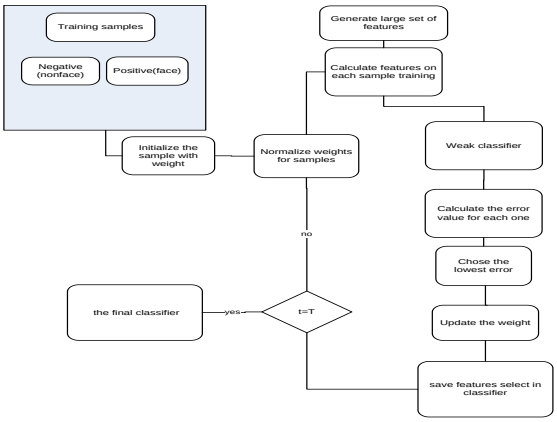


Fig.9 The Adaboost training flowchart

The Adaboost algorithm will have two inputs, the sample weight and the values that are generated from applying the Haar features on the images. For each alteration, the Adaboost computes the threshold, weak classifier, and calculates the error value for each classifier. After that the algorithm chooses the classifier with lowest error, updates the weights, and then normalizes the weights after each update. The feature that was chosen in the classifier is saved, and then the round is iterated. Finally, the final classifier contains all the features that were saved. At the end of the algorithm, there will be a single strong classifier. The accuracy of this classifier depends on the training samples and the weak classifier. After several strong classifiers are trained, they are combined together to build the detector.

3.2.2 The Strong Classifier

After using the Adaboost algorithm to reduce the number of features, by selecting the best features and building a strong classifier from combining the weak classifiers, and according to Viola et.al, the detection performance of a single classifier with a limited number of features is very poor for a face detection system. They suggest the concept of a cascade, instead of evaluating one large strong classifier on a detection window. It is simply a sequence of strong classifiers which all have a high correct detection rate.

The key idea is using a multi-stage classifier as shown in Fig.10. The system needs another algorithm to help reduce significantly high computation time for the face detection system, and achieves better detection performance. It is an efficient algorithm, because it depends on the principle of rejecting the negative subwindow quickly in the earlier stages of the cascade, which uses a small number of features to increase the computation process. If the subwidow is positive, it will pass it to the next stage which is more complex from the previous stage, and so on, until it reaches the last stage. The last stage is more complex, and has a large number of features compared the other stages. The cascade structure uses a degenerate decision tree. In the cascade classifier, the subwindow which is used to input the classifier has two probabilities. The first probability is to reject in one of the stages, which is classified as a negative sample (nonface), or pass all the stages, then it will be classified as a (positive) face.

The training cascade structures the number features on each stage, and the number of stages depend on the two constraints, which are the face detection rate and the false positive rate.

The cascade structure has three main parameters that need to be determined: The total number of classifiers, the number of features in each stage, and the threshold of each stage.

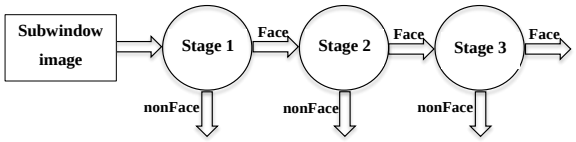


Fig.10 The Cascade Classifier structure.

3.3 Evaluating the Classifier and Discarding Correctly Detected Non-Faces

To obtain better results for the next strong classifier, evaluate the training images for all of the non-face images, and correctly discard all of the non-face images. The training set is then decreased, and used to build a **New Training Set Evaluated By New Classifier.** Since the data training is not too big, it might cause a problem in the training with the Adaboost, and cause errors in computing. Therefore we need to add data training in the step with 50 images in each Adaboost train. About 100 images will be added to the original image test to enhance the training part.

3.3.1 The Detector and the Detection

The detector is considered as a second part for this system in other papers. It is used after applying the previous steps. The detector's structure is based on the Adaboost algorithm. For each strong classifier that the Adaboost generates, the threshold of the

stage will be computed and saved to be compared later, this threshold is different than the threshold used in the training stage. It will be decided if the input image is a face or not as shown in fig.11.

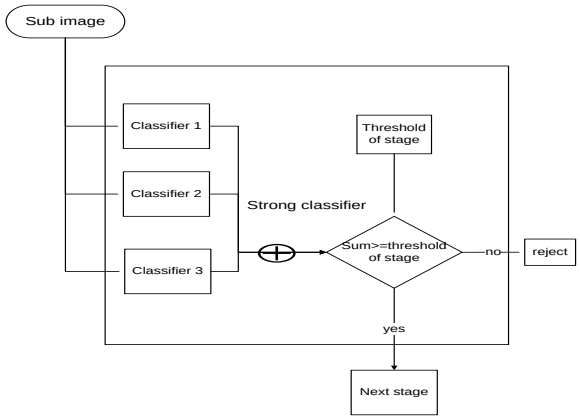


Fig.11 The detection procedure

Another thing that must be known in this principle is the size of the image that was scanned and the detector scale. The detector that was generated from the training has a specific 24×24 size. To scan images bigger than this size, the detector will have to scan the entire image to find whether a face exists or not, also there might be other things that interfere with the detector to find whether a face exists or not. Therefore two solutions to solve this issue could be used. First, resize the detector make the detector bigger (feature values), or resize the image to make the image smaller. Each time the detector will scan the image to find whether a face exists or not. To make the image that the system can scan, a function is needed to convert the image if it is bigger than 384 ×288 to an image in this size.

In the proposed system, the idea of resizing the image was used, which is shown in fig.12. There are 12 layers for any image size 384 ×288 to pass, and in each one the detector will scan all the images trying to find any face. First layer, the original images is divided into subwindows, each window size will be 24×24. The subwindows will be inputted in the detector to decide if the sub image is a face or nonface. If the image was nonface, it would be discarded, otherwise the image is saved. This operation is repeated on the 11 sizes of the image until the image size becomes 24×24 or smaller, and then plot the result on the original image.

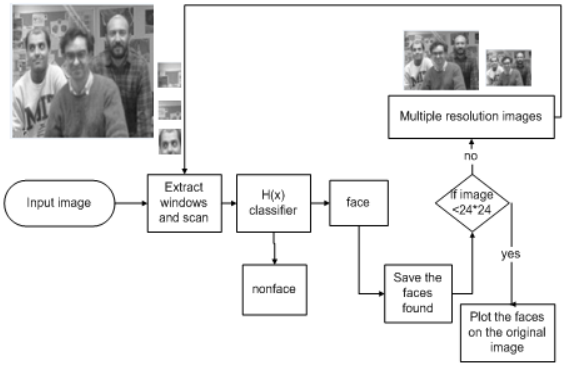


Fig.12 The used block diagram of the face detector

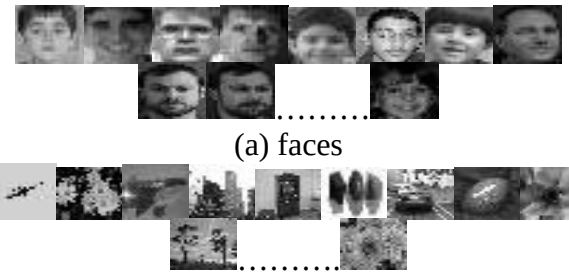
4 Experimental Results

The face detection systems presented in this paper was trained and tested using MATLAB 7.0 on Intel core (TM) i3 2.13 GHz 4GB of RAM and windows Vista ^{TM} Ultimate operating system.

The 12 systems were built similarly, but they differ in what features they have and the method used to compute the threshold. The systems will be grouped into six groups based on the number of features, and two groups based on the threshold's calculation method.

5.1 Training

The database used in training is built by hand for the purpose of obtaining a database that has everything in terms of face details like glasses, scarf on the head, beards, Mustaches, face color, Illumination, and anything that may help build a strong database. Even though it may have different type of images, the number of the images in this database not too big like in other databases. In this database 250 images were used. An examples of the images that were used in the training database is shown in fig.13. All of the images were scaled to the size of the subwindow, which is used in the systems (24×24).



(a) faces

(b) nonfaces

Fig.13 Examples of images that were used in the training database

All of the training data was labeled as face and non-face images manually. The dataset has 2 inputs to the system for training, the first input is the image and the second input is a label for image groups as 1 for face and -1 for nonfaces.

In parallel with processing the dataset images there are feature generation process. In this process, the features will be generated for each type, in every scale and location, and for each system. The number of features in each system is explained in table 1. After that the training process will apply every feature of these features on the training images, and extract features to prepare them to compute the threshold, and generate the weak classifier and then continue the other process. The time needed to apply the features on the images, increases as the number of features increase.

After computing the threshold values, using the two methods, the weak classifiers for each system before and after the training using the Adaboost will give the rates for every system as shown in the tables 2 and 3 where FP is **False positive.** An image is called false positive if the image is not a face, but the detector labels it as positive, TP is a **True positive** image where an image of a face that the detector correctly labeled positive. **False negative** FN is a face image, but the detector labels it as negative, which means it does not find that face. TN is **True negative** which a negative image is correctly labeled as negative.

Table 2. Systems results before and after training based on Threshold 1 (Average of Means)

Fp	Tn	Fn	Tp	Training	Rate 1
0.436	0.564	0.419	0.582	before	System 1
0	1	0.273	0.727	after	
0.452	0.548	0.442	0.558	before	System 2
0	1	0.28	0.72	after	
0.446	0.554	0.429	0.571	before	System 3
0	1	0.28	0.72	after	
0.641	0.359	0.432	0.568	before	System 4
0.095	0.905	0.053	0.947	after	
0.451	0.549	0.438	0.562	before	System 5
0	1	0.347	0.653	after	
0.450	0.550	0.437	0.563	before	System 6
0	1	0.293	0.707	after	

Table 3. Systems results before and after training based on Threshold 2 (Optimal Threshold)

Fp	Tn	Fn	Tp	Training	Rate 1
0.672	0.327	0.109	0.890	before	System 1
0	1	0.187	0.813	after	
0.716	0.283	0.093	0.907	before	System 2
0	1	0.22	0.78	after	
0.698	0.302	0.101	0.897	before	System 3
0	1	0.22	0.78	after	
0.831	0.169	0.136	0.864	before	System 4
0	1	0.013	0.987	after	
0.718	0.282	0.095	0.905	before	System 5
0	1	0.287	0.713	after	
0.716	0.285	0.096	0.904	before	System 6
0	1	0.213	0.787	after	

5.3 Testing

In the testing, all systems have been tested on the CMU+MIT database and compare between them. One image from this database is shown in Fig.14. The image consists of 25 faces. The result of each detection is shown on it.

Table 4 will display the detection window, false positive and true negative. The left side of the table represents the result of the group that is based on the average of mean, to compute the threshold, and the right side contains the results of the other method.

Fig.14 Sample of testing classifiers

Table 4. The result of detector on the test image

FP	TP	Detected	Systems	FP	TP	Detected	Systems
107	8	115	4a-2	12	5	17	4a-1
54	8	62	4b-2	11	4	15	4b-1
67	8	75	5-2	13	5	18	5-1
5	1	6	6-2	52	0	52	6-1
13	3	16	7-2	11	7	18	7-1
42	5	47	8-2	7	6	13	8-1

As a result of these systems, the first group generates faster, but is less accurate than the second group. Furthermore , the second group generates more rectangles than the first, so the FP images in the second group is higher than the FP in the first systems.

5 Discussion

The detector that was used in this paper was a simple detector; it was used to gather statistical data when comparing between features and the two

methods to compute the threshold. As seen in the tables 2 and 3, the results confirm that the Adaboost algorithm increases the efficiency of the system for all of the systems. As for the feature types, the results have shown that when using eq.2 to calculate the threshold, the results were better if Viola & Jones system, that included, feature 5 was used. On the other hand, if eq.3 was used, the 4 feature (not basic), 8 features system is better. Also it was noted that the systems that used the diagonal features showed that the detection rate was decreased, except when it was used for the Viola & Jones system. This shows why researchers don't use this feature in their research, and why it was not used by Intel in open CV.

The number of features is directly proportional to the detector's accuracy, it was mentioned in different papers that the number of features in a single classifier should be at least 15 to give good results. But in the 3 stages in this paper is the highest number was 10.
According to the table 4, the threshold values were better for systems that used eq.3 in the training stages. On the other hand during the testing stages that were done on the CMU+MIT databases, and according to table 4, the data show that the results varied between the systems, but in general it slightly better for systems used eq.3. The difference between the two is the time required to calculate the threshold. The time required to calculate a single value using eq.3 is approximately 0.32ms, while the time required for eq.2 is approximately 0.02ms.

The number of the databases greatly affects the detector's efficiency, and since the data is not big compared to the other systems, it will affect the threshold's calculation. That's why adding 50 non-face images to the data is used after discarding some images.

Since the detector has low accuracy, the number of faces detected will be low. Another problem that this might cause is the high number of false positives (FP), furthermore we haven't mentioned the overlap that might happen which caused an increased number of FP.

6 Conclusion

In this paper 12 systems were implemented based on the Haar features and Adaboost algorithm. There are 8 different types of Haar features, on the other hand there are two methods that were used to compare the threshold. Based on the above configurations, the systems were divided into 6 groups based on the features, and 2 groups based on the threshold calculation methods. These features were divided to create 6 groups based on them; each group containing 4 simple, 4, 5, 6, 7, or 8 of these Haar features. On the other hand, the 2 groups based on the threshold were based on the average of means and the optimal threshold methods.

Based on these results, we could solve the problem of the overlap to get better detection rates, do more comparisons with more threshold methods (single or multi threshold), add multi-face detection and building a face detection system that's more accurate and faster.

As a recommendation for the designed systems to be more efficient and achieve higher detection rate, we could enhance the stages by adding more features or by adding more stages (strong classifier).

Acknowledgment

The authors would like to acknowledge financial support for this work from the Deanship of Scientific Research (DSR), University of Tabuk, Tabuk, Saudi Arabia, under grant no. 0188/1436/S

References:

[1] Viola P. & Jones M., "Robust Real-Time Object Detection", in Second International Workshop on Statistical Learning and Computational Theories of Vision Modeling, Learning, Computing and Sampling, July 2001.
[2] Viola P. and Jones M., "Rapid Object Detection Using A Boosted Cascade Of Simple Features." In ***IEEE CVPR*** 2001, 2001.
[3] Freund, Y. , Iyer, R., Schapire, R. and Singer, Y. An efficient boosting algorithm for combining preferences. In *Machine Learning: Proceedings of the Fifteenth InternationalConference*, 1998.
[4] Zhou Z.-H. and Yu. Y." Adaboost". In X. Wu and V. Kumar, editors, The Top Ten Algorithms in Data Mining. Chapman & Hall, Boca Raton, FL, 2009.
[5] Mita, T., Kaneko, T. and Hori, O.: Joint Haar-like features for face detection. In Proc of ICCV ,2005.
[6] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," IEEE ICIP, vol. 1, pp. 900-903, Sep. 2002.

[7] J. Meynet and V. Popovici and JP. Thiran, “Mixtures of Boosted Classifiers for Frontal Face Detection,” Signal Image and Video Processing, 2007.
[8] Kim J.B. , Kee S.C. And Kim J.Y. " Fast Detection Of Multi-View Face And Eye Based On Cascaded Classifier " Mva2005 Iapr Conference On Machine Vision Applications, May 16-18, 2005 Tsukuba Science City, Japan.
[9] Pham T. V., Smeulders A. W. M. and Ruis S. " Adaboost Learning Of Shape And Color Features For Object Recognition, " In ICML 2005 Workshop on Machine Learning Techniques for Processing Multimedia Content, Bonn, Germany.
[10] S Zhang, X Zhao, and B Lei, Facial Expression Recognition Based on Local Binary Patterns and Local Fisher Discriminant Analysis,WSEAS TRANSACTIONS on Signal Processing,Vol. 8, No. 1, 2012, pp. 21-31.
[11] S Zhang, X Zhao, and B Lei, Facial Expression Recognition Using Sparse Representation. Wseas Transactions On Systems, Vol.11, No.8, 2012, pp.: 440-452.
[12] Hazem M. El-Bakry, Nikos Mastorakis, Fast Image Matching on Web Pages, WSEAS Transactions on Signal Processing, Vol.4, No. 5, 2009, pp.157-166.