Application of Haar-like Features in Three AdaBoost Algorithms for Face Detection

Dhyaa Shaheed Sabr Al-Azzawy
Ph. D. Computer Sciences. - University of Wasit – College of Sciences

تطبيق خصائص هار-لايك في ثلاثة خوارزميات لتحديد الوجه

ضياء شهيد صبر - جامعة واسط / كلية العلوم

الخلاصة:

نقدم في هذا البحث نظام التصنيف التدفقي والمؤلف من خمس مراحل تصنيفات قوية، وبصورة عملية تم تحليل ثلاثة أنواع من الخوارزميات (Real و Modest و Gentel)، التي تم استخدامها لتقوية المصنف القوي. ولكل مرحلة من مراحل المصنف القوي تم استخدام خمسة نماذج من صفات (Haar-like)، هي (مربعين أثنين افقيين، مربعين اثنين عموديين، ثلاثة مربعات افقية، ثلاثة مربعات عمودية، و اربعة مربعات). وتم تطبيق صفات المربعين الافقيين والعمودين وخوارزمية ادابوست في المصنفات القوية الاولى والثانية على التوالي، و تطبيق صفات المربعات الثلاثة الافقية والعمودية وخوارزمية ريل ادابوست في المصنفات القوية الثالثة والرابعة على التوالي، واخيرا تم تطبيق صفات المربعات مع خوارزمية ريل ادابوست في المصنف القوي الخامس من المصنف التدفقية. ومن خلال التنفيذ لوحظ بأن افضل استخدام لخوارزمية ادابوست هي: خوارزمية موديست ادابوست لكل من المصنفات القوية الاولى والثانية، و خوارزمية ريل ادابوست للمصنفات القوية الثالثة والرابعة.

Abstract

In this paper we introduce a proposed cascade classifier system consists of five strong classifiers (stages), and empirically analysis three types of Adaboost algorithms (Real, Modest, and Gentle) which it were used for boosting the strong classifiers. Five prototypes of haar-like features (two-horizontal rectangles, two-vertical rectangles, three-horizontal rectangles, three-vertical rectangles, and four-rectangles) are used for each strong classifier (stage) of cascade classifier. Two-horizontal, two-vertical rectangles features and Modest-Adaboost Algorithm are applied for the 1st and 2nd strong classifier respectively, three-horizontal, vertical-rectangles features and Real-Adaboost Algorithm are applied for 3rd and 4th strong classifier respectively, and the last, four-rectangles and Real-Adaboost Algorithm are applied for the 5th strong classifier of cascade classifier. The implementation shows that the best use of Adaboost algorithm is: the Modest-Adaboost algorithm for both 1st and 2nd strong classifier, and the Real-Adaboost algorithm for 3rd,4th and 5th strong classifier.

Key words: Face Detection, Haar-Like Features, and Adaboost Algorithm.

1- Introduction

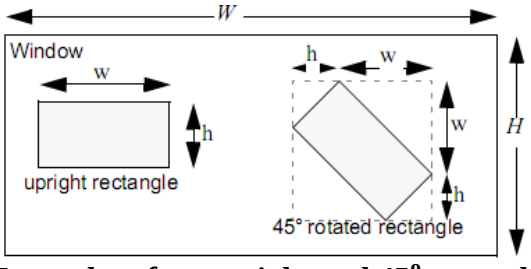
Recently Viola et al. have proposed a multi-stage object classification procedure that reduces the processing time substantially while achieving almost the same accuracy as compared to a much slower and more complex single stage classifier [1]. This paper extends their rapid object detection framework by constructing cascade of strong classifiers consists of five stages, each stage use one type of haar-like features, applying three Adaboost Algorithms (Real-Adaboost, Modest-Adaboost, and Gentle-Adaboost algorithm), analysis the results that occurs with each algorithm independently and analysis the effect of training rounds of the three Adaboost Algorithms.

Sections 2 and 3 explain the Haar-like features, section 4,5, and 6 discusses weak learner, strong classifier and cascade of strong classifiers respectively, section 7 shows the analysis and results of the implementation for the cascade of strong classifiers.

2- Features

The detection process of proposed system is based on the value of simple features. Any detection system depend on either features-based or pixel based in detection process, but most the systems using features rather than the pixel. The most common reason is that features can act to encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. The other reason is our features can be computed at any position and scale in the constant time and high speed. The speed of feature evaluation is also a very important aspect since almost all object detection algorithm slide a fixed-size window at all scales over the input image.

According to Kasinski et el [2], and Lienhart et el [3], the basic unit for testing the presence of an object is a window of WxH pixels. The base resolution of our proposed system is 24x24 pixels and the exhaustive set of rectangle features is quite large, over 180,000, but we will show that we used a few number of set in the result section.



Figure(1) Examples of an upright and 45° rotated rectangle

Viola et. el. [1], and Lienhart et. el. [4] assumed that the rectangle feature is specified by the tuple r=(x, y, w, h, α) with $0 \le x, x+w \le W$, $0 \le y, y+h \le H$, $w, h > 0, \alpha \in \{0^\circ, 45^\circ\}$, where, x, y left most corner of window, w, h width and height of rectangle. Rectangle's pixel sum is denoted by RecSum(r). Two

examples of such rectangles are given in (Fig. 1). The raw feature set is then the set of all possible features of the form.

$$feature_I = \sum_{i \in I = \{1,\ldots,N\}} \omega_i \operatorname{Re} cSum(r_i), \quad (1)$$

where the weights $\omega_i \in \Re$, the rectangles r_i, and N are arbitrarily chosen. They described the prototype of rectangle features according to the following rules [2] [3]:

1- Only weighted combination of pixel sums of two rectangles are considered.
2- The weights have opposite signs, and are used to compensate for the difference in area size between the two rectangles.
3- The features mimic haar-like features and early features of the human visual pathway such as center-surround and directional responses.

These restrictions lead us to the 14 features prototypes shown in (Fig. 2).

- Four edge features,
- Eight line features, and
- Two center-surround features.
- One special diagonal line feature.

These prototypes are scaled independently in vertical and horizontal direction in order to generate a rich, over-complete set of features. Note that the line features can be calculated by two rectangles only. Hereto it is assumed that the first rectangle r_0 encompasses the black and white rectangle and second rectangle r_1 represents the black area. In our experiments we used (1a), (1b), (2a), (2c), and (4).

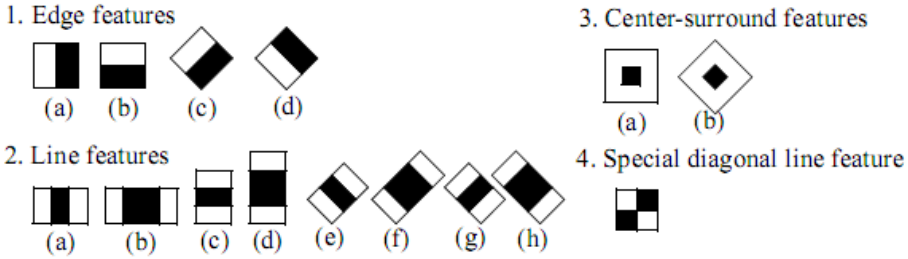


Figure (2) Feature prototypes of simple haar-like and center surround features. Black areas have negative and white areas positive weights.

The number of features derived from each prototype can be calculated as follows. Let X=[W/w] and Y=[H/h] be the maximum scaling factors in x and y direction. An upright feature of size wxh then generates [4]:

$$XY \bullet (W+1-w\frac{X+1}{2}) \bullet (H+1-h\frac{Y+1}{2}) \quad (2)$$

And features for an image window of size WxH, while 45^o rotated feature generates:

$$XY \bullet (W+1-z\frac{X+1}{2}) \bullet (H+1-z\frac{Y+1}{2}) \text{ , with z=w+h.} \quad (3)$$

3- Integral Image

Rectangle image can be computed very rapidly using an intermediate representation for the image which we call the integral image [1]. The integral image at location x,y contains the sum of the pixels above and to the left of x,y, inclusive (Fig. 3a):

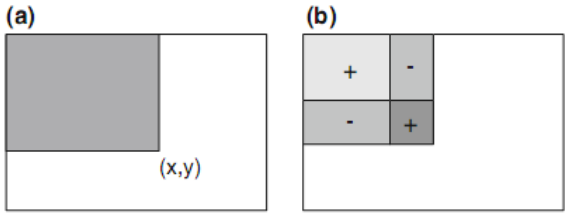
$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y'), \qquad (4)$$

Where, ii(x,y) is the integral image and i(x,y) is the original image. In [2] used the notation of SAT(x,y) to represent ii(x,y), the whole table can be computed in a single pass using the following formula [2] [4] [5] :

$$SAT(x,y)= SAT(x,y-1)+SAT(x-1,y)+i(x,y)-SAT(x-1,y-1) \qquad (5)$$

With SAT(-1,y)=SAT(x,-1)=SAT(-1,-1)=0. Once filled, the SAT enables computation of RecSum(r) for any upright rectangle r=(x,y,w,h,0^o) with only four look-ups (Fig. 3b).

RecSum(r)=SAT(x-1,y-1)+SAT(x+w-1,y+h-1)-SAT(x+w-1,y-1)-SAT(x-1,y+h-1) (6)



Figure(3) Auxiliary image representation: (a) the idea of SAT, (b) fast feature calculation using SAT.

4- Weak classifier

For each detection window WxH of the image being processed, a **weak classifier** gives a decision $\delta_w(Window) \in \{-1,+1\}$ indicating membership of the window to one of two classes, labeled by -1 (negative, e.g. a non-face) and +1 (positive, e.g. a face).

Recalling the formula number (1) for computing the feature of window, Viola et. el. [5] used the following formula of weak classifier to classify the window :

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

Her x is a window WxH pixel sub-window of an image, *f* is the feature, *p* is the direction of inequality, and θ is the threshold.

In the proposed system the size of window are 24x24 pixels, and the number of features prototypes are 5 types (Fig. 2 (1a) (1b) (2a) (2c) (4)). And the weak classifier was CART (Classification and Regression Tree).

CART is classification method which uses historical data to construct decision trees. Depending on available information about the dataset, classification tree or regression tree can be constructed. Constructed tree can be then used for classification of new observations. Classification trees are used when for each observation of learning sample we know the class in advance. Classes in learning sample may be provided by user or calculated in accordance with some exogenous rule [6].

Decision tree is a tree graph, with leaves representing the classification result and nodes representing some predicate. Branches of the tree are marked true or false. Classification process in case of decision tree is a process of tree traverse. We start from root and descend further, until we reach the leaf, the value associated with the leaf is the class of the presented sample. At each step we compute the value of the predicate associated with current node. We choose next node (or leaf) that is connected with current by the branch with the value of current nodes predicate.

The tree consists of a number of nodes (splits), if number of splits is one it means this tree is a **Stump**, and if the number of splits more than one, the tree is a **CART**.

5- Strong Classifier

A strong classifier is composed of a number of weak classifiers. Its decision is made by weighted voting: decision of a t^{th} weak classifier is multiplied by alpha α_t [5]:

$$c(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

Strong classifier used Adaboost algorithm to find a weak hypothesis h_t, this algorithm takes as input a training set $(x_1,y_1), \ldots, (x_n,y_n)$ where each x_i belong to some domain or instance space X, and each label y_i is in some label set Y, Y={-1,+1}. Adaboost calls a given weak or base learning algorithm repeatedly in a series of rounds m=1, … , M. One of the main ideas of the algorithm is to maintain a distribution or set of weights over the training set. The weight of this distribution on training example i on round m is denoted $w_m(i)$. Thus, the main job of weak learner is to find the weak hypothesis h_m::X→ {-1,+1}. [7]

In the proposed system, we experimentally used three types of Adaboost algorithm (Real-Adaboost Algorithm [1], Modest-Adaboost Algorithm [8], and Gentle Adaboost algorithm [2]). See (Fig. 4), (Fig. 5), and (Fig. 6). (Fig. 7) illustrates algorithm of the implementation of the all three Adaboost algorithms.

6- Cascade of Classifiers

A cascade of classifiers is degenerated decision tree where at each stage a strong classifier is trained to detect almost all objects of interest (frontal faces in our example) while rejecting a certain fraction of the non-object patterns, see (Fig. 8). [3] [9].

For instance, in our case, five stages were trained using five prototypes of haar-like features; each stage was trained with one feature only and using either Real, Modest or Gentle Adaboost learning algorithm. We will analysis the proposed system for each algorithm with details in the results section and the training time depends on the number of tree nodes, the time for 200 nodes require more than one hour for training the strong classifier.

Adaboost algorithm is a powerful machine learning algorithm it can learn strong classifier based on a (large) set of weak classifiers by re-weighting the training samples and adjusting the threshold to minimize false negatives. Our sets of weak classifiers are all classifiers which use one feature from our feature pool in combination with a simple binary thresholding decision. At each round "Iteration" of boosting, the feature-based classifier is added that best classifiers the weighted training samples. With increasing stage number the number of weak classifiers, which are needed to achieve the desired false alarm rate at the given hit rate, increases. [3] [1] [7]

0. (Input)

(1) Training examples {(x_1,y_1), … , (x_N,y_N)}, where N=a+b; of which a examples have y_i=+1 and b examples have y_i=-1;

(2) The maximum number *Mmax* of weak classifier to be combined;

1. (Initialization)

$$w_i^{(0)} = \frac{1}{2a} \quad \text{for those examples with } y_i = +1 \text{ or}$$

$$w_i^{(0)} = \frac{1}{2a} \quad \text{for those examples with } y_i = -1$$

***M*=0;**

2. (Forward Inclusion)

(1) while M < *Mmax*

(2) *M* ← *M* + 1;

(3) Choose h_M according with respect to the weighted error:

$$\varepsilon_i^M = \min_{f,p,\theta} \sum_i w_i \mid h(x_i, f, p, \theta) - y_i)$$

(4) Define $h_M(x) = h(x, f_M, p_M, \theta)$ where $f_M, p_M, and \theta_M$ **are the minimizers of** ε_M **.**

(5) Update $w_i^{(M)} \leftarrow w_i^M * \beta_M^{1-e_i}$**, where, *ei*=0 if example xi is classified correctly, *ei*=1 otherwise, and** $\beta_M = \frac{\varepsilon_M}{1-\varepsilon_M}$**.**

3. (Output) $C(x) = \begin{cases} 1 & \sum_{M=1}^{M_{\max}} \alpha_M h_M(x) \geq \frac{1}{2}\alpha_M \\ 0 & \text{otherwise} \end{cases}$ **Where,** $\alpha_M = \log \frac{1}{\beta_t}$

Figure (4) Real-Adaboost Algorithm

0. (Input)

(5) Training examples {(x_1,y_1), … , (x_N,y_N)}, where N=a+b; of which a examples have y_i=+1 and b examples have y_i=-1;

(6) The maximum number *Mmax* of weak classifier to be combined;

1. (Initialization) $w_i^{(0)} = \frac{1}{2a}$ for those examples with $y_i = +1$ or

$$w_i^{(0)} = \frac{1}{2a} \quad \text{for those examples with } y_i = -1$$

***M*=0;**

2. (Forward Inclusion)

while M < M_{max}

(1) *M* ← *M* + 1;

(2) Choose h_M according with respect to the weighted error:

$$\varepsilon_i^M = \min_{f,p,\theta} \sum_i w_i \mid h(x_i, f, p, \theta) - y_i)$$

(3) Define $h_M(x) = h(x, f_M, p_M, \theta)$ where $f_M, p_M, and \theta_M$ **are the minimizers of** ε_M **.**

(4) $s_1 = \sum_i for \text{ all } y_i = +1, h_m * w_i$ **and s2=** $\sum_i for \text{ all } y_i = -1, h_m * w_i$

(5) Update $w_i^{(M)} \leftarrow w_i^M * \beta_M^{1-e_i}$**, where, e_i=0 if example xi is classified correctly, e_i=1 otherwise, and** $\beta_M = \frac{s_1 - s_2}{s_1 + s_2}$**.**

4. (Output)

$C(x) = \begin{cases} 1 & \sum_{M=1}^{M_{\max}} \alpha_M h_M(x) \geq \frac{1}{2}\alpha_M \\ 0 & \text{otherwise} \end{cases}$ **Where,** $\alpha_M = \log \frac{1}{\beta_t}$

Figure (5) Modest-Adaboost Algorithm

0. (Input)

(3) Training examples $\{(x_1,y_1), \ldots, (x_N,y_N)\}$, where N=a+b; of which a examples have y_i=+1 and b examples have y_i=-1;

(4) The maximum number *Mmax* of weak classifier to be combined;

1. (Initialization) $w_i^{(o)} = \frac{1}{2a}$ for those examples with $y_i = +1$ or

$w_i^{(o)} = \frac{1}{2a}$ for those examples with $y_i = -1$ **M=0;**

2. (Forward Inclusion)

while M < M_{max}

(1) $M \leftarrow M + 1$;

(2) Choose h_M according with respect to the weighted error:

$$\varepsilon_i^M = \min_{f,p,\theta} \sum_i w_i \mid h(x_i, f, p, \theta) - y_i)$$

(3) Define $h_M(x) = h(x, f_M, p_M, \theta)$ where $f_M, p_M, and \theta_M$ **are the minimizers of** ε_M **.**

(4) $s_1 = \sum_i for$ all $y_i = +1, h_m * \frac{1}{w_i}$ **and s2=** $\sum_i for$ all $y_i = -1, h_m * \frac{1}{w_i}$

(5) $s3 = \sum_i for$ all $y_i = +1, h_m * \frac{1}{w_i}$ **and s4=** $\sum_i for$ all $y_i = -1, h_m * \frac{1}{w_i}$

(6) Update $w_i^{(M)} \leftarrow w_i^M * \beta_M^{1-e_i}$**, where, e_i=0 if example xi is classified correctly, e_i=1 otherwise, and** $\beta_M = s_1 * (1 - s_3) - s_2 * (1 - s_4)$**.**

4. (Output)

$$C(x) = \begin{cases} 1 & \sum_{M=1}^{M_{max}} \alpha_M h_M(x) \geq \frac{1}{2}\alpha_M \\ 0 & \text{otherwise} \end{cases}$$ **Where,** $\alpha_M = \log \frac{1}{\beta_t}$

Figure (6) Gentle-Adaboost Algorithm

0. (Input)

(1) Training examples $\{(x_1,y_1), \ldots, (x_N,y_N)\}$, where N=a+b; of which a examples have y_i=+1 and b examples have y_i=-1;

(2) The maximum number *Mmax* of weak classifier to be combined;

1. (Initialization) $w_i^{(o)} = \frac{1}{2a}$ for those examples with $y_i = +1$ or

$w_i^{(o)} = \frac{1}{2a}$ for those examples with $y_i = -1$ **M=0;**

2. (Forward Inclusion)

Choose one of the following:

(1) Real-Adaboost.

(2) Modest-Adaboost.

(3) Gentle-Adaboost.

3. (Output)

$$C(x) = \begin{cases} 1 & \sum_{M=1}^{M_{max}} \alpha_M h_M(x) \geq \frac{1}{2}\alpha_M \\ 0 & \text{otherwise} \end{cases}$$ **Where,** $\alpha_M = \log \frac{1}{\beta_t}$

Figure (7) Algorithm of all Adaboost Algorithms

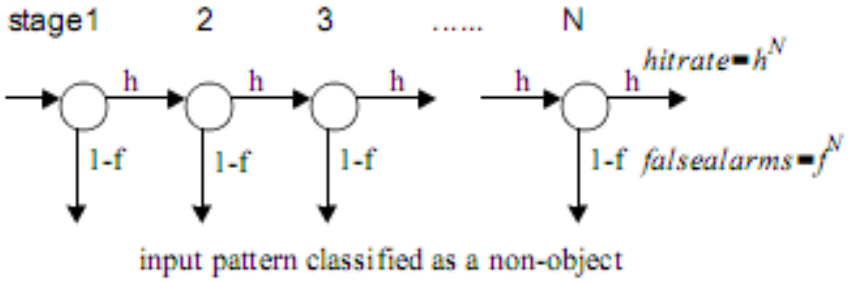


Figure (8) Cascade of Classifiers with *N* stages. At each stage a classifier is trained to achieve a hit rate off *h* and a false alarm rate of *f*.

The proposed cascade system of increasingly specialized stages, each one being trained to reject the false positives of previous stages, while detecting all positive instances. In other words, a positive result from the first strong classifier triggers the evaluation of a second strong classifier which has also been adjusted to achieve very high detection rates, a positive result from the second strong classifier triggers a third strong classifier, and so on, a negative outcome at any point leads to the immediate rejection of sub-window. [1]

A summary of the architecture of the proposed cascade classifier is illustrated in (Fig. 9), where the numbers of stages are five stages. When an instance, WxH sub-window of image, enters the detector, it is examined by the first stage, which either rejects the instance immediately, or passes it on to the next stage for further processing. This process is repeated for subsequent stages until the instance is either rejected or the final (5^{th}) stage accepts it (Fig. 9(a)). An individual stage consists of an ensemble of weak classifiers, whose outputs are combined by a weighted vote (Fig. 9(b)), and this represents Adaboost learning algorithm. Each weak classifier (Fig. 9(c)) is based on a small subject of the image features, it represents CART, which can be any function computed over a sub-window of the image (Fig. 9(d)), represents five prototypes of features that were mentioned previously in (Fig. 2).

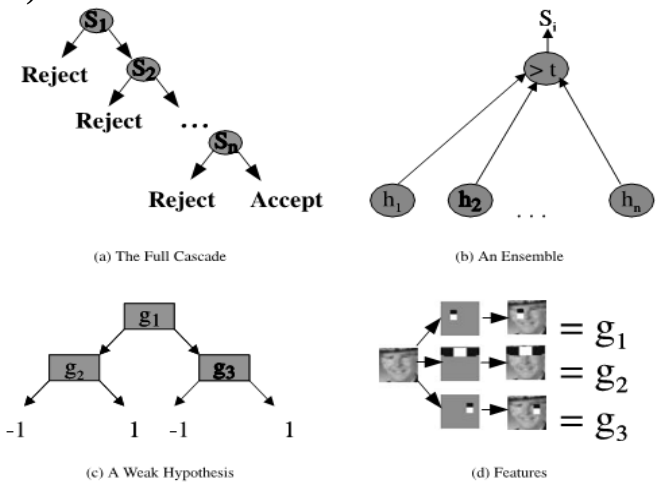


Figure (9) the architecture of the proposed cascade of five strong classifiers system from high level cascade to the low level features

7- Results

7.1. Training dataset.

The training set consisted of 450 hand labeled faces scaled and aligned to base resolution of 24 by 24 pixels, and 120 non-faced images, where each layer of cascade classifier was trained with this dataset. The faces were extracted from images downloaded from the World Wide Web. Some typical faces examples are shown in (Fig. 9). The training faces are only roughly aligned. This work done by having a person place a bounding box around each face just above the eyebrows and about half-way between the mouth an the chin. This bounding box was normalized to the base 24 by 24 window pixels. The non-faces were extracted from any image that not has any face. The dataset of test image composed of two classes, the first about 300 test images were downloaded from the World Wide Web, and the second class was belong to my family and friends.

7.2. The Cascade classifier

The cascade classifier consists of 5 stages of strong classifiers, the first strong classifier was trained with feature's type of (Fig. 2) (1a), the second with type of (Fig. 2) (1b), the third with type of (Fig. 2) (2a), the fourth with type of (Fig. 2) (2c), and the last strong classifier was trained with type of (Fig. 2) (4). Each stage was based on detector of size 24 by 24 pixels. Table 1 list the count of features for each stage of cascade classifier.

Figure (10) examples of frontal upright face images used for training.

Table 1: Count of features of (24x24) for each stage of cascade classifier

Stage of Cascade Classifier	Feature type	w/h	X/Y	Number of features
First stage	Fig. 2 (1a)	4/2	6/12	7986
Second stage	Fig. 2 (1b)	2/4	12/6	7986
Third stage	Fig. 2 (2a)	6/3	4/8	2800
Fourth stage	Fig. 2 (2c)	3/6	8/4	2800
Fifth stage	Fig. 2 (4)	4/4	6/6	4356

7.3. Experimental Results

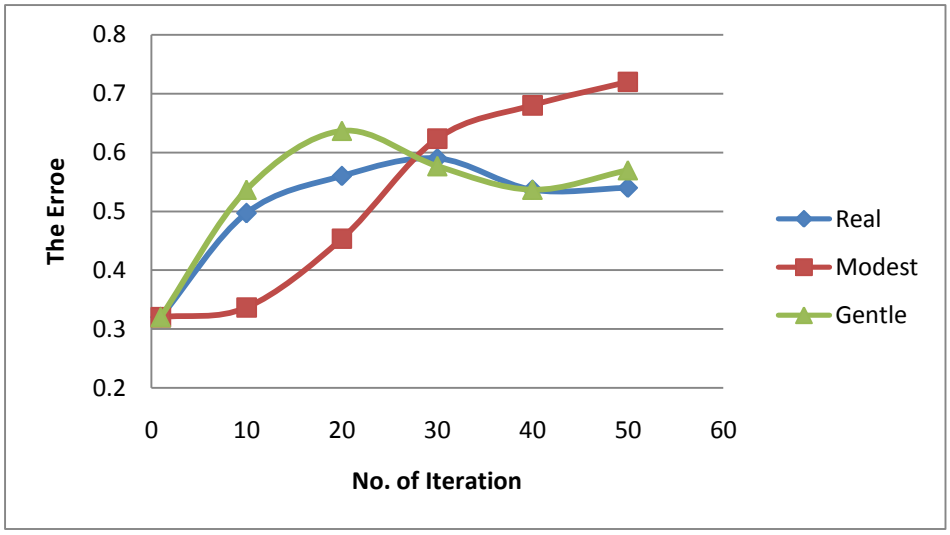
When any stage of cascade is trained with one node and once iteration then that stage would be a Stump, and otherwise, it would be a CART. (Fig. 11), (Fig. 12) and (Fig. 13) show the errors that occurred when the one stage of proposed system was Stump or CART (the iteration (rounds) starts from 10) for each one of the three types of Adaboost algorithms. These figures represent also the performance of the strong classifier with the number of rounds for training the nodes of Adaboost Algorithm.

The (Fig. 11) shows the implementation performed at the (a) first stage of the proposed system only, and (b) 2nd stage only. Figure (12) shows implementation of (a) 3rd stage (b) 4th stage only. (Fig. 13) shows implementation of 5th stage only. The observation of the three figures refers to:

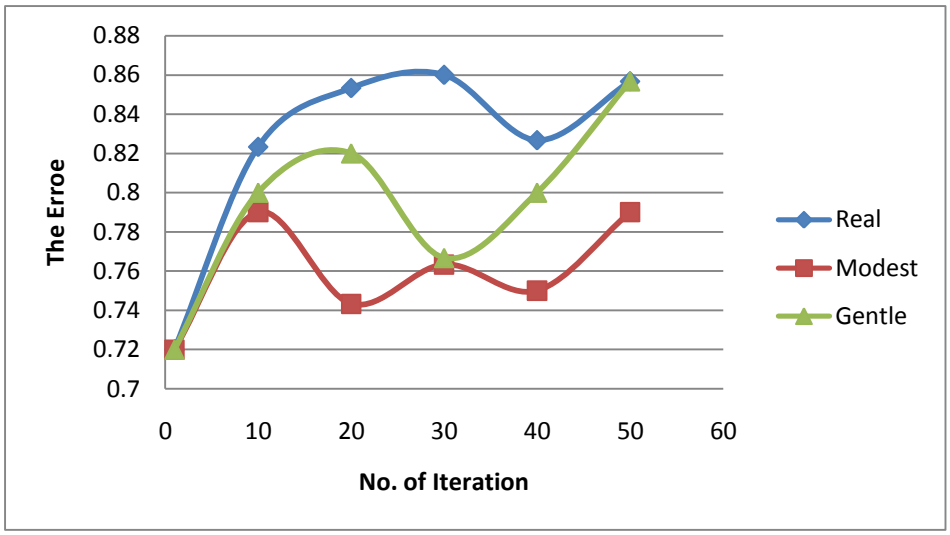
1- For the 1st and 2nd stage of cascade classifier, the best algorithm is Modest Adaboost algorithm, and best iteration count (rounds) is 10 for 1st stage and 20 for 2nd stage.
2- For the 3rd and 4th stage, the best is Real Adaboost algorithm for both stages, 50 for the 3rd stage and 10 for the 4th stage.
3- For the 5th stage, the best is Real Adaboost algorithm and 20 iterations.

Scanning the Detector

The final detector scans across the whole image at multiple scales and locations. Scaling achieves by scaling the detector itself, rather than scaling the image. This process makes sense because the features can be evaluated at any scale with the same cost. We got the result by using a set of scales factor of 1.2 apart. The detector scans across location, subsequent locations are obtained by shifting the window some number of pixels (we used 3 pixels for shifting the sub-window to right and down directions).

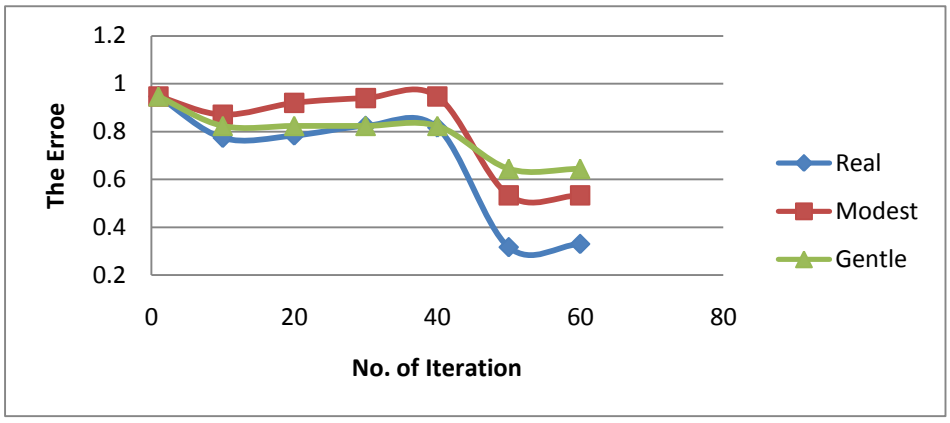


(a) Error versus count of iterations for implementation of 1st stage

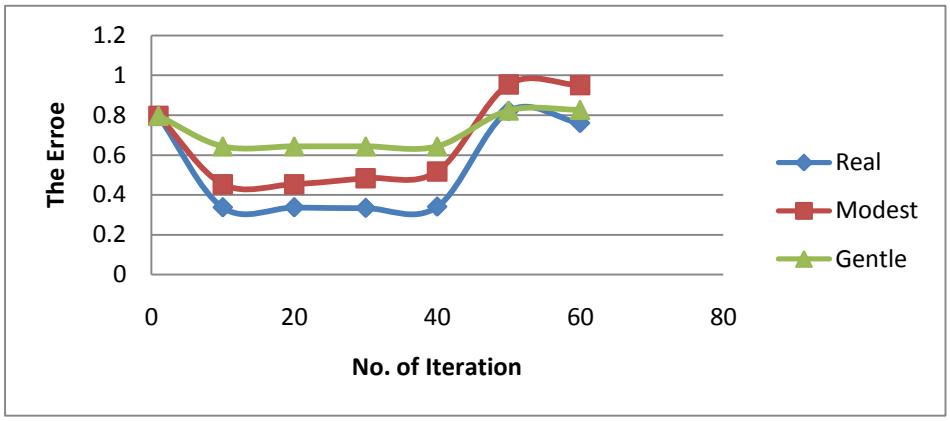


(b) Error versus count of iterations for implementation of 2nd stage

Figure (11) implementation of 1st and 2nd stages of proposed cascade system



(a) Error versus count of iterations for implementation of 3rd stage



(b) Error versus count of iterations for implementation of 4th stage

Figure (12) implementation of 3rd and 4th stages of proposed cascade system

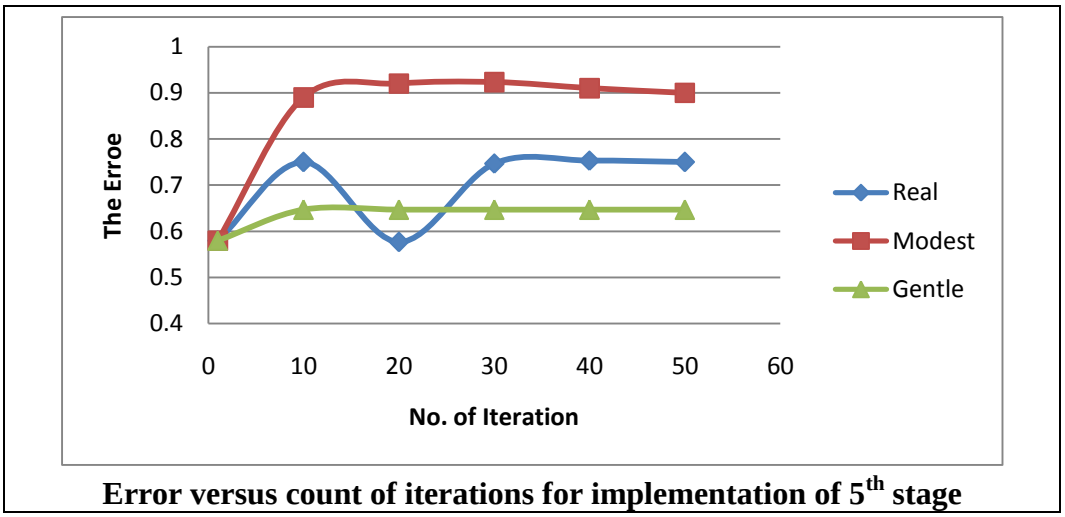


Figure (13) implementation of 5th stage of proposed cascade system

Detecting the faces in image

Now by depend on the best counts of the rounds and two algorithms (Modest and Real) that were gotten from the empirical analysis of Figures (11, 12 and 13) and got low detection error, we can construct the cascade strong classifiers with those counts of training rounds and two algorithms. For detecting the face in particular image, the first job, we normalize the size of that image to 150x150 pixels, then scan the whole image with 24x24 sub-windows, scale these sub-windows by 1.2 factors for each scanning process. Collect all these sub-windows of different scales and entering it into the proposed cascade classifier system.
(Fig. 14) (a-d) shows the results of detected faces for images contained at faces which belong training set. (Fig. 15) (a-f) shows the results of detected faces for images contained at the faces which not belong to the training set. The (Fig 15) (b) and (d) shows the errors that happened cause the rotation of faces in these images.
Tables (2) and (3) list counts of sub-windows that are entered to and exit from each strong classifier for images of figure (14) and figure (15) respectively. The rows represent the count of sub-windows for each figure, the 1st column is the name of figure, the 2nd is the count of sub-windows that is containing at the features and would be entered to the first stage of cascade classifier, the 3rd is the count of sub-windows that accepted by 1st stage and would be the input of 2nd stage of cascade classifier, and so on. For instance, the first row of table (3) is the information of image of (Fig. 15) (a), the count of sub-windows are 4824 WxH sub-windows, all those sub-windows are entered to 1st stage, the output of 1st stage would be 499 sub-windows, the 2nd stage would accept only 39 sub-windows from those 499, and so on, at the last stage the output would be 25 sub-windows only.

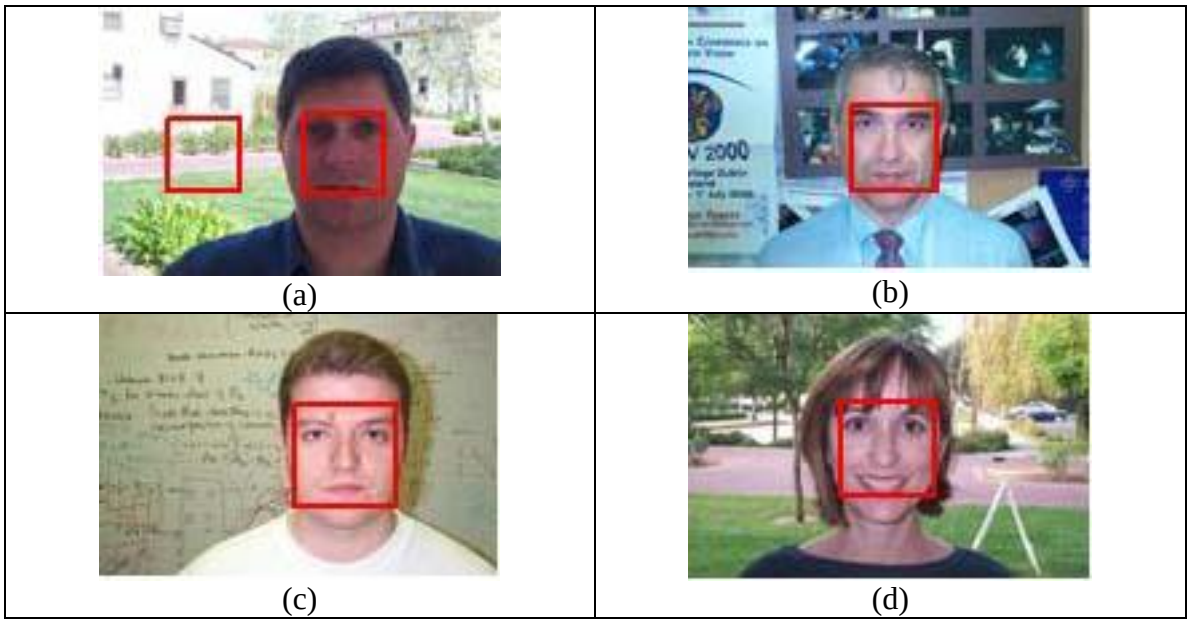
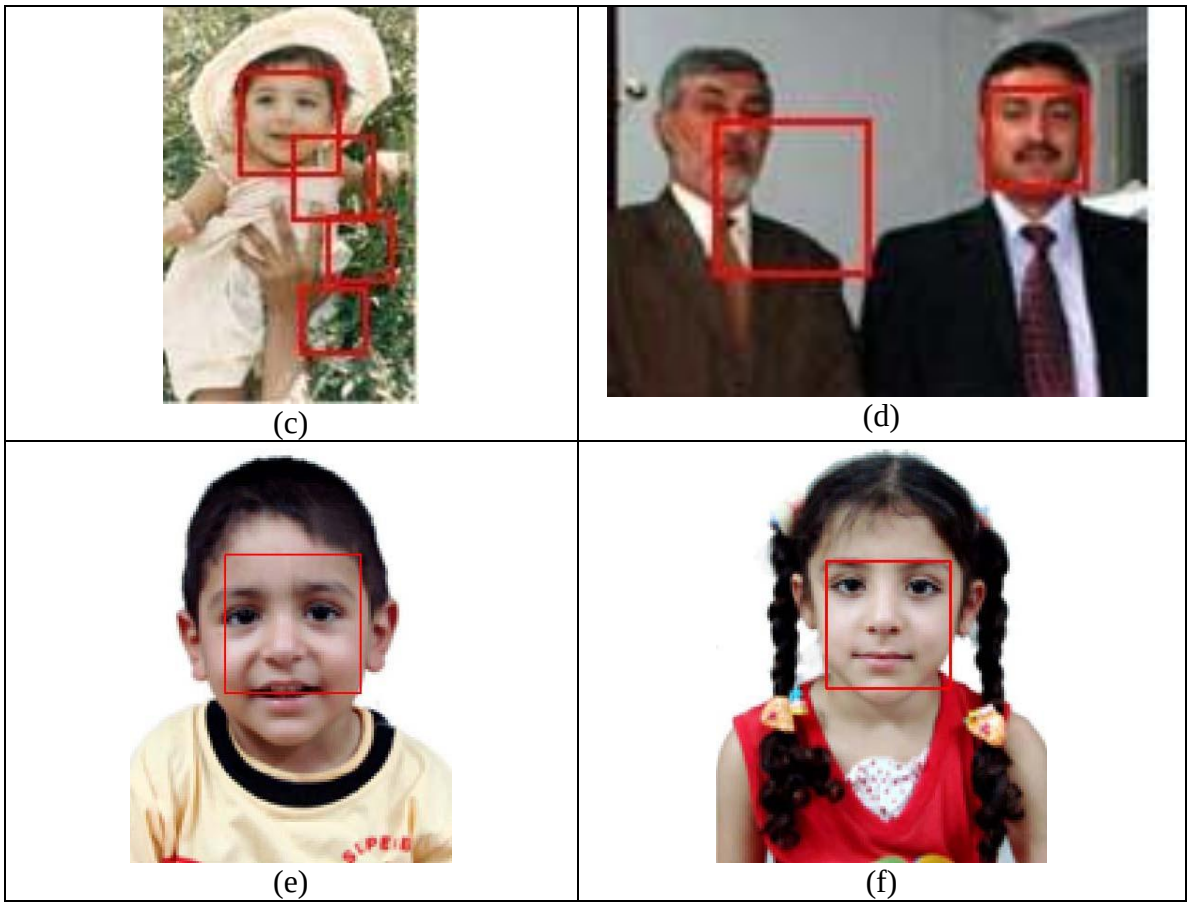


Figure (14) Results of the implementation of the proposed cascade of strong classifiers with images contained at the faces which are belong to the training dataset.

The last stage would produce many counts of overlapped sub-windows as shown in figure (16), and to eliminate these overlapped sub-windows we used small program that groups all sub-windows that are far from the others by small number, and calculate the average of that group and produce one sub-window.



(c) (d) (e) (f)

Figure (15) Results of the implementation the proposed cascade classifier system with images contained at the faces which it weren't belong to the training dataset.

Table (2): Count of sub-windows for each image in the (Fig.14)

Figure (14)	No. of windows before input	Accepted by stage 1	Accepted by stage 2	Accepted by stage 3	Accepted by stage 4	Accepted by stage 5
(a)	2704	277	58	37	27	24
(b)	2704	327	85	55	40	34
(c)	2704	213	48	33	21	19
(d)	2704	406	119	79	43	33

Table (3): Count of sub-windows for each image in the (Fig. 15)

Figure (15)	No. of windows before input	Accepted by stage 1	Accepted by stage 2	Accepted by stage 3	Accepted by stage 4	Accepted by stage 5
(a)	4824	499	39	34	26	25
(b)	4718	903	9	5	4	4
(c)	2608	556	78	45	25	21
(d)	3322	230	21	17	7	6
(e)	3960	637	185	137	85	68
(f)	3960	754	179	108	73	65

Figure (16) results of classification with overlapped windows

8- Conclusions

This paper introduced a novel and fast to compute set of haar-like features as well as a three types of Adaboost algorithms (Real, Modest, and Gentle) that applied to cascade classifier. It was shown that the overall performance could be improved by use Modest Adaboost algorithm for the 1^{st} and 2^{nd} strong classifier of cascade classifier and 20 rounds in the training. And use Real Adaboost algorithm with 50 rounds for the 3^{rd} stage and 20 rounds for the 4^{th} and 5^{th} strong classifier. The results of detection of faces in images that contained at faces was very high performance for the faces belong to the training dataset and with few error for images that contained at the faces that not belong to the training dataset.

The analysis shows that: the Modest Adaboost Algorithm is suitable for the two-horizontal (10 rounds) and a two-vertical rectangle features (20 rounds). And Real-Adaboost Algorithm is suitable for the three-horizontal (50 rounds), three-vertical (10 rounds) and four rectangles features (20 rounds).

References

[1]. P. Viola and M. Jones (2001). Rapid object detection using a boosted cascade of simple features. In IEEE Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii.

[2]. A. Kasinski and A. Schmidt (2010). The architecture and performance of the face and eyes detection system based on the haar cascade classifiers. Pattern Anal Applic, Vol.13: 197-211.

[3]. R. Lienhart and S. Clara (2002). An extended set of haar-like features for rapid object detection. IEEE ICIP, Vol. 1: pp. 900-903.

[4]. R. Lienhart, A. Kuranov and V. Pisarevsky (2002). Empirical analysis of detection Cascade of boosted classifiers for rapid object detection. Intel Labs, Microprocessor Research Lab Technical report.

[5]. P. Viola and M. Jones (2004). Robust real-time face detection. International Journal of Computer Vision 57(2): 137-154.

[6]. R. Timofeev (2004). Classification and Regression Trees (CART) Theory and Applications. Master thesis, Statistics and Economics, Humboldt University, Berlin.

[7]. Y. Freund and R. Schapire (1999). A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence, 14(5):771-780.

[8]. J. Thongkam, G. Xu, Y. Zhang, and F. Huang (2008). Breast cancer survivability via Adaboost algorithms. Conference in Research and Practice in Information Technology (CRPIT), Vol. 80.

[9]. S. Charles Brubaker, JianxinWu Jie Sun, Matthew D. Mullin and James M. Rehg (2008). On the design of cascades of boosted ensembles for face detection. International Journal of Computer Vision 77: 65–86.

Recived ……..……. (16/8/2010)
Accepted ……...…..……... (2/1/2011)