The Effectiveness of Haar Features and Threshold in Multiple Face Detection Systems Based on Adaboost Algorithm

Prepared by

Khadija Mustafa Khalil Al-Noori

Supervised by

Prof. Nidal .F Shilbayeh

A Thesis Submitted in Partial Fulfillment of the Requirements for the Master Degree In Computer Science

Department of Computer Science

Faculty of Information Technology

Middle East University

Amman – Jordan

July, 2011

Middle East University

Authorization Statement

I, khadija Mustafa khalil Alnoori, authorize Middle East University to supply hard and electronic copies of my thesis to libraries, establishments, or bodies and institutions concerned with research and scientific studies upon request, according to the university regulations.

Name: Khadija Mustafa Khalil Al- Noori

Date: 25/ 7 / 2011

Signature: 

جامعة الشرق الأوسط

التفويض

أنا الموقعة أدناه (خديجة مصطفى خليل النوري) أفوض جامعة الشرق الأوسط بتزويد نسخ من رسالتي للمكتبات الجامعية أو المؤسسات أو الهيئات أو الأشخاص المعنية بالأبحاث والدراسات العلمية عند طلبها.

الاسم :خديجة مصطفى خليل النوري

التاريخ :2011/7/25

التوقيع : 

DISCUSSION COMMITTEE

DECISION

This dissertation was discussed under title:

"The Effectiveness of Haar Features and Threshold in Multiple Face Detection Systems Based On Adaboost Algorithm "

It was approved on 25 / 7/ 2011

Discussion Committee

Signature

Prof. Nidal .F Shilbayeh

Dept of Computer Science

Faculty of Information Technology

Middle East University

Prof. Mohammad M. Al-Haj Hassan

Dept of Computer Science

Faculty of Information Technology

Middle East University

Dr. Abdel Fatah A. Yehya

Dean' Faculty of Science and Information Technology

AlZaytoonah University of Jordan

DEDICATION

I dedicate this dissertation to my parents, Mustafa and Ibtisam, who first planted the seeds of knowledge and wisdom in me. From my birth and throughout the development of my life, they have encouraged me with love and care to seek out knowledge and excellence. They challenged me to pursue my dreams, which led me to the completion of this endeavor.

ACKNOWLEDGEMENTS

I wish to acknowledge my supervisor Dr. Shilbayah, who offered me guidance and assistance throughout this process. I must credit my Uncle Saad who gave me help, hope and encouragement along the way. I want to thank my sister Mariam and brother in law Ahmad, who provided help and support, again and again. Finally, I wish to thank all my friends and family, who helped me by contributing in many ways, big and small.

The Effectiveness of Haar Features and Threshold in Multiple Face Detection Systems Based on Adaboost Algorithm

Prepared by

Khadija Mustafa Khalil Al-Noori

Supervised by

Prof. Nidal .F Shilbayeh

Abstract

The problem of face detection is still a standing problem in the research area. One of the most famous methods that is successful is the Viola & Jones approach. In this thesis, 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. There are 8 different Haar features, which can make 6 systems that would contain 4, 4, 5, 6, 7, or 8 of these Haar features. The two methods used for computing thresholds are the average of means and the optimal threshold methods. 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 MIT+CUM database. Some recommendations have been suggested according to the Haar features and the computed threshold to improve the face detection of Viola Jones approach.

فعالية كل من مزايا الهار والعتبة في أنظمة إكتشاف الوجه المتعددة إعتماداً على خوارزمية Adaboost

إعداد

خديجة مصطفى خليل النوري

إشراف

الدكتور نضال فوزي شلباية

الملخص

مشكلة اكتشاف الوجه لا تزال قائمة في مجال البحوث الى الآن. أحد أشهر الأساليب الناجحة المستخدمة في السنوات العشر الماضية وحتى الآن هو نهج فيولا وجونز.و في هذه الأطروحة تم تصميم أنظمة اعتمادا على هذا النهج لقياس كفاءة أنواع مختلفة من Haar feature والمقارنة بين نوعين من طرق حساب حد العتبة .

يوجد هناك 8 انواع من Haar feature والتي من خلالها نستطيع بناء 6 أنظمة تحتوي على 8,7,6,5,4,4 من ال Haar feature . إن الطريقتين المستخدمتين لحساب حد العتبة هما حساب المعدل وحد العتبة المثالي.

والأنظمة المقترحة دربت باستخدام قاعدة بيانات تحتوي على 350 صورة مختارة يدويا تحتوي على وجوه والقسم الأخر لا يحتوي عليها ومن ثم يتم استخدام خوارزمية ال Adaboost لبناء كاشف ما.

كل كاشف يتكون من 3 مراحل متتالية في كل مرحلة ,نستخدم عددا عشوائيا لبناء مصنف قوي . كل المصنفات الضعيفة تحسب على أساس حد العتبة قبل دخول خوارزمية adaboost فإذا نجحت الصورة في عبور هذه المراحل الثلاثة فإنها ستصنف على أنها وجه . وقد تم اختيار الكاشف على قاعدة بيانات MIT+CUM وقد اقترحت بعض التوصيات طبقا لل Haar features وحسب العتبة لتحسين استغلال نهج فايولا جونز في اكتشاف الوجوه .

Content

Page No.

List of Tables

Page No.

List of Figures

Page No.

Chapter 1

Introduction

1.1 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.

Face detection is considered a part of object detection as in Viola & Jones (2001), Pham et al.,(2005); 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. Face detection is also used in biometrics. A biometrics system is an automated method of recognizing an individual based on their unique physical or behavioral characteristics. Biometric systems use the biological features of human for user authentication.

Face detection is an extensive research field. Numerous methods have been investigated and there are several ways that can be used to categorize methods of face detection. One of them presented in Krishna (2009) reviewed the categories in Yang et al (2002), in which the majority of the face detection techniques can be broadly classified. It categorizes them according to the methods they use as the four types below:

1- Knowledge based methods : Encode human knowledge of what constitutes a typical face (usually, the relationships between facial features)(features)
2- Feature invariant approaches: Aims to find structural features of a face that exists even when the pose, viewpoint, or lighting conditions vary.

3- Template matching methods: Several standard patterns stored to describe the face as a whole or the facial features separately.

4- Appearance based methods: The models (or templates) are learned from a set of training images which capture the representative variability of facial appearance.

In the Appearance-Based Methods, there are train classifiers using positive (and usually negative) examples of faces, representation**,** preprocessing**,** train a classifier**,** Search strategy**,** post processing and View-based. There are a lot of the Appearance-Based Methods or Classifiers like: Neural network which is Multilayer Perceptron, Principal Component Analysis (PCA), Factor Analysis, Support vector machine (SVM), Mixture of PCA, Mixture of factor analyzers, Distribution-based method, Naïve Bayes classifier, Hidden Markov model, Sparse network of winnows (SNoW), Kullback relative information, Inductive learning: C4.5 and Adaboost. These techniques are categorized as shown in table 1.1:

Approach	Related work
Knowledge based	
	Multi resolution rule based method Yang and Huang (1994)
Feature based	
- Facial Features	Grouping of edges Leung et al. (1995) Yew and Cipolla (1997)
- Texture	Space Gray Level Dependence matrix of face pattern Dai and Nakano (1996)
- Skin Color	Mixture of Gaussian Yang and Waibel (1996) McKenna et al. (1998)
- Multiple Features	Integration of skin color, shape and size Kjeldsen and Kender (1996)
Template Matching	
- Predefined face templates	Shape template Craw et al. (1992)
- Deformable Templates	Active Shape model Lanitis et al. (1995)
Appearance based	
- Eigenface	Eigenvector decomposition and clustering Turk and Pentland (1991)
- Distribution-based	Gaussian distribution and multilayer prediction Sung and Poggio (1998)
- Neural Network	Ensemble of neural networks and arbitration schemes Rowley et al. (1998)
- Support Vector Machine SVM	SVM with polynomial kernel Osuna et al. (1997)
- Naive Bays Classifier	Joint statistics of local appearance and position Schneiderman and Kanade (1998)
- Hidden Markov Model	Higher order statistics with HMM Rajagopalan et al. (1998)
- Information Theoretical Approach	Kullback relative information Lew (1996), Colmenarez and Huang (1997)

Table 1.1: Summary of face detection techniques Yang et al. (2002)

The other way that can be used to categorize the face detection, is by dividing the Face Detection into two parts depending on the type, whether it was a single image or a video. The video is further divided into color and gray scales. The single image is also divided

into color and gray scales. The gray scale could be divided into several parts like: Upright Frontal, Pose Rotation, and Occlusion as figure 1.1 shows .

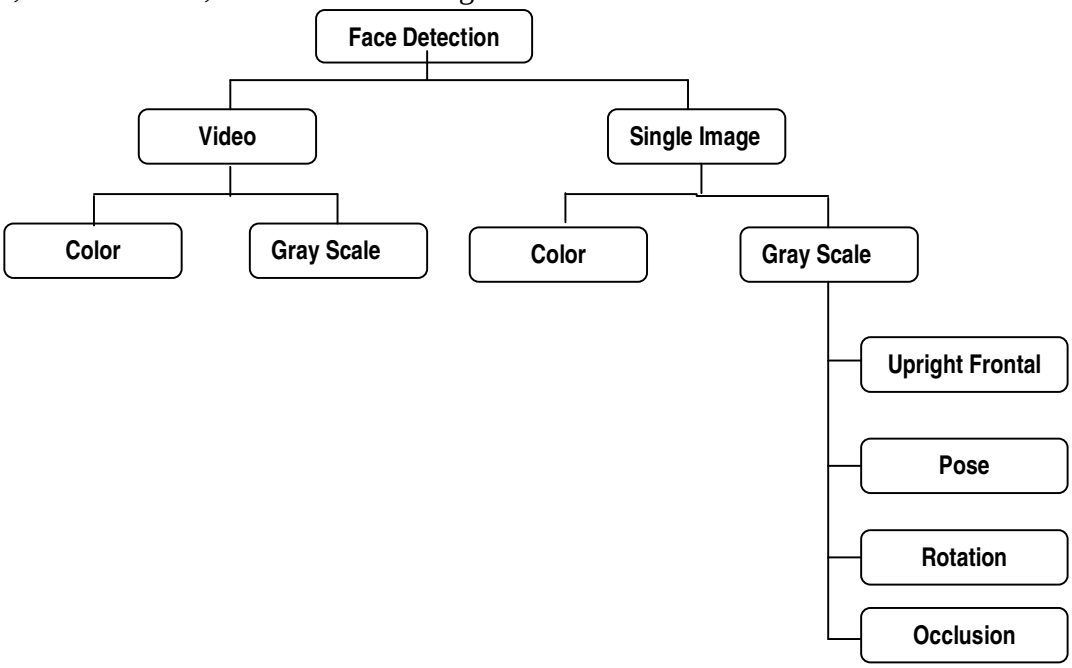


Figure 1.1: Describes the category of face detection depending on their types

Early face-detection algorithms were focused on the detection of upright frontal human faces, but now the newer algorithms that have been developed are able to solve more general and difficult problems of multi-view face detection, such as the detection of faces that are either rotated along the axis from the face to the observer (in-plane rotation), or rotated along the vertical or left-right axis (out-of-plane rotation), or both. Although a face detection module is typically designed to deal with single images, its performance can be further improved if a video stream is available.

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.

Ten years ago, Paul Viola and Michael Jones published a famous approach that achieved a great result, and is being used extremely in researches because of its low computational complexity (unlike other techniques), containing only simple operations such as addition, subtraction and comparison. Because of the algorithm's better performance it is implemented in both Intel IPP (2008) and Open CV (2008).

Intel built xml files in open compute vision library that are based on the Viola-Jones' approach to increase the robustness in real time face detection, and facilitate it by reducing the time for users, because it takes a lot of time to train. There are different types of these xml files based on window size: 20, 24, furthermore there are different algorithm types: Adaboost, gentle Adaboost or facial, and the position can be either: frontal or profile. There are a lot of applications that can benefit from such systems including security and surveillance, robotics, human computer interaction, environment monitoring and advanced driver assistance in the automotive industry.

1.2 Statement of the Problem

Face detection is one of the extensive research areas in which researchers can find a lot of topics to discuss. Furthermore, it comes with a lot of challenges, like Human faces that have invariant characteristics. In fact, a person’s face can change drastically during short periods of time (from one day to another), so you can only imagine what would happen in

long periods of time (a difference of months or years). Some of the most important problems include changes in illumination, variability in facial expressions, the presence of accessories (glasses, beards, etc); and finally, the rotation of the face, which may change many facial characteristics.

We can abstract the reasons that make the face detection system difficult in:

- Pose(Out-of-Plane Rotation): due to mismatch in facial position, facial expression, scale, frontal, 45 degree, profile and upside down
- Facial expression: face appearance is directly affected by a person's facial expression.
- Orientation (In-Plane Rotation): face appearance vary based on the rotation around the camera's optical axis
- Deblurring: due to mismatch in camera focus, camera lenses, and camera resolution.
- Imaging conditions: lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, gain control, lenses), resolution.
- Illumination: due to mismatch in lighting conditions in both indoor and outdoor environments.
- Occlusion : faces may be partially occluded by other objects
- Presence or absence of structural components : beards, mustaches, and Glasses

In the paper of “Robust Real-time Object Detection” that was published in 2001, the authors constructed a frontal face detection system that didn’t mention the details of the method that was used to compute the threshold value after. A few years later they published it.

There are a lot of researches that depend on the ideas contained in that paper. Researchers created different ways to improve this technique to find the threshold. Two of these techniques will be compared in this thesis, to expand the research area and to show how decreasing, or increasing the features can affect the results of the original paper. Some features will be removed, while others will be added to the original paper.

1.3 Thesis Objectives

The objectives of this thesis are as follows :

- Design and implement a simple face detection detector that is based on almost all of the ideas of Viola/Jones's system.
- Several systems will be divided into two different groups; each group will compute the threshold differently. The results will be compared to find the best detector based on the threshold.
- Each system will be compared based on the changes of the features, to determine which is the best detector based on the feature type in the two groups.

1.4 Thesis Organization

In addition to this chapter, the thesis includes four other chapters:

Chapter 2 provides an overview of face detection using Haar feature and Adaboost algorithm's methods and threshold methods, with listing and explaining different related works in the area of the system used. Chapter 3 describes the proposed system in details with respect to its different stages, modules, and Algorithms. Chapter 4 contains the experimental results of the designed system and it's also discusses the results with other systems ; Finally, chapter 5 contains the conclusion, future work suggestions, and some recommendations.

Chapter 2

Literature Review

2.1 Face detection

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 the 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.

2.2 Face Detection Using Haar Feature and Adaboost Algorithms Method and Threshold

There are several different methods that have been successfully used to achieve high detection with an acceptable number of false positives in face detection. This system is considered to be successful, because the detection algorithm possesses the two key features: accuracy and speed(Roy & Marcel 2009). It also has a high detection rate, with a low number of false positives.

2.2.1 Feature and Integral Image

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 (Viola & Jones 2001.)

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.

Haar-Like Features

Haar-like features, or 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.

Figures 2.1 and 2.2, display the basic Haar features , and the fifteen of famous features respectively.

Figure 2.1: A set of basic Haar-like features

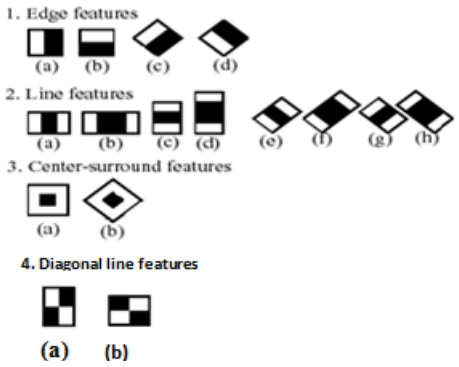


Figure 2.2: A set of extended Haar-like features.

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 equation 2.1.

f(x) = Sum$_{\text{white rectangle}}$ (pixel gray level) – Sum$_{\text{black rectangle}}$ (pixel gray level)

or

f(x) =∑ (pixels in white area) – ∑ (pixels in black area) ……..(2.1)

Compared with raw pixel values, Haar-like features can reduce/increase the in-class/out-of-class variability, thus making classification easier. The motivation behind using rectangular features as opposed to more expressive steerable filters is due to their extreme computational efficiency. Steerable filters are excellent for the detailed analysis of boundaries, image compression, and texture analysis. The Haar features computation needs a huge space memory to do this operation in order to reduce computation time.

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), as shown in equation 2.2 and figure 2.3.

However, it can quickly be computed in one pass through the image and can be calculated by:

$$P(x, y) = \sum_{x' \le x, y' \le y} i(x', y') \qquad \text{……..(2.2)}$$

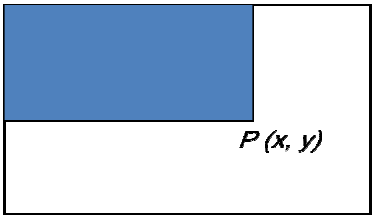


Figure 2.3: integral image for point (x,y)

Or by using the following two recurrences, where $i(x,y)$ is the pixel value of original image at the given location and $s(x,y)$ is the cumulative column sum, the system can calculate the integral image representation of the image using the equations 2.3, 2.4 in a single pass as shown in figure 2.4 .

$$s(x,y) = s(x,y\text{-}1) + i(x,y) \qquad \text{……..(2.3)}$$

$$ii(x,y) = ii(x\text{-}1,y) + s(x,y) \qquad \text{……..(2.4)}$$

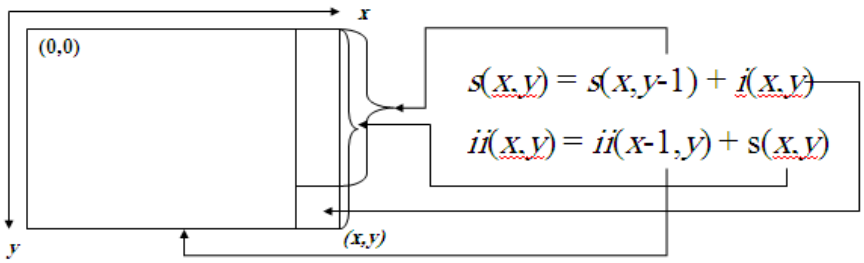


Figure 2.4: The Integral Image Calculation Process

Using the integral image representation, one can compute the value of any rectangular sum in constant time, at any position or scale, using only 4 lookups. For example, the integral sum inside rectangle D in figure 2.5 can be computed using equation 2.5:.

$$\left.\begin{array}{l} P_1=A, P_2=A+B, P_3=A+C, P_4=A+B+C+D \\ P_1+P_4-P_2-P_3=A+A+B+C+D-A-B-A-C=D \end{array}\right\} \qquad \text{……..(2.5)}$$

Or by equation 2.6:

$$ii(4) + ii(1) - ii(2) - ii(3) \quad \text{……..(2.6)}$$

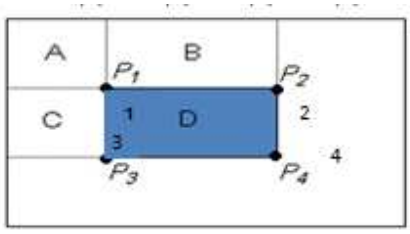


Figure 2.5: An explanation on how to find
the sum of d by using 4 references

Therefore, as a result this allows computing two-, three-, and four-rectangular features in figure 2.1 with 6, 8 and 9 array references respectively.

2.2.2 Adaboost Algorithm

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

Where the **Weak classifier** is defined to be a classifier which is only slightly correlated with the true classification (it can label examples better than random guessing).

And **Strong classifier** is a classifier that is arbitrarily well correlated with the true classification .

The origins of boosting lie in PAC learning theory (valiant 1984). The adaptive boosting (Adaboost) algorithm exists in various varieties (freund and schapire 1996, 1997). In addition, there are three modifications of the original algorithm that were proposed: Gentle-, Logit-, and Real Adaboost. (Friedman et al. 2000)

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.

A. 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 used the weak classifier to consider the input of the Adaboost. Therefore, it needs to build the simple classifier (weak classifier) before applying the Adaboost algorithm.

To determine the weak classifier, first we must compute the threshold value. There are several methods used in different researches.

Threshold

The threshold is a value used to separate the value of face and non-face to build the weak classifier as the input of the Adaboost algorithm the threshold gets. It is important because it is the base of the weak classifier build. The threshold can be categorized into two types according to the number of value it uses, as a single threshold and multiple thresholds

Single Threshold

There are several types mentioned in different papers that are used to compute the single threshold value. Here are some of these types:

- In the lab manual Image Based Recognition and Classification, they mention two basic methods for determining the threshold value associated with a feature vector. Both methods rely on estimating two probability distributions - the distribution of the values of the feature when applied to the positive samples (face data), and to the negative samples (non-face data). With these distributions, the threshold can be determined by either one of the two ways :

- Taking the average of their mean's single threshold, as in equation 2.7, where μi+ is the mean of positive samples and μi− is means of the negative samples.

  threshold = (μi+ + μi−)/2 ……..(2.7)

- Finding the crossover point. This cross-over point shown in figure 2.6 corresponds to equation 2.8 where f_i single feature vector p is probability distributions :

  f_i such that $p(f_i|\text{non-face}) = p(f_i|\text{face})$ ……..(2.8)

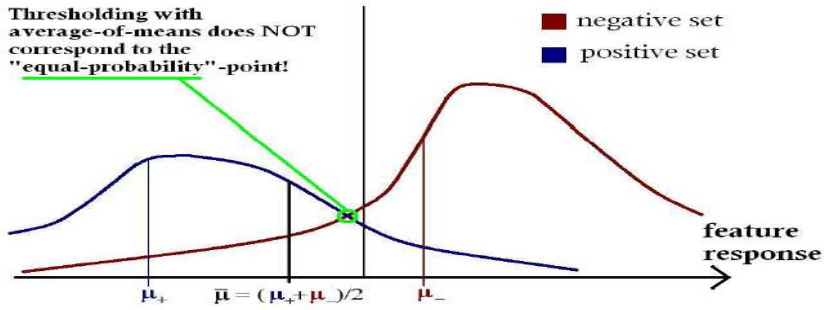


Figure 2.6: An example of how the distribution of feature values for a specific feature may look like over the set of all training samples.

- The optimal threshold is another method to compute the signal. It uses an algorithm which chooses the value that best separates the faces from the non-faces.

There are five steps for this approach per each weak classifier:

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

$$\text{threshold}(t) = \text{Hfaces}(t) - \text{Hnonfaces}(t) \qquad \ldots\ldots..(2.9)$$

- Another way to find the threshold is to divide the arbitrary points in two groups, one called the positive set and the second called the negative set. Examples are classified as positive, if the following condition apply:

$$\min\left(\left\{P_i^+, i = 1,\ldots,N_+\right\}\right) - \max\left(\left\{P_j^-, j = 1,\ldots,N_-\right\}\right) > V$$

OR

$$\min\left(\left\{P_j^-, j = 1,\ldots,N_-\right\}\right) - \max\left(\left\{P_i^+, i = 1,\ldots,N_+\right\}\right) > V$$

V is the minimum separation threshold between the two point groups, Pi^+ a point from the positive group, Pj^- a point from the negative group, and N+ and N- are the number of points in their respective groups. In a linear representation of the pixel values, an example is classified as positive if the two point groups are separated by at least the value of threshold V (see figure 2.7). Negative Examples are those that do not respect these characteristics: Values of the control-points of the two groups are interleaved (see figure 2.7).

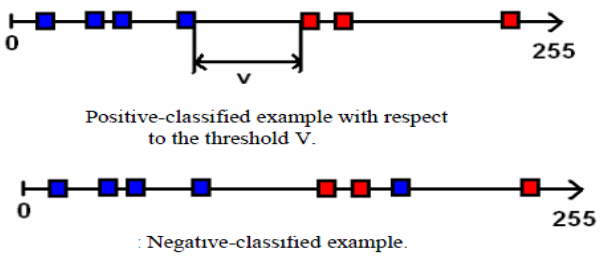


Figure 2.7: An example of the two groups and the threshold representation.

- The threshold for the feature is calculated using the probability density function (PDF) (see figure 2.8 (a) of the face and non face variables). From that, the corresponding face and non face PDFs, the respective cumulative distribution function (CDF) in figure 2.8 (b) is derived, and the value of the variable at which these thresholds have the maximum Euclidian distance is considered as the threshold value.

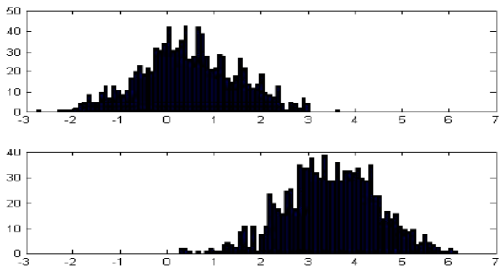
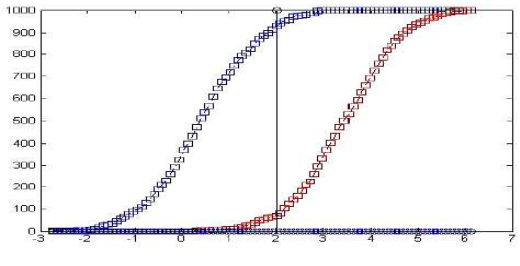


Figure 2.8: (a) PDF of the face and clutter values



2.8 (b) Threshold calculation at maximum distances CDF

Multiple Thresholds

There are several methods proposed to compute multiple thresholds. Some of them are:

- In (2006) the thresholds where these two distribution PDFs intersect in magnitude, are equally probable, by parameterizing

  1. Parameterizing the Gaussian case:

  Assuming that the two distribution projections are somewhat Gaussian, one can model the two thresholds by knowing the mean and variance of the two sets for that particular feature. (See Figure 2.9)

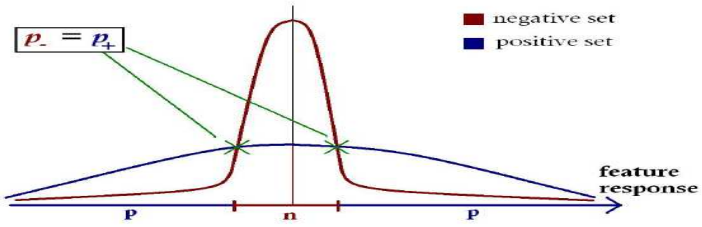


Figure 2.9: By parameterizing the two distributions with a Gaussian model, one can find the two intersection points that will represent the two thresholds. The parameterized approach is less sensitive to noise than statistics/numerical ways to find the two intersecting points of the histogram plots

2. The MAP-classifier and Response Binning:

The generalization of this procedure leads to a method for determining thresholds for multimodal PDFs. The propose is that the two histograms of the feature for the positive and negative examples, H+(fi) and H−(fi) are computed (for a certain number of bins) and their relative inequalities (the MAP) are then stored:

$$\text{MAP(fi)} = \begin{cases} 1 \text{ if } H+(\text{fi}) > H-(\text{fi}) \\ -1 \text{ otherwise} \end{cases} \qquad \text{……..(2.10)}$$

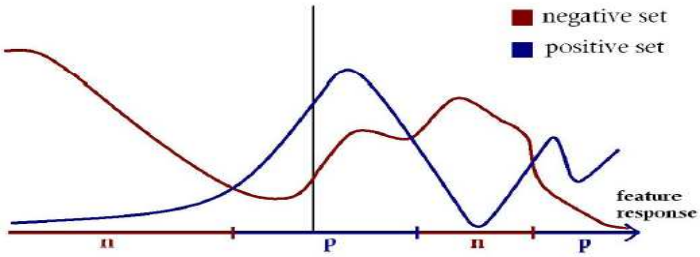


Figure 2.10: This figure illustrates the case of multimodal PDFs (given by feature fi) for the positive and negative training set. By following the largest probability while classifying, one can adapt the multi-threshold principle also for this general case

- For each possible feature and given training set, the weak learner determines two optimal values (thresholds), to ensure no training sample is misclassified as shown in Figure 2.11 there are two values for the threshold, big and small threshold.

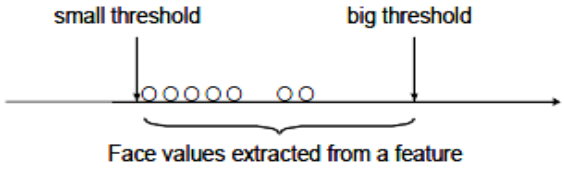


Figure 2.11: The determination of thresholds by weak learner.

An easy way to link the weak learner and Haar features is to assign one weak classifier to one feature, so the Adaboost algorithm will select the feature that provides the best separation between positive and negative at each round. Example: the form of given a single feature vector, f_i, evaluated at x, p_i Parity indicating the direction of the inequality sign , the output of the weak classifier $h_i(x)$ is either -1 or 1. The output depends on whether the feature value is less than a given threshold θ_i as shown in equation 2.11.

$$h_i(x) = \begin{cases} 1 & \text{if } p_i f_i(x) < p_i \theta_i \\ -1 & otherwise \end{cases} \qquad \text{……..(2.11)}$$

The result from just one weak classifier does not give enough information, but several classifiers combined into a strong classifier that can determine whether a detection window contains a face or not .

B. 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.

The hypothesis of Viola, Jones which was born by experiment shows that a very small number of these features can be combined to form an effective classifier. The main challenge is to find these features. They suggest using a variant of the Adaboost to select the features and to train the classifier. The algorithm of Adaboosting is published in the Paul Viola, Michael Jones "Robust Real-time Object Detection".

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  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. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Table 2.1: The boosting algorithm for learning a query online. T hypotheses are constructed each using a single feature. The final hypothesis is a weighted linear combination of the T hypotheses where the weights are inversely proportional to the training errors.

There are several types of the Adaboost algorithms used for boosting, Gentle-, Logit-, and Real Adaboost. Lienhart, Kuranov, and Pisarevsky (2003) 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, Chen, Zhang, Zhu and Li, (2004) extended the idea and

proposed a novel algorithm to generate a unified output for a series of Adaboost classifiers.

Kim & Kee (2005) used a cascaded classifier trained by gentle Adaboost algorithm, one of the appearance-based pattern learning method. Mita, Kaneko and Hori (2005) joint Haar-like features using Adaboost, Meynet, Arsan, Mota and Thiran (2007) fast and effective multi-view face tracking algorithm based on Adaboost algorithm. Later, Shihavuddin, Arefin, Ambia, Haque & Ahammad (2010) proposed working together to combine Adaboost algorithm with the hyper planes concept creating a noble efficient detection system for real time use.

2.2.3 Cascade classifier

A cascade of classifiers is a degenerated decision tree, where at each stage classifier it is trained to detect most of the objects of interest (frontal faces as example), while rejecting a certain fraction of the non-object patterns.

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 .

Training a Cascade of Classifiers

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 an optimization framework by:

1. the number of classifier stages,
2. the number of features in each stage, and
3. 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.

Where **False positive rate 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.

$$\textit{False positive rate}\ (\alpha) = FP / (FP + TN) \quad \text{.....(2.12)}$$

Or

$$\textit{False Positive Rate}\ (A) = 1 - \text{Specificity} \quad \text{.....(2.13)}$$

Where

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}} \quad \text{.....(2.14)}$$

2.3 Related Works

Face detection generally has been an attractive field for researchers for many years. The contribution of Viola & Jones in face detection made researchers interested in this field in the last few years. The research combined one or more of this contribution with other face detection methods to enhance their result, develop further, change parts to obtain

new results, or use it in other systems. The following literature survey shows the researchers who researched in this field and what they had achieved:

Rowley, et al., (1998) presented a neural network-based upright frontal face detection system. A retinal connected neural network examines small windows of an image, and decides whether each window contains a face. The system arbitrates between multiple networks to improve performance over a single network. To collect negative examples, they used a bootstrap algorithm, which adds false detections into the training set as training progresses. The system can detect between 77.9% and 90.3% of faces in a set of 130 test images, with an acceptable number of false detections. It has been tested on a wide variety of images, with many faces and unconstrained backgrounds. A fast version of the system can process a 320x240 pixel image in 2 to 4 seconds on a 200 MHz R4400 SGI Indigo 2.

Viola and Jones, (2001) presented an approach for object detection which minimizes computation time while achieving high detection accuracy. The approach was used to construct a face detection system, which is approximately 15 times faster than any previous approach. This paper brings together new algorithms, representations, and insights which are quite generic and may well have broader application in computer vision and image processing. There are three key contributions. The first is the introduction of a new image representation called the “Integral Image,” which allows the features used by their detector to be computed very quickly. The second is a learning algorithm, based on Adaboost, which selects a small number of critical visual features and yields extremely efficient classifiers. The third contribution is a method for

combining classifiers in a “cascade,” which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions. A set of experiments in the domain of face detection are presented. Finally this paper presents a set of detailed experiments on a difficult face detection dataset.

Lienhart & Maydt, (2002) introduced a novel and a fast-to-compute set of rotated Haar-like features as well as a novel post-optimization procedure for boosted classifiers. It was shown that the overall performance could be improved by about 23.8% of which 10% could be contributed to the rotated features and 12.5% to the stage post-optimization scheme.

Lienhart et al., (2003) in their experimental results suggested that 20x20 is the optimal input pattern size for frontal face detection. In addition, they show that Gentle Adaboost outperforms Discrete and Real Adaboost. Logitboot could not be used due to a convergence problem on later stages in the cascade training. It is also beneficial not just to use the simplest of all tree classifiers, i.e., stumps. They also introduced an extended set of Haar-like features. Although frontal faces exhibit little diagonal structures, the 45 degree rotated features increased the accuracy.

Viola & Jones, (2003) extended the face detection framework proposed (2001) to handle profile views and rotated faces. As in the work of Rowley at al. 1998, and Schneiderman et al. 2000, they built different detectors for different views of the face. A decision tree is then trained to determine the viewpoint class (such as right profile or rotated 60 degrees)

for a given window of the image being examined. This is similar to the approach of Rowley et al. 1998. The appropriate detector for that viewpoint can then be run instead of running all detectors on all windows. This technique yields good results and maintains the speed advantage of the Viola-Jones detector. They detect 70.4% of the profile faces with 98 false positives and 83.1% with 700 false positives.

Zhao, et al., (2003) presented an extensive survey of machine recognition of human faces, and a brief review of related psychological studies. They have considered two types of face recognition tasks: one from still images and the other from video.

They have categorized the methods used for each type, and discussed their characteristics, including their pros and cons. In addition to a detailed review of representative work, they have provided summaries of current developments and of challenging issues.

They have also identified two important issues in practical face recognition systems: the illumination problem and the pose problem. They have categorized proposed methods of solving these problems and discussed the pros and cons of these methods. To emphasize the importance of system evaluation, three sets of evaluations were described: FERET, FRVT, and XM2VTS. Getting started in performing experiments in face recognition is very easy.

Chen, et al., (2004) proposed a method to calculate probabilistic-like output for each pixel of image. This output describes the similarity of a patch of image to the training samples. The probabilistic-like output is then used to locate the feature points using a

localization approach they proposed. Their algorithm for facial feature localization is fast, accurate and robust. It takes only about 10ms on a computer with a P4 CPU to locate five feature points including eye centers, nose tip and mouth corners. The localization accuracy is comparable with hand labeled results. And experimental results on a large size of face database (more than 12,000 images from PIE) demonstrate that the proposed method is robust to the variances of pose, illumination, expression and other appearance factors, such as glasses and beard.

Zhang et al., (2004) presented a method for face recognition using boosted Gabor feature based classifiers. Two kinds of Gabor feature for face image representation with a suggested efficient combination approach in face recognition were applied, and also presented a face recognition system by Adaboost cascade structure. Experimental results on FERET (fafb) database have proven the effectiveness of the new approach. Weak classifiers are constructed based on both magnitude and phase features derived from Gabor filters. The multi-class problem is transformed into a two-class one of intra- and extra class classification, using intra-personal and extra-personal difference images. A cascade of strong classifiers is learned using bootstrapped negative examples, similar to the way in face detection framework. Proposed method just uses 2672 features to achieve a recognition rate of 96.5%

Le & Satoh, (2004) presented a three-stage method to speed up a SVM-based face detection system. In this proposed system, a large number of simple non-face patterns are rejected quickly by the first two stages' cascaded classifiers using flexible sizes of

analyzed windows, while the last stage uses a non linear SVM classifier to robustly classify complex 24x24 pixel patterns as either faces or non-faces. For all stage classifiers, an optimal subset of over complete Haar wavelet feature set selected by Adaboost learning is used to achieve both fast and high detection rate.

Experimental results show that the two first stage cascaded classifiers rejects almost 99.88% non face patterns in which the first stage classifier contributes average up to 58.73%. Only 0.12% hard patterns are put into the final stage SVM classifier to make to final decisions. Time for background rejection is about 38% total time of detection while time of classification using SVM is about 62%.

Menezes et al., (2004) developed new human-machine interfaces for mobile robots and autonomous systems, based on computer vision techniques. The article presented an approach for real-time face recognition and tracking which can be very useful for human-robot interaction systems. In a human robot interaction environment this system starts with a very fast real-time learning process and then allows the robot to follow the person and to be sure it is always interacting with the right one under a wide range of conditions including: illumination, scale, pose, and camera variation.

This paper also presents a Pre-Learnt User Recognition System which works in almost real-time and that can be used by the robot to create a set of known people that can be recognized anytime. The system contains 13 stage cascaded classifier trained to detect frontal upright faces. Each stage was trained to eliminate 50% of the non-face patterns while falsely eliminating only 0.2% of the frontal face patterns. In the optimal case, they can expect a false alarm rate about $0.002^{13} = 8 \cdot 10^{-36}$ and a hit rate about $0.998^{13} = 0.97$

The robot has a certain number of people in the database and once a known face is found it can start following and interacting with it. Of course this system can also be used in security applications since it has the ability of tracking a set of known people.

Mita et al., (2005) described a new visual feature, called joint Haar-like feature, for face detection and show how it can be selected. The feature captures the characteristics of human faces and improves the performance of each weak classifier. A face detector is learned by stagewise selection of the joint Haar-like features using Adaboost. This is based on co-occurrence of multiple Haar-like features. Feature co-occurrence, which captures the structural similarities within the face class, makes it possible to construct an effective classifier. The joint Haar-like feature can be calculated very fast and has robustness against addition of noise and change in illumination. A small number of distinctive features achieve both computational efficiency and accuracy. Experimental results with 5,676 face images and 30,000 non-face images show that this detector yields higher classification performance than Viola and Jones' detector, which uses a single feature for each weak classifier. Given the same number of features, this method reduces the error by 37%. This detector is 2.6 times as fast as Viola and Jones' detector to achieve the same performance.

Kim, et al., (2005) presented multi-view face and eye detection, used as core technologies to face recognition and image retrieval. They use a cascaded classifier trained by gentle Adaboost algorithm, one of the appearance-based pattern learning method. Specifically, in order to detect a multi-view face, they propose a special

cascaded classifier using coarse-to-fine search, simple-to-complex search, and parallel-to-separated search. In order to detect eyes, they propose a four-step eye detection method. Using proposed methods, they got face and eye detection ratios as high as 99.5%, 88.3%, respectively for 7 different DBs including 17,018 various multi-view faces.

Chin & Suter, (2005) presented algorithm capable of reducing false positive instances without additional weak classifiers and classifier stages. While the false positives were not eliminated completely, fewer false positives indicate that fewer weak classifiers will be needed to construct an additional stage using the Adaboost algorithm for a classifier cascade to account for the generated false positives.

Nevertheless, more empirical evidence of the effectiveness of the algorithm should be obtained. The proposed bootstrap algorithm should be tested on face detectors which are trained on more extensive or more standard datasets, to allow the results to be comparable with other researches. Since the algorithm was derived heuristically, efforts should be made to explain how the proposed algorithm affects the Adaboost classifier from the standpoint of learning theory.

Deng & Su, (2006) extended the work of the Viola & Jones and achieved two contributions. Firstly, they proposed a novel feature definition and introduced a feature shape mask to represent it. The defined features are scale-invariant which means the features can be rescaled easily and reduce the performance degradation introduced by rounding the coordinates to nearest integer. In addition, the novel features are robust to whole image illumination changing, and illumination correction can be processed with

integral images of both original image and image squared. Fast feature computation methods are also introduced and present an improved cascade-structured classifier which is called fuzzy cascade classifier. The cascade-structured classifier owns the disadvantage of neglecting confidence of the prior stage classifiers while only using the binary output of prior stages.

Motivated by fuzzy theory, they expand the output of each stage to three states: face, non-face, and potential face and set probability being face to each candidate window to make full use of the information of prior stages.

Whitehill & Omlin, (2006) examined the effectiveness of using Haar features and the Adaboost boosting algorithm for FACS action unit (AU) recognition. They evaluated both recognition accuracy and processing time of this new approach compared to the state-of-the-art method of classifying Gabor responses with support vector machines. Empirical results on the Cohn-Kanade facial expression database showed that the Haar+Adaboost method yields AU recognition rates comparable to those of the Gabor+SVM method but operates at least two orders of magnitude more quickly.

Monteiroet, al., (2006) described a vision-based pedestrian detection system for robots, and autonomous vehicles. For that purpose the Haar-like features were used to discriminate pedestrians. Those features were used as input in a learning algorithm, based on Adaboost, which selects a small number of critical visual features from a larger set and yields an extremely efficient classifier. The proposed system has the following properties: a) it is able to detect pedestrians in various poses, shapes, sizes and clothing;

b) it runs in real-time with a good accuracy; and c) it is robust to lighting variation, background changes and camera motion. The missing detection of pedestrians is due to lack of contrast between pedestrians and background and to the enormous flexibility of the human body.

Moreover, a filter - Kalman Filter is going to be integrated for the tracking of the pedestrians. Since, most of the time, false positives are completely independent from the time sequence, they will decrease significantly.

Furthermore, it is also planned to train the classifier to detect other objects, like cars, traffic signs, etc

Rasolzadehet al., (2006) demonstrated the value of improving the discriminating strength of weak classifiers in the context of boosting by using response binning. Moreover, in the case of the Haar-feature that Viola & Jones utilizes for their detection systems, the common shape of the underlying probability distributions is not suitable for such an over simplified boundary. Instead, two methods based on techniques in the statistical toolbox were derived. The first method takes a parameterized approach by modeling the underlying distributions as two Gaussians, which in the case of the Haar-feature, is a good approximation for the larger part of the feature set. Using these two Gaussians, a two threshold boundary was derived that demonstrated very good performance. The second method generalized the approach to also be applicable to multi-modal probability distributions through a so called MAP-histogram.

Extending this principle to other types of weak classifiers, they introduced the concept of Response Binning. In the case of RealBoost, this proved to in fact be a local (per bin)

confidence measure for the previous MAP-classifier. The results indicate that the Response Binning approach could in principle improve any type of weak classifier that returns weak hypotheses.

Meynet, et al., (2007) proposed a system for fast multi-view face tracking in video sequences. The modeling of the face at different poses has been performed using a tree of classifiers. The first group of classifiers is computationally very efficient and they discard the background regions quickly. Then, more specific classifiers have been trained with Gaussian filters to distinguish between different poses. The output is thus the position and pose of the detected faces. A tracking algorithm based on Condensation has been used for tracking the faces in time. Some experiments show the accuracy of the system on real world sequences.

Mohd Zin, (2007) implemented the Genetic Algorithm inside Adaboost framework to select features, and the seven new feature types, which have been proposed to increase the quality of the feature solutions set. The results of this research confirmed that Genetic Algorithm is very useful as an evolutionary searching algorithm in a very large search space. The new seven feature types also influenced the selections of features by Adaboost algorithm. They have shown their different significance in the trained cascades of boosted classifiers.

Berggren & Gregersson ,(2008) investigated the possibility to use the GPU of a mobile platform to detect faces and adjust the focus of the camera accordingly.

The Adaboost training proved to be challenging in a number of ways, and was constantly refined throughout the duration of the thesis project. Using the results from the training, a detector prototype was implemented in a simulated OpenGL ES 2.0 environment.

Requirements for such an application are given, and different methods for face detection are described and evaluated according to these requirements.

A GPU implementation of the cascade classifier is described. A CPU reference implementation with results is also presented. Methods for setting focus correctly when faces are detected are discussed. Issues when working with OpenGL ES 2.0 are described, and finally future work in the area is proposed.

A framework for the target platform - capable of fetching images from the camera module, processing them and presenting them on the platform screen - was successfully implemented. Using a CPU reference implementation of the detector, this framework ran at a frame rate of approximately 1 - 2 frames per second without platform-specific optimization.

Jianxin Wu, S. et al., (2008) presented a new approach to design node classifiers in the cascade detector. There are three contributions in their paper. The first is a categorization of asymmetries in the learning goal, and why they make face detection hard. The second is the Forward Feature Selection (FFS) algorithm and a fast precomputing strategy for Adaboost. FFS and the fast Adaboost can reduce the training time by approximately 100 and 50 times, in comparison to a naive implementation of the Adaboost feature selection method. The last contribution is Linear Asymmetric Classifier (LAC), a classifier that explicitly handles the asymmetric learning goal as a well-defined constrained

optimization problem. They demonstrated experimentally that LAC results in improved ensemble classifier performance.

Moutarde et al., (2008) proposed a method that deals with real-time visual detection, by mono-camera, of object categories such as cars and pedestrians. They report on improvements that can be obtained for this task, they also presented a new feature type, called “connected control-points”, for Adaboost training of visual object classifiers . They used new visual features as Adaboost weak classifiers, while still operated fast enough for real-time pedestrians and vehicles detection. They reported here on a test of these new features on two publicly available databases: one for lateral cars, and one for pedestrians on which many classification algorithms have already been tested and results published. It turns out that the Adaboost strong classifiers obtained with their new features, while being extremely fast (~0.4ms per pedestrian image classification on a 2Ghz laptop), clearly outperform both standard Viola-Jones boosted cascade and even the most powerful (but very slow) classification algorithms reported so far on the pedestrian database.

Cho, et al., (2009) presented hardware architecture for face detection based system on Adaboost algorithm using Haar features. They describe the hardware design techniques including image scaling, integral image generation, pipelined processing as well as classifier, and parallel processing multiple classifiers to accelerate the processing speed of the face detection system. Also, they discussed the optimization of the proposed architecture which can be scalable for configurable devices with variable resources. The

proposed architecture for face detection has been designed using Verilog HDL and implemented in Xilinx Virtex-5 FPGA. Its performance has been measured and compared with an equivalent software implementation. They show about 35 times increase of the system performance over the equivalent software implementation

Shihavuddin et al., (2010) described a face detection framework that is capable of processing input images pretty swiftly while achieving high detection rates. The existing method for face detection can be divided into image based methods and feature based methods. The developed system is intermediary of these two, using a hybrid method comprising boosting algorithm and a hyper plane to train a classifier which is capable of processing images rapidly while having high detection rates. Using the response of simple Haar-based features used by Viola and Jones, Adaboost algorithm and an additional hyper plane classifier, the presented face detection system is developed.

This system is further modified by some intuitive noble heuristics. A set of experiments in the domain of face detection is presented. The system yields face detection performance comparable to the best previous systems.

Chapter 3

Face Detection System Using Haar Features and Adaboost Algorithm

3.1 Introduction

The Viola & Jones approach achieved great results in the face detection field. This thesis uses 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 from their approach in their paper, 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, and later will be compared.

3.2 The 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 thesis 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 the systems in this thesis.

Figure 3.1 describes the architecture of the system that will be used as face detection. It will be based on the Haar features and the Adaboost algorithm in general and each stage will have a detailed description:

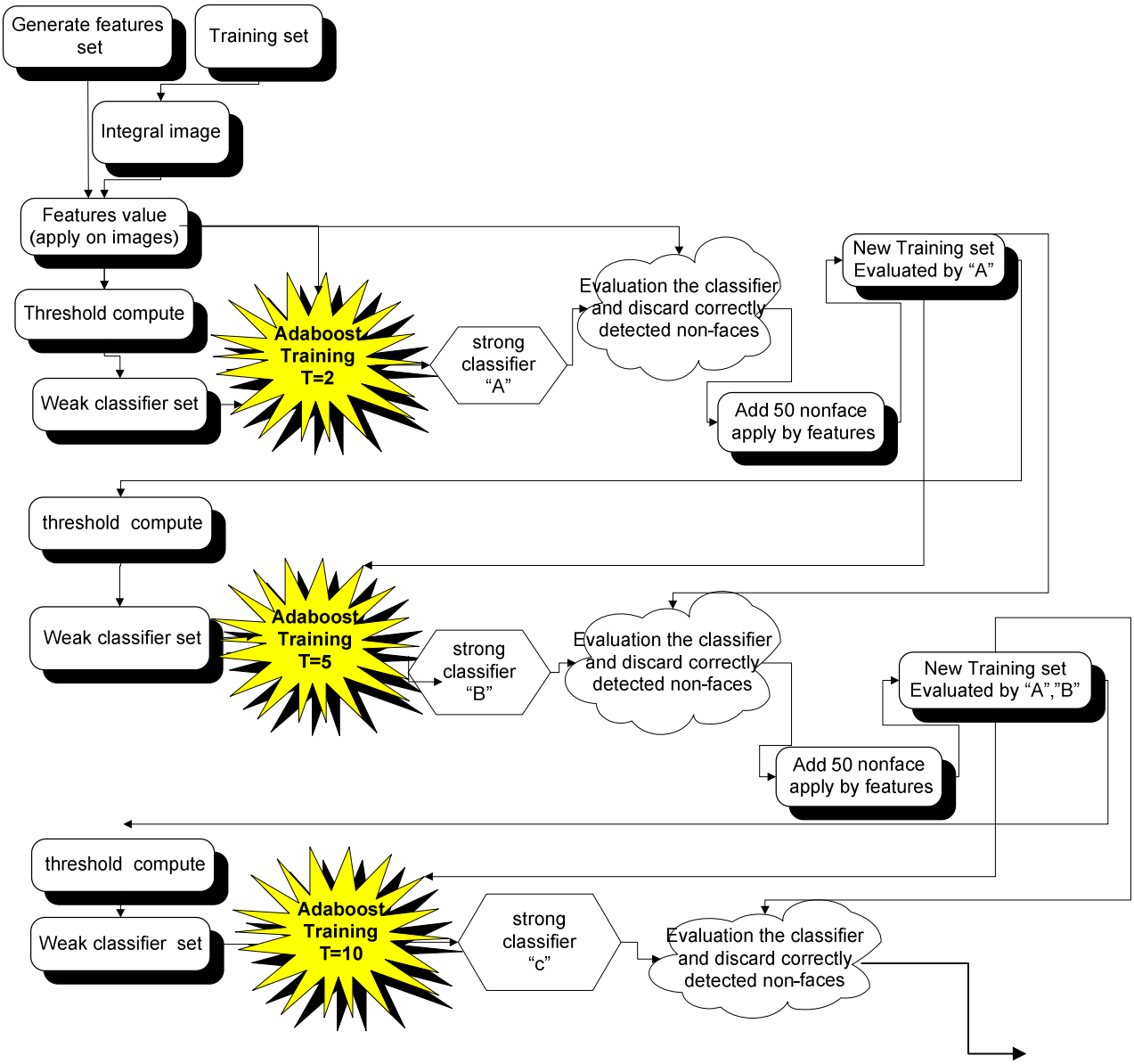


Figure 3.1: The Cascade Training Process for three stages.

- **Generate Features Set**

For a subwindow size 24×24 there are a number of Haar like feature sets that could be generated from each feature that's used in the system. The features are saved in the array to be used later. The numbers of the features generated differ from a system to another depending on the number of features that are used as we will see later in table 3.2 .

- **Training Set**

The training set will contain two different groups. A group that consists of positive (face) images and the other group consists of negative (non-face) images. The images that will be used are taken from two different sources, the IMM (Informatics and Mathematical Modelling) Face Database and images added by the author of this thesis.

- **Integral Image**

The integral image computes a value at each pixel (x,y). This value is computed by summing the values of the pixels located above and to the left of the pixel. The rectangle Haar-like features can be computed rapidly using “Integral Image”.

- **Features Value (Apply On Images)**

For each image in the system, all of the features that are generated in different sizes are computed. The computed value will be saved to be used later in other operations.

- **Threshold Value**

For each feature in the system, the threshold value are computed using one of the algorithms which are used in the comparison (average of means, the optimal threshold).

- **Weak Classifier Set**

Simple classifiers built using the Haar like features are generated based on the threshold value and the Haar features value. A set of weak classifier can be also generated based on

the single feature value. This could be considered the first step in the Adaboost training. As shown in figure 3.1, the threshold will be computed in each stage; therefore the weak classifier will also be computed in each stage too.

- **Adaboost Training**

The Adaboost algorithm allows combining a set of weak classifiers to a more precise resulting classifier. This classifier aims to build a strong classifier by applying the algorithm of the Adaboost. The algorithm uses the weak classifiers of the system as an input, and outputs a strong classifier for this system. The number of weak classifiers that could be used is 2, 5 and 10.

- **Strong Classifier**

The strong classifier is a classifier that contains a combination of weak classifiers that are generated by the Adaboost algorithm. The system could generate 3 different stages of the strong classifiers.

- The first stage consists of 2 weak classifiers.
- The second stage consists of 5 weak classifiers.
- The third stage consists of 10 weak classifiers.

- **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 images that

aren't correctly detected in the training set are discarded, decreasing the number of images. Then 50 new non-face images are added and used to build a new training set.

3.3 Generate Features Set

As mentioned before, the generation of Haar like features depends on the subwindow size to locate the number of the features generated. The features which are generated will be in different sizes and locations as shown in figure 3.2. In this thesis the subwindow size used is (24×24).

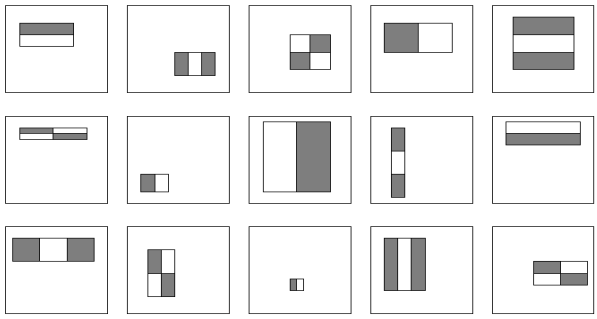


Figure 3.2: Examples of Haar-like features in different sizes and

To find the number of the features that could be obtained for any subwindow in any size, we can use equation (3.1). The number of features derived from each prototype is quite large and differs from prototype to another . This number could be calculated for any prototype by equation 3.1. 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 windowsize H is the high of windowsize and w is the width of feature rectangle h is the high of feature rectangle . An upright feature of size *w*x*h* then generates:

$$XY \cdot \left(W+1-w\frac{X+1}{2}\right) \cdot \left(H+1-h\frac{Y+1}{2}\right) \quad \ldots\ldots(3.1)$$

Equation 3.1 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 is a point that needs to be discussed before reading the table of features, which is related to a number of features in other papers, like in Lienhart & Maydt (2002), and Reddy & Bhuyan (2008). In table 3.1 which explains the total number of raw features computed within a subwindow of size 24×24, when equation 3.1 used to compute the number of features 2b & 2d, it will result in (19800) features instead of (20,736) features. The last number belongs to the Diagonal line feature 4 when equation 3.1 is applied on this feature. This feature is not mentioned in the table, and it is used in both papers of Viola & Jones (2001) and Viola & Jones (2001). Therefore the total number must be changed as shown in table 3.1.

TABLE I
TOTAL NUMBER OF RAW FEATURES COMPUTED WITHIN A SUB-WINDOW OF SIZE 24 X 24.

Feature Type	w/h	X/Y	Number of Features
$1a : 1b$	2/1 : 1/2	12/24: 24/12	43,200
$1c : 1d$	2/1 : 1/2	8/8	8,464
$2a : 2c$	3/1 : 1/3	8/24: 24/8	27,600
$2b : 2d$	4/1 : 1/4	6/24: 24/6	20,736
$2e : 2g$	3/1 : 1/3	6/6	4,356
$2f : 2h$	4/1 : 1/4	4/4	3,600
$3a$	3/3	8/8	8,464
$3b$	3/3	3/3	1,521
Total			117,941

19,800

117,005

Table 3.1: Total number of raw features computed within a subwondow of size 24x24, Lienhart & Maydt (2002), and how it must be corrected.

There are several types of features used in this thesis. Figure 3.3 shows the feature types that will be used in all systems. Table 3.2 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.

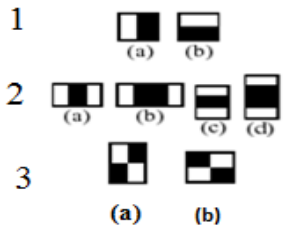


Figure 3.3: The Haar features that are used in this thesis.

The system	Features number	Feature types used in	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

Table 3.2: Lists the feature numbers and the type used in each system

3.4 Training Set (The dataset)

Viola & Jones approach deals with gray scale images. In this system 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:

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.

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 dataset1 manually.

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)

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 figure 3.4 explains the four steps which were discussed before.

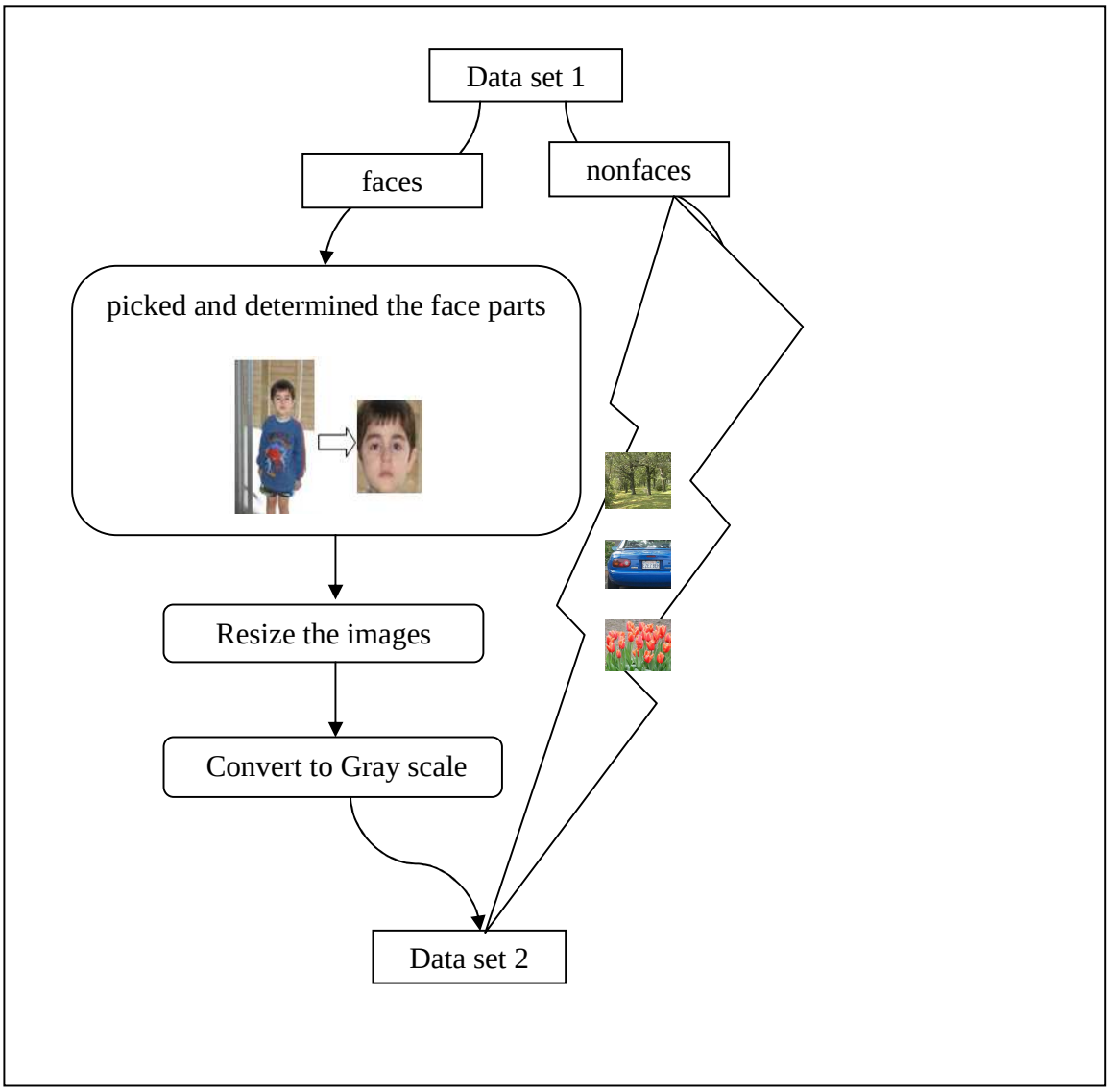


Figure 3.4: The training set building.

The dataset build contains (250) images, 150 images as face image, and 100 images as nonface images.

3.5 Integral Image (Fast Feature Evaluation)

Integral Image seems similar to the "Summed Area Table (SAT)" idea which is used in computer graphics for texture mapping. The rectangle Haar-like features can be computed rapidly by using this approach, and the new image will be generated so that the

value of each pixel in the new image will contain the value of the pixel above and the pixel to the left of the original pixel in the old image.

Using eqution 3.2, the value of the integral image will be computed for all the images introduced into the system by an integral image function:

$$P(x, y) = \sum_{x' \le x, y' \le y} i(x', y') \quad \text{.....3.2}$$

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 figures 3.5 and 3.6**.**

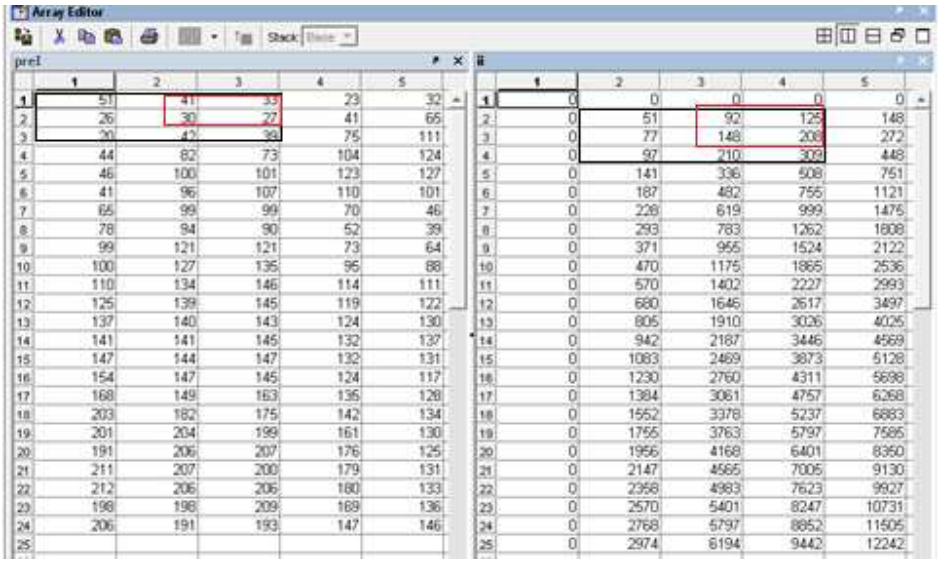


Figure 3.5: The implementation of the pad and the integral image function as an array

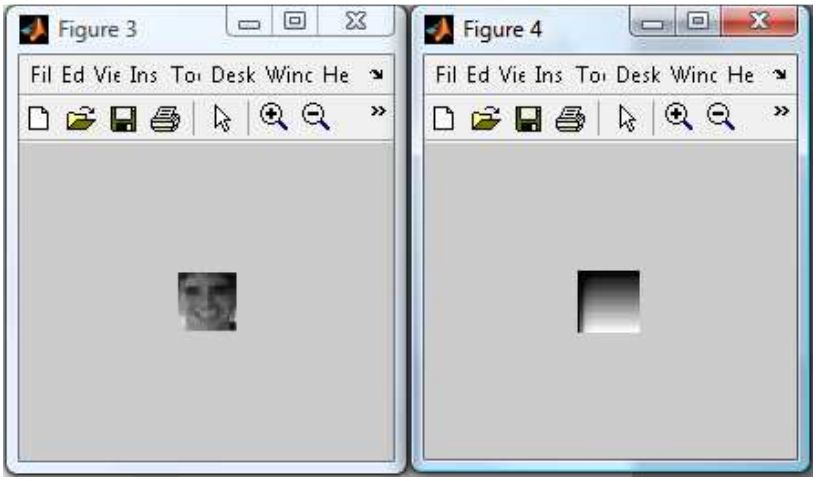


Figure 3.6: The implementation of the pad and the integral image function in an image

To explain the idea of the integral image let's say that there are 2 rectangles 3*3 in figure 3.5, one represents part of original image and the other represent parts of the integral 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.

3.6 Features Apply On Images (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 in this thesis to extract the features of every image in the training set.

The purpose of this section is 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.

3.7 Threshold Computing

Before talking about the machine learning approach used in this thesis (Adaboost algorithm), there is an important term that must be mentioned which is the threshold value. It is the value that separates the face from the non-face images to compute the weak classifier. The weak classifier is the input to the Adaboost algorithm; therefore, the threshold is important because it is the base of the weak classifier build.

There are more than one way to compute the threshold value as seen in chapter two.This thesis 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 in this thesis as follows:

- The average of their means by using equation 3.3.

$$threshold = (\mu i+ + \mu i-)/2 \qquad \text{…….(3.3)}$$

  where μi+ mean of faces, μi− mean of non-faces.

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

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 3.4.

$$threshold(t) = Hfaces(t) - Hnonfaces(t) \qquad \text{…….(3.4)}$$

Based on this information about threshold algorithms and the features number in this thesis, there are 12 systems used in this thesis. 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 3.2. 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.

3.8 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

   a- Feature (f_i),

   b- Threshold ($\theta_{i)}$

   c- 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 f_i is evaluated at x, and the output

of the weak classifier $h_i(x)$ is either -1 or 1. The output depends on whether the feature value is less than a given threshold θ_i in equation 3.5.

$$h_i(x) = \begin{cases} 1 & \text{if } p_i f_i(x) < p_i \theta_i \\ -1 & otherwise \end{cases} \quad \ldots\ldots\ldots\ldots(3.5)$$

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.

This is also known as a weak learner. The learner is called weak because even the best classification function is not expected to classify the training data well (i.e. for a given problem the best perceptron may only classify the training data correctly 51% of the time).

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.

3.9 Adaboost training (Adaboost algorithm)

Boosting is an efficient classifier generating algorithm, used to convert a weak classifier into a strong one, by combining a collection of weak classifiers to form strong classifier.

The Adaboost learning algorithm is used to boost the classification performance of a simple learning algorithm (e.g., it might be used to boost the performance of a simple perceptron).

The hypothesis of Viola and Jones, which is born out of an experiment, hypothesized that a very small number of these features can be combined to form an effective classifier. The main challenge however, is to find these features. To find these features, they suggest using a variant of Adaboost that is used to select the features and to train the classifier. This suggested that the system uses the original Adaboost algorithm, which utilizes a combination of simple weak classification functions to build a strong classifier. The main idea behind Adaboost is to boost the performance of a simple weak learning algorithm.

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.

The Training Algorithm:

In this thesis the Adaboost algorithm that is shown in figure 2.10 is used for boosting part. 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 are non-face, where the Y is label of 1,-1 for face , non-face.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_{i=1,-1}$ respectively, where m and l are the numbers of negatives (non-face) and positives (face) respectively in this thesis m=100, l=150.

❖ For t=1……..T, in this thesis T=3 in each system

1. Normalize the weights, by using equation 3.6.

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \quad \text{…….(3.6)}$$

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 ∈ is evaluated with respect to w_t computed by equation 3.7.

$$\in j = \sum_i wi\,|hj(xi) - yi| \quad \text{…….(3.7)}$$

3. Choose the classifier h_t, with the lowest error$\in_t$

4. Update the weights, using equation 3.8.

$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i} \quad \text{…….(3.8)}$$

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

❖ The final strong classifier can be calculated using equation 3.9.

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases} \quad \text{…….(3.9)}$$

where $\alpha_t = \log\frac{1}{\beta t}$.

To explain how the Adaboost algorithm works Figure 3.7 describes the process of the Adaboost training.

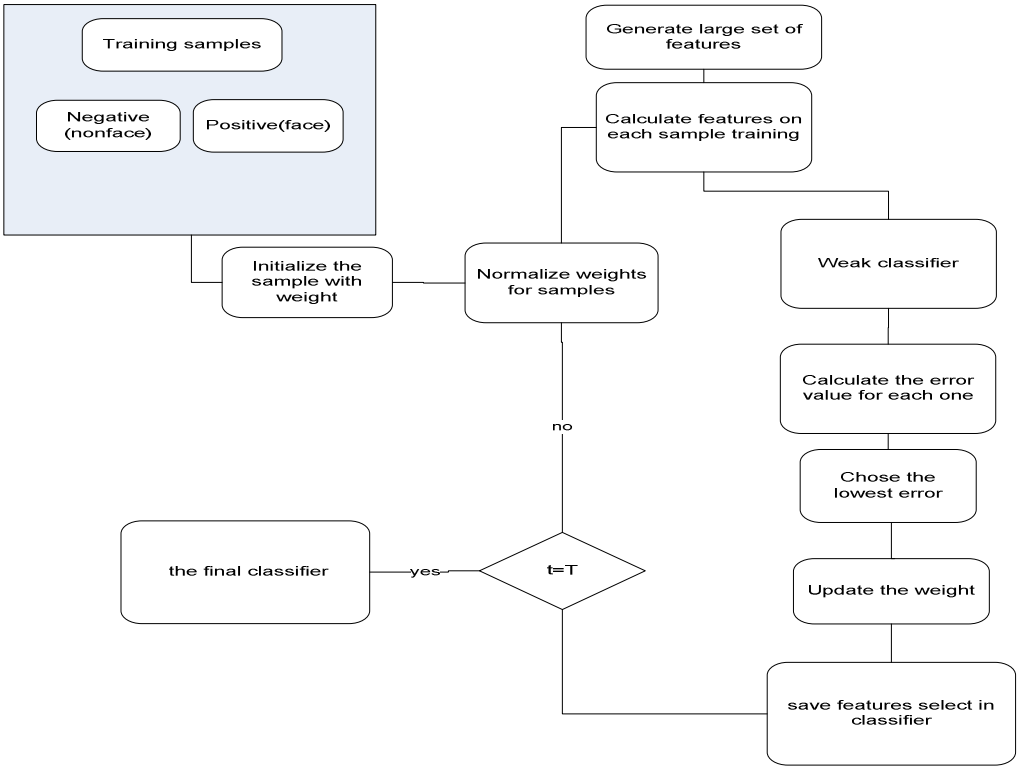


Figure 3.7: the Adaboost training diagram explains how the Adaboost algorithm works.

As shown in figure 3.7, a large set of Haar features are generated before the Adaboost training, to be used later in the training process. 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 choses 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.

3.10 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 Figure 3.8. 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.

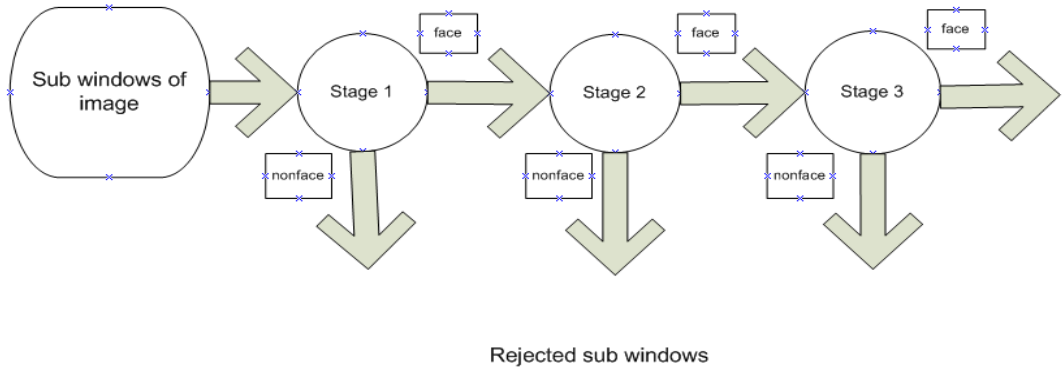


Figure 3.8: The Cascade Classifier structure.

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:

1- The total number of classifiers.

2- The number of features in each stage.

3- The threshold of each stage.

In this thesis, the cascade structure is based on the method used by Berggren & Gregersson (2008), but it differs in that this thesis consists of 3 stages, and each stage has different number of features. In stage one there are 2 features, in stage two there are 5 features, and in the final stage the number of features is 10. The first strong classifier was trained with the whole training set of faces and non-faces. The second strong classifier used a training set containing all faces, but only the non-faces that the previous strong classifier could not classify correctly. This way, the strong classifiers in the later steps of the cascade can focus on different non-faces than the previous steps. This means that the cascade consists of strong classifiers that are good at separating different sets of non-

faces from the faces. As an optimization of the algorithm, the thresholds of the weak classifiers can be retrained after updating the training set. This will remove the dependencies from discarding non-face examples.

3.11 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.

3.12 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 figure 3.9, which shows the detection procedure.

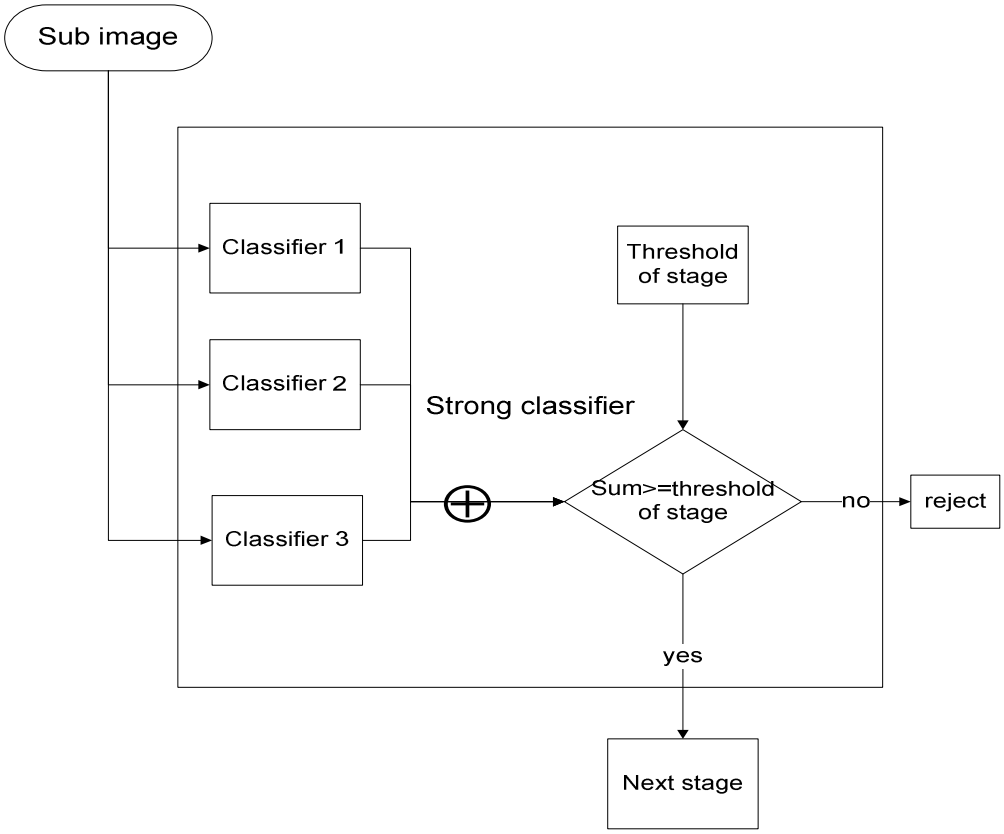


Figure 3.9: The detection procedure.

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 our thesis, the idea of resizing the image was used, which is shown in figure 3.10. 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.

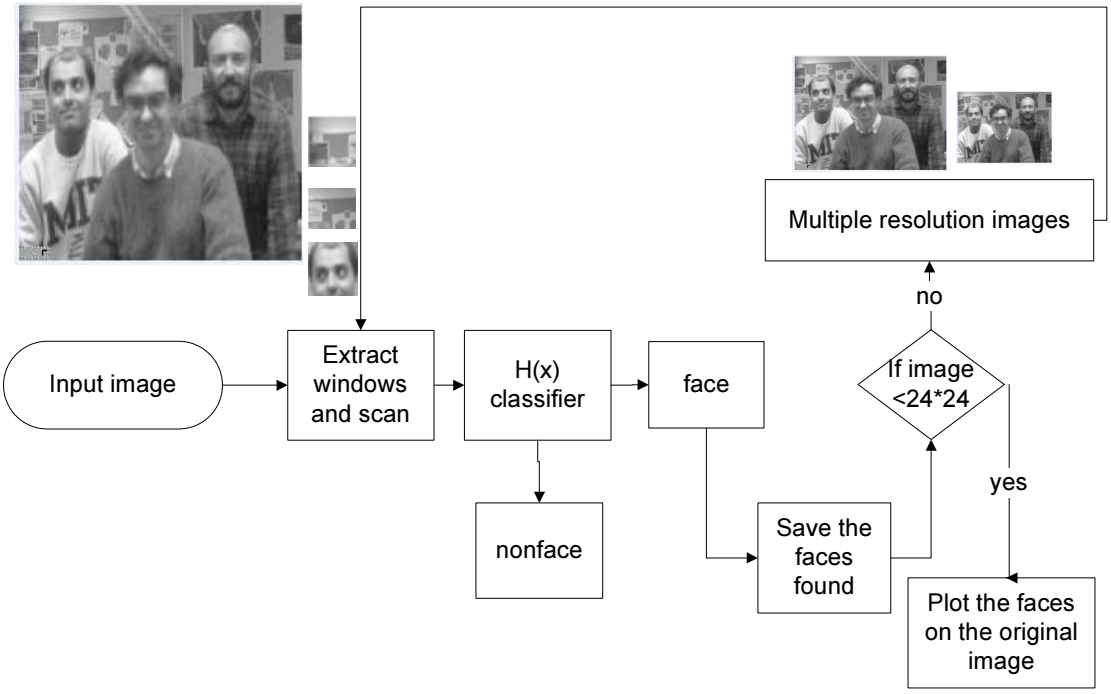


Figure 3.10: The block diagram of the face detector used in the system.

Chapter 4

Experimental Result and Discussion

4.1 Experimental Result

The face detection systems presented in this thesis was trained and tested using MATLAB 7.0 on Intel core (TM) i3 2.13 GHz 4GB of RAM and windows Vista ™ 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. Each system will be divided into two sections. The first is the training section and the second is the detection (testing) section**.**

4.1.1 Training

As mentioned before, the database used in training is not a famous database. It 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. In the figure 4.1 below, you can see some of the examples of the images that were used in the database. All of the images were scaled to the size of the subwindow, which is used in the systems (24×24).

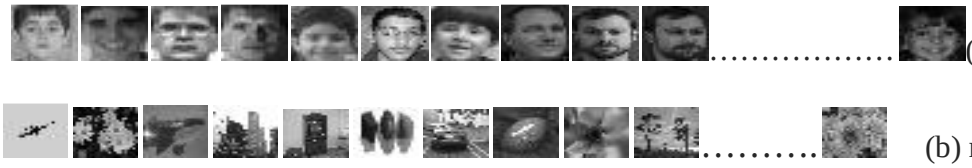


Figure 4.1: Examples of images that were used in the training.

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 3.2. 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. It takes about 7-12 hours to apply the features on all of the images, depending on the system.

After computing the threshold values, using the two methods, the weak classifiers for each system before the training using the Adaboost will give the rates for every system as shown in the tables 4.1 ,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.

The rate1	Tp	Fn	Tn	Fp
System1	0.58183	0.41817	0.56404	0.43596
System2	0.55833	0.44167	0.54831	0.45169
System3	0.57066	0.42934	0.55429	0.44571
System4	0.56779	0.43221	0.35864	0.64136
System5	0.56233	0.43767	0.54923	0.45077
System6	0.56332	0.43668	0.55027	0.44973

Table 4.1: The system based on the threshold 1 (average of means) before training

The rate2	Tp	Fn	Tn	Fp
System1	0.89072	0.10928	0.32724	0.67276
System2	0.90738	0.092618	0.2838	0.7162
System3	0.89873	0.10127	0.30207	0.69793
System4	0.86361	0.13639	0.16852	0.83148
System5	0.90462	0.095382	0.28182	0.71818
System6	0.90377	0.096228	0.28454	0.71546

Table 4.2: The system based on the threshold 2 (optimal threshold) before training

From these results, system 1 in 4.1 the Tp is higher than the other systems. While in 4.2 system 2 the Tp is highest, even though the numbers of the features in these systems is low. These results have not gone through the Adaboost training algorithm yet.

The second method to compute the threshold is better than the first method as shown in Tables 4.1 and 4.2. The first system in table 1 is better, while in the second method the second system is better than the systems compared. These results will change after the Adaboost algorithm training, the system will be:

The rate1	Tp	Fn	Tn	Fp
System1	0.72667	0.27333	1	0
System2	0.72	0.28	1	0
System3	0.72	0.28	1	0
System4	0.94667	0.053333	0.90476	0.095238
System5	0.6533	0.34667	1	0
System6	0.70667	0.29333	1	0

Table 4.3: The system based on the threshold 1 (average of means) after training

The rate2	Tp	Fn	Tn	Fp
System1	0.81333	0.18667	1	0
System2	0.78	0.22	1	0
System3	0.78	0.22	1	0
System4	0.98667	0.013333	1	0
System5	0.71333	0.28667	1	0
System6	0.78667	0.21333	1	0

Table 4.4: The system based on the threshold 2 (optimal threshold) after training

From Table 4.3 we could observe that system 4 has a higher Tp rate, but it still has Fp rate unlike other systems. System 1 had the best results when Fp rate was considered. On the other hand, from Table 4.4 we could observe that system 4 had the best results. These results were obtained from the he training section are initial results, and it might change during the next section.

As seen in the training results, all the systems that were built based on the second method of threshold give a better performance than the first method, but it takes a long of time to

compute the results. This makes the training of Adaboost of these systems slower than the others. Tables 4.3 and 4.4 are the final result of the training.

4.1.2 Testing

In this section all systems will be run on the MIT +CUM database and compare between them. One image from this database is shown in Figure 4.2. The image consists of 25 faces as shown below, and the result of each detection is shown on it. Table 4.5 will display the detection window, false positive and true negative.

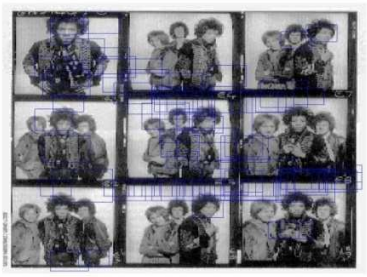
4a-1

4a-2

4b-1

4b-2

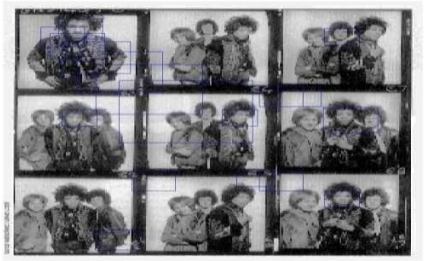
5-1



5-2

6-1

6-2



7-1

7-2

8-1

8-2

Figure 4.2: The implantation of the classifiers on an image using all systems.

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

Table 4.5: The result of detector on the test image.

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

The detection rate for this image is 25% in most systems. That’s because the simple detector that was built, the average time for detection each image is 3 to 4 minutes. Therefore all the training database will take approximately 7 hours.

In general, this implementation on MIT +CUM of these systems doesn't achieve a good tp, fp. In 8-1, the tp is 235 and the fp is 1240, in 8-2 the tp 414, fp is 3201.

System	Detection rate	System	Detection rate
4a-1	29%	4a-2	54%
4b-1	39%	4b-2	35%
5-1	42%	5-2	29%
6-1	21%	6-2	03%
7-1	31%	7-2	36%
8-1	27%	8-2	45%

Table 4.6 the detection rate of the systems

Table 4.6 shows that the system that was built using the second threshold has better results in general, except in the case of 6-2, because the features that were used in this detector were not good. Comparing between the features in different systems, the features that were used 5-1 were better in first group, while in the second group systems 4a-1, 8-2 also provided good results.

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 in the first systems. These results are not very accurate because the detector consists of 3 layers and the third layer only has 10 features.

4.2 Discussion

The detector that was used in this thesis was a simple detector; it was used to gather statistical data when comparing between features and the two methods to compute the threshold. The detector used is a it was created to achieve the objective of this thesis.

As seen in the tables in section 4.1, 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 equation 3.2 to calculate the threshold, the results were better if Viola & Jones system, that included, feature 5 was used. On the other hand, if equation 3.3 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, but instead used (1a, 1b, 2a, 2b, 2c) as shown in (2007).

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 thesis the highest number was 10.

According to the tables in 4.1 the threshold values were better for systems that used equation 3.3 in the training stages. On the other hand during the testing stages that were done on the MIT+CUM databases, and according to table 4.6, the data show that the results varied between the systems, but in general it slightly better for systems used equation 3.3. The difference between the two is the time required to calculate the

threshold. The time required to calculate a single value using equation 3.3 is approximately 0.32ms, while the time required for equation 3.2 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 nonface 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.

There are many researches that used Viola & Jones method. In case of comparing the different researches that emerged in this field, there are many of them. Therefore we chose a small number of the researches that were most related to our topic, to compare with our thesis.

Classifier	Detection rate	Threshold	Features number/Classifier	Database use
Viola & Jones(2001)	93.7%	Doesn't mention	4297 features	MIT+CUM
ZALHAN BIN MOHD ZIN(2007)	94.25%	Mutli threshold		BioID
Grafulla & Gregersson	96.71%	Optimal threshold	6 strong classifiers	Cambridge AT&T Laboratories
Used system	54%	Optimal threshold	15	MIT+CUM

Table 4.7: Results compared to other system.

In 2001, Viola & Jones created a detector is with a 32 layer cascade of classifiers which includes a total of 4297 features. They ran tests on the MIT+CUM database using that detector and the detection rate was 93.7% and their false positives were 422. Their face

detector can process a 384 by 288 pixel image in about .067 seconds (using a starting scale of 1.25 and a step size of 1.5 described below). This is roughly 15 times faster than the Rowley-Baluja-Kanade detector, though they didn't mention how their threshold was calculated in their paper.

Zin Mohd(2007), achieved a detection rate of 94.25%. The tests were run on the BioID database, which contained 1,00 images containing faces of different people, gender, expression, size, location, level of illumination and appearance of non-face objects.. He used gentle Adaboost, and multi threshold to calculate the threshold. Seven new feature types were proposed in order to increase the quality of feature selections. As a consequence, the feature solution sets and the search space become bigger and genetic algorithm has been used to overcome the problem of feature.

Grafulla & Gregersson (2008) used face detection based on the Viola & Jones approach. They built a system that consisted of 6 strong classifiers. They achieved 96.71% detection rate, by using optimal threshold. They also built 2 other systems, an optimization system and an extension cascade. They used Cambridge AT&T Laboratories database, this database contained 2429 face images and 4548 nonface images. All the images were 19x19 pixels in grayscale, stored in pgm format.

Finally, in the system used by this thesis, used 24x24 pixel images based on the Viola & Jones approach. The detection rate was 54%, the threshold used was the optimal threshold, which is the same method as Grafulla & Gregersson. It consists of 3 strong classifiers. The test database was MIT+CUM. The face detector process time, was from 3 to 4 minutes.

Chapter 5

Conclusion and Future work

5.1 Conclusion

In this thesis 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.

The system used a custom database that was generated manually; it tried to include everything in terms of face details like glasses, scarf on the head, beards, mustaches, different face colors, different Illumination, and anything that could help build a strong database. The databases contained 350 face and nonface images.

The system used the Adaboost algorithm to build the detector, it consists of 3 stages of strong classifiers; the first consisted of 2 weak classifiers, the second consisted of 5 weak classifiers, and the last strong classifier consisted of 10 weak classifiers to be built. Each weak classifier is computed based on the threshold before entering the Adaboost algorithm.

For all of the systems that were implemented in this thesis, the researcher has shown that the second method has improved computing the threshold compared to the first method.

This means that it would provide better and more accurate results. On the other hand, the first method showed that it takes less time to generate the results. Therefore , if speed and less computation is needed, and having less accuracy is acceptable, it would be advised to use the first method, though accuracy could be increased by increasing the number of iterations used in the Adaboost algorithm.

The total number of raw Haar features computed with a subwindow of size 24x24 in table 3.1 gave different results than the results that were observed; therefore the data in the table appears to be incorrect according to this thesis' results. The four non-basic features are better than the basic features based on the results that was computed in table 4.4. Furthermore, it was observed that feature 3 does not provide better results. And we'd advise not to use it in the future, and instead use one of the basic features in Viola & Jones.

5.2 Future work

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). Furthermore 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.

References

Abaza, A. Hebert, C. Harrison, M.A.F. ," Fast Learning Ear Detection For Real-Time Surveillance", West Virginia High Technol. Consortium Found., Fairmont, WV, USA Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on , 27-29 Sept. 2010.

Athitsos v. And Sclaroff S. , "Boosting Nearest Neighbor Classifiers For Multiclass Recognition" To Appear In Proceedings Of Ieee Workshop On Learning In Computer Vision And Pattern Recognition, June 2005.

Berggren Karl , Gregersson Pär ,"Camera focus controlled by face detection on GPU", Master Thesis Lund UniversityEricsson Mobile Platforms October 16, 2008.

Chen Longbin , Zhang Lei , Zhu Long , Li Mingjing A Novel Facial Feature Localization Method Using Probabilistic-like Output" This work was performed at Microsoft Research, Asia 2004.

Chin T.-J. and Suter D. "A New Boostrapping Strategy For The Adaboost-Based Face Detector", Department of Electrical and Computer Systems Engineering Technical Report ,MECSE-13-2005.

Cho Junguk, Mirzaei Shahnam , Oberg Jason, Kastner Ryan," FPGA-Based Face Detection System Using Haar Classifiers", ***FPGA***'09, February 22-24, 2009, Monterey, California, USA. Copyright 2009 ACM 978-1-60558-410-2/09/02.

Culp M., Johnson K., Michailidis G. , "Ada: An R Package For Stochastic Boosting" ***Journal Of Statistical Software*** October 2006, Volume 17, Issue 2.

Deng Y.& Su Guangda "Face Detection Based on Fuzzy Cascade Classifier with Scale-invariant Features" . ***International journal of information technology*** VOLUME 12, NUMBER 05, 2006.

Jones J . Michael. Viola Paul "Fast Multi-view Face Detection" IEEE conference on computer vision and pattern recognition, June 15, 2003.

Kailash J. Karande ,Sanjay N.Talbar. "Face Recognition Under Variation Of Pose And Illumination Using Independent Component Analysis" ***Icgst-Gvip, Issn 1687-398x***, Volume (8), Issue (Iv), December 2008.

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.

Lei Zhang, Stan Z. Li, Et.Al. "Boosting Local Feature Based Classifiers For Face Recognition". First Ieee Workshop On Face Processing In Video. 2004, Washington, Usa.

Lienhart, R. ; Kuranov, A. ; Pisarevsky, V.:" Empirical Analysis Of Detection Cascades Of Boosted Classifiers For Rapid Object Detection." In: Dagm 25th Pattern Recognition Symposium, 2003.

Lienhart Rainer and Maydt Jochen,"An Extended Set of Haar-like Features for Rapid Object Detection", ***IEEE ICIP*** *2002*, Vol. 1, pp. 900-903, Sep. 2002.

Le Duy-Dinh, Satoh Shin'ichi, "Feature Selection By Adaboost For Efficient Svm-Based Face Detection", ***Information Technology Letters***, Vol.3, Pp. 183-186, Sep 2004.

Menezes, P., Barreto, J. C. And Dias, J." Face Tracking Based On Haar-Like Features And Eigenfaces". 5thifac Symposium On Intelligent Autonomous Vehicles, Lisbon, Portugal, July 5--7, 2004.

Mita T., KanekoT., and Hori O.," Joint Haar-like features for face detection." In ICCV, 2005.

Meynet Julien, Arsan Taner, Mota Javier Cruz and Thiran Jean-Philippe, " Fast Multi view Face Tracking with Pose Estimation", Technical report TR-ITS.2007.01 January 26, 2007.

Mohd Zin Zalhan ," Enhanced Feature Selections Of Adaboost Training For Face Detection Using Genetic Algorithm" , A thesis submitted in fulfillment of the requirements for the award of the degree of Master of Engineering (Electrical) Faculty of Electrical Engineering Universiti Teknologi Malaysia , AUGUST 2007.

Monteiro G., Peixoto P., & Nunes U., “Vision-Based Pedestrian Detection Using Haar-Like Features”, Robotica, 2006.

Moutarde Fabien, Stanciulescu Bogdan and Breheret Amaury, " Real-time visual detection of vehicles and pedestrians with new efficient Adaboost features" June 10, 2008.

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.

Krishna. R Radha "Speeding Up Adaboost Object Detection With Motion Segmentation and Haar Feature Acceleration", A thesis submitted for the award of M.Eng. in Electronic Engineering at Dublin City University School of Electronic Engineering, August 19, 2009.

Rasolzadeh B., Petersson L. and Pettersson N., "Response Binning: Improved Weak Classifiers for Boosting", IEEE Intelligent Vehicles Symposium (IV2006), Tokyo, Japan, June 2006.

Reddy Chandan K. and Bhuyan Fahima A., "Retrieval and Ranking of Biomedical Images using Boosted Haar Features", In Proceedings of International Conference *on* BioInformatics and BioEngineering *(BIBE)*, Athens, Greece, October 2008.

Rowley, H., Baluja, S. & Kanade, T. “Neural Network-Based Face Detection,” ***IEEE-Transactions on Pattern Analysis and Machine Intelligence (PAMI),*** Vol. 20, No. 1, January, 1998.

Roy Anindya, Marcel Sébastien " Haar Local Binary Pattern Feature for Fast Illumination Invariant Face Detection ", British Machine Vision Conference, BMVC 2009, London, UK, September 7-10, 2009.

Ruta Andrzej, Li Yongmin, and Liu Xiaohui, "Pedestrian detector based on heterogenous features and the Adaboost framework",NiSIS Competition 2007 Analysis and Classification of the Daimler Chrysler Automotive Dataset Images.

Shihavuddin A.S.M , Mir Arefin Mohammad Nazmul, Mir Ambia Nahidul, Haque Shah Ahsanul and Ahammad Tanvir , "Development of real time Face detection system using Haar like features and Adaboost algorithm" , ***IJCSNS International Journal of Computer Science and Network Security***, VOL.10 No.1, January 2010.

Shirazi Hamed Masnadi, "Adaboost Face Detection", https://cseweb.ucsd.edu/ classes/fa04/cse252c/ projects/hamed.pdf.

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.

Viola P. and Jones M., "Rapid Object Detection Using A Boosted Cascade Of Simple Features." In ***IEEE CVPR*** 2001, 2001.

Whitehill J. And Omlin C. W., “Haar Features For Facs Au Recognition”, Proc. Of Intl Conf. Automatic Face And Gesture Recognition, 2006.

Wu J.x.; Brubaker S.C.; Mullin M.D. ; Rehg, J.M. **"Fast Asymmetric Learning For Cascade Face Detection**" Pattern Analysis And Machine Intelligence, ***Ieee Transactions***, Volume 30, Issue 3, March 2008 Page(S):369 – 382.

Zhao W. , Chellappa R., Phillips P. J. , And Rosenfeld A. ,"Face Recognition: A Literature Survey" ***Acm Computing Surveys***, Vol. 35, No. 4, December 2003, Pp. 399–458.

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.

Web Site : http://greta.cs.ioc.ee/~khoros2/one-oper/equalization/front-page.html.

Web Site: http://en.Wikipedia.Org/Wiki/Lookup_Table , This page was last modified on 23 Jan 2009.

web site http://www.codeproject.com/KB/recipes/ImgAlign2.aspx , This page was last modified on 21 Apr 2008.

web site http://en.wikipedia.org/wiki/Feature_extraction, This page was last modified on 30 December 2010 at 18:28.