Implementing the Viola-Jones Face Detection Algorithm

Ole Helvig Jensen

Kongens Lyngby 2008
IMM-M.Sc.-2008-93

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-M.Sc.: ISBN 87-643-0008-0
ISSN 1601-233X

Abstract

The emergence of high resolution digital cameras for recording of still images and video streams has had a significant impact on how communication and entertainment have developed during the recent years. At the same time Moore's law has made tremendous computing power readily available that only some 20 years ago was reserved for high profile research establishments and intelligence services. These two tendencies have respectively called for and fostered the advent of unprecedented computationally heavy image processing algorithms. Algorithms that in turn have allowed for new processing of existing image based material.

Parallel to this technological development the measures deployed in the protection against attacks from the enemies of modernity calls for more surveillance of the public space. As a result of this regrettable circumstance more video cameras are installed in airports, on stations and even on open streets in major cities.

Whether or not the purpose is entertainment or dead serious surveillance, tasks like detection and recognition of faces are solved using the same methods. Due to the varying and generally adverse conditions under which images are recorded there is a call for algorithms capable of working in an unconstrained environment.

In 2004 an article by Paul Viola and Michael J. Jones titled "Robust Real-Time Face Detection" was publish in the International Journal of Computer Vision. The algorithm presented in this article has been so successful that today it is very close to being the de facto standard for solving face detection tasks. This success is mainly attributed to the relative simplicity, the fast execution and the remarkable performance of the algorithm.

This report documents all relevant aspects of the implementation of the Viola-Jones face detection algorithm. The intended input for the face detection algorithm is any conceivable image containing faces and the output is a list of face positions.

Resumé

Fremkomsten af digitalkameraer med høj opløsning til optagelse af både billeder og video har betydeligt ændret udviklingen af kommunikation og underholdning inde for de seneste par år. Samtidig har Moores lov stillet en enorm computerkraft til rådighed, der for kun 20 år siden var forbeholdt højt profileret forskningsinstitutioner og efterretningstjenester. Disse to tendenser har henholdsvis efterspurgt og tilladt udviklingen af billedbehandlingsalgoritmer med hidtidig uset kompleksitet. Disse algoritmer har efterfølgende muliggjort ny behandling af eksisterende billedmateriale.

Parallelt med denne udvikling kræver beskyttelsen imod angreb fra modstandere af moderniteten en stadig større overvågning af det offentlige rum. Som en konsekvens af denne beklagelige omstændighed installeres flere overvågningskameraer i lufthavne, på togstationer og sågar også på åbne gader i større byer.

Uanset om formålet er underholdning eller gravalvorlig overvågning, så kræves de samme løsninger til opgaver såsom detektion og genkendelse af ansigter. Pga. de varierede forhold som disse løsninger skal arbejde i, er der et stort behov for robuste algoritmer.

I 2004 publicerede Paul Viola og Michael J. Jones artiklen ”Robust Real-Time Face Detection” i tidsskriftet International Journal of Computer Vision. Algoritmen, der præsenteres i denne artikel har været så succesfuld, at den i dag kan betragtes som værende de facto standard når ansigter skal findes i billeder. Algoritmens succes skyldes hovedsageligt dens forholdsvise simple udformning, dens hurtige afvikling og dens bemærkelsesværdige resultater.

Denne rapport dokumenter alle relevante aspekter vedrørende implementeringen af Viola-Jones’ ansigtsdetektionsalgoritme. Det er meningen at algoritmen skal kunne behandle ethvert tænkeligt billede, indeholdende ansigter, og som resultat lave en liste med registrerede ansigter.

Preface

This project was originally born of a co-operation between Denmark's national broadcasting corporation, DR, and the Department of Informatics and Mathematical Modelling, IMM, at the Technical University of Denmark, DTU. This co-operation was established as a part of the larger project called “Den Digitale Kulturarv” (English: the digital cultural heritage) that aims at making all material produced by DR accessible from the internet. In this context this project was supposed to be a proof of concept for an algorithm for detection, tracking and recognition of known TV personalities appearing in old broadcasts.

Unfortunately, due to the time consuming task of implementing and especially training the Viola-Jones face detection algorithm this project was reduced to focus only on this implementation. While this work may not leave much space for innovative thought it still requires proficiency in areas such as image processing, programming and the ability to understand complex algorithms. In many respects this project does not seems to differ from the everyday work by a professional engineer and consequently it is the hope of the author that the presented work will be sufficient to pass as a master of science.

Many thanks go out to my supervisor Associate Professor Rasmus Larsen at IMM·DTU for his kind help and counselling during the course of this project.

Kongens Lyngby, September 2008

Table of contents

Introduction

The purpose of this project is to implement and thereby recreate the face detection algorithm presented by Viola-Jones. This algorithm should be capable of functioning in an unconstrained environment meaning that it should detect all visible faces in any conceivable image. The ideal goal of any face detection algorithm is to perform on par with a human inspecting the same image, but this project will constrain itself to only match the figures posted by Viola-Jones.

In order to guarantee optimum performance of the developed algorithm the vast majority of images used for training, evaluation and testing are either found on the internet or taken from private collections. In other words these images are not created under controlled conditions and are therefore believed to be a trustworthy representation of the unconstrained environment in which the face detector is expected to work.

The first part of this report presents the problem analysis and a small survey of the most popular face detection algorithms. The second part describes the theory behind the Viola-Jones algorithm and documents the implementation process concurrent with a discussion of intermediate results. Lastly the report is rounded off with considerations regarding future work and a conclusion.

A DVD is made alongside this project. This DVD contains the facial databases used for training, the developed scripts and source code, the processed data and the results. The contents of the DVD can be seen in Appendix 2, page 34.

Problem analysis

The basic problem to be solved is to implement an algorithm for detection of faces in an image. At a first glance the task of face detection may not seem so overwhelming especially considering how easy it is solved by a human. However there is a stark contrast to how difficult it actually is to make a computer successfully solve this task.

In order to ease the task Viola-Jones limit themselves to full view frontal upright faces. That is, in order to be detected the entire face must point towards the camera and it should not be tilted to any side. This may compromise the requirement for being unconstrained a little bit, but considering that the detection algorithm most often will be succeeded by a recognition algorithm these demands seem quite reasonable.

A typical input image to a face detection algorithm is shown in Figure 1. This image has a relatively low contrast, contains many different kinds of textures and lastly it contains many faces. Since more or less all the faces are frontal upright it is the hope that they will be detected by the algorithm developed in this project.

Figure 1 - A typical input image.

Existing methods

During the last decade a number of promising face detection algorithms have been developed and published. Among these three stand out because they are often referred to when performance figures etc. are compared. This section briefly presents the outline and main points of each of these algorithms.

Robust Real-Time Objection Detection, 2001 [1]
By Paul Viola and Michael J. Jones.
This seems to be the first article where Viola-Jones present the coherent set of ideas that constitute the fundamentals of their face detection algorithm. This algorithm only finds frontal upright faces, but is in 2003 presented in a variant that also detects profile and rotated views [2]. The 'Methods' chapter will elaborate more on the basic version of this algorithm.

Neural Network-Based Face Detection, 1998 [3]
By Henry A. Rowley, Shumeet Baluja and Takeo Kanade.
An image pyramid is calculated in order to detect faces at multiple scales. A fixed size sub-window is moved through each image in the pyramid. The content of a sub-window is corrected for non-uniform lightning and subjected to histogram equalization. The processed content is fed to several parallel neural networks that carry out the actual face detection. The outputs are combined using logical AND, thus reducing the amount of false detections. In its first form this algorithm also only detects frontal upright faces.

A Statistical Method for 3D Object Detection Applied to Faces and Cars, 2000 [4]
By Henry Schneiderman and Takeo Kanade.
The basic mechanics of this algorithm is also to calculate an image pyramid and scan a fixed size sub-window through each layer of this pyramid. The content of the sub-window is subjected to a wavelet analysis and histograms are made for the different wavelet coefficients. These coefficients are fed to differently trained parallel detectors that are sensitive to various orientations of the object. The orientation of the object is determined by the detector that yields the highest output. Opposed to the basic Viola-Jones algorithm and the algorithm presented by Rowley et al. this algorithm also detects profile views.

One of the fundamental problems of automated object detection is that the size and position of a given object within an image is unknown. As two of the mentioned algorithms demonstrate the standard way to overcome this obstacle is to calculate an image pyramid and scan the detector through each image in the pyramid. While being relatively straightforward to implement this process is rather time consuming, but in their paper Viola-Jones presents a novel approach to this problem.

The Viola-Jones face detector

Introduction to chapter

This chapter describes the work carried out concerning the implementation of the Viola-Jones face detection algorithm. The first part elaborates on the methods and theory behind the algorithm. In order to avoid copying the original Viola-Jones paper this section is kept relatively short, but still the most important points are explained.
Secondly interesting aspects of the actual implementation are emphasized and presented together with results and comments on performance. This structure is preferred since many intermediate results have affected implementation decisions and vice versa.

Methods

The basic principle of the Viola-Jones algorithm is to scan a sub-window capable of detecting faces across a given input image. The standard image processing approach would be to rescale the input image to different sizes and then run the fixed size detector through these images. This approach turns out to be rather time consuming due to the calculation of the different size images.
Contrary to the standard approach Viola-Jones rescale the detector instead of the input image and run the detector many times through the image – each time with a different size. At first one might suspect both approaches to be equally time consuming, but Viola-Jones have devised a scale invariant detector that requires the same number of calculations whatever the size. This detector is constructed using a so-called integral image and some simple rectangular features reminiscent of Haar wavelets. The next section elaborates on this detector.

The scale invariant detector

The first step of the Viola-Jones face detection algorithm is to turn the input image into an integral image. This is done by making each pixel equal to the entire sum of all pixels above and to the left of the concerned pixel. This is demonstrated in Figure 2.

1	1	1
1	1	1
1	1	1

Input image

1	2	3
2	4	6
3	6	9

Integral image

Figure 2 – The integral image.

This allows for the calculation of the sum of all pixels inside any given rectangle using only four values. These values are the pixels in the integral image that coincide with the corners of the rectangle in the input image. This is demonstrated in Figure 3.

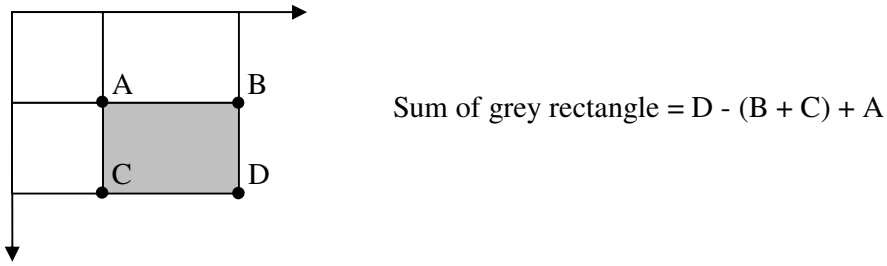


Figure 3 - Sum calculation.

Since both rectangle B and C include rectangle A the sum of A has to be added to the calculation.

It has now been demonstrated how the sum of pixels within rectangles of arbitrary size can be calculated in constant time. The Viola-Jones face detector analyzes a given sub-window using features consisting of two or more rectangles. The different types of features are shown in Figure 4.

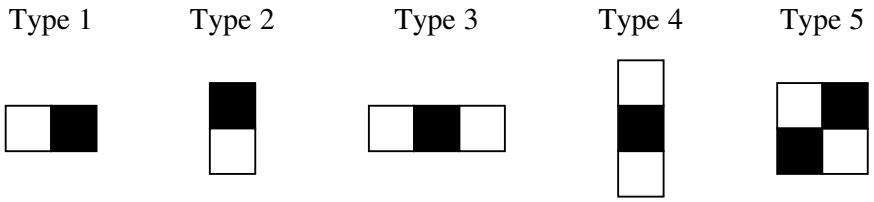


Figure 4 - The different types of features.

Each feature results in a single value which is calculated by subtracting the sum of the white rectangle(s) from the sum of the black rectangle(s).

Viola-Jones have empirically found that a detector with a base resolution of 24*24 pixels gives satisfactory results. When allowing for all possible sizes and positions of the features in Figure 4 a total of approximately 160.000 different features can then be constructed. Thus, the amount of possible features vastly outnumbers the 576 pixels contained in the detector at base resolution. These features may seem overly simple to perform such an advanced task as face detection, but what the features lack in complexity they most certainly have in computational efficiency.

One could understand the features as the computer's way of perceiving an input image. The hope being that some features will yield large values when on top of a face. Of course operations could also be carried out directly on the raw pixels, but the variation due to different pose and individual characteristics would be expected to hamper this approach. The goal is now to smartly construct a mesh of features capable of detecting faces and this is the topic of the next section.

The modified AdaBoost algorithm

As stated above there can be calculated approximately 160.000 feature values within a detector at base resolution. Among all these features some few are expected to give almost consistently high values when on top of a face. In order to find these features Viola-Jones use a modified version of the AdaBoost algorithm developed by Freund and Schapire in 1996 [5].

AdaBoost is a machine learning boosting algorithm capable of constructing a strong classifier through a weighted combination of weak classifiers. (A weak classifier classifies correctly in only a little bit more than half the cases.) To match this terminology to the presented theory each feature is considered to be a potential weak classifier. A weak classifier is mathematically described as:

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

Where x is a 24*24 pixel sub-window, f is the applied feature, p the polarity and θ the threshold that decides whether x should be classified as a positive (a face) or a negative (a non-face).
Since only a small amount of the possible 160.000 feature values are expected to be potential weak classifiers the AdaBoost algorithm is modified to select only the best features. Viola-Jones' modified AdaBoost algorithm is presented in pseudo code in Figure 5.

- Given examples images $(x_1,y_1),...,(x_n,y_n)$ where y_1=0,1 for negative and positive examples.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for y_1=0,1, where m and l are the numbers of positive and negative examples.
- For t=1,...,T:

1) Normalize the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

2) Select the best weak classifier with respect to the weighted error:

$$\varepsilon_t = \min_{f,p,\theta} \sum_i w_i \left| h(x_i, f, p, \theta) - y_i \right|$$

3) Define $h_t(x) = h(x, f_t, p_t, \theta_t)$ where f_t, p_t and θ_t are the minimizers of ε_t.

4) Update the weights:

$$w_{t+1,i} = w_{t,i} \beta^{1-e_i}$$

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

- The final strong classifier is:

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

where $\alpha_t = \log \frac{1}{\beta_t}$

Figure 5 - The modified AdaBoost algorithm.

An important part of the modified AdaBoost algorithm is the determination of the best feature, polarity and threshold. There seems to be no smart solution to this problem and Viola-Jones suggest a simple brute force method. This means that the determination of each new weak classifier involves evaluating each feature on all the training examples in order to find the best performing feature. This is expected to be the most time consuming part of the training procedure.

The best performing feature is chosen based on the weighted error it produces. This weighted error is a function of the weights belonging to the training examples. As seen in Figure 5 part 4) the weight of a correctly classified example is decreased and the weight of a misclassified example is kept constant. As a result it is more 'expensive' for the second feature (in the final classifier) to misclassify an example also misclassified by the first feature, than an example classified correctly. An alternative interpretation is that the second feature is forced to focus harder on the examples misclassified by the first. The point being that the weights are a vital part of the mechanics of the AdaBoost algorithm.

With the integral image, the computationally efficient features and the modified AdaBoost algorithm in place it seems like the face detector is ready for implementation, but Viola-Jones have one more ace up the sleeve.

The cascaded classifier

The basic principle of the Viola-Jones face detection algorithm is to scan the detector many times through the same image – each time with a new size. Even if an image should contain one or more faces it is obvious that an excessive large amount of the evaluated sub-windows would still be negatives (non-faces). This realization leads to a different formulation of the problem:

In stead of finding faces, the algorithm should discard non-faces.

The thought behind this statement is that it is faster to discard a non-face than to find a face. With this in mind a detector consisting of only one (strong) classifier suddenly seems inefficient since the evaluation time is constant no matter the input. Hence the need for a cascaded classifier arises.

The cascaded classifier is composed of stages each containing a strong classifier. The job of each stage is to determine whether a given sub-window is definitely not a face or maybe a face. When a sub-window is classified to be a non-face by a given stage it is immediately discarded. Conversely a sub-window classified as a maybe-face is passed on to the next stage in the cascade. It follows that the more stages a given sub-window passes, the higher the chance the sub-window actually contains a face. The concept is illustrated with two stages in Figure 6.

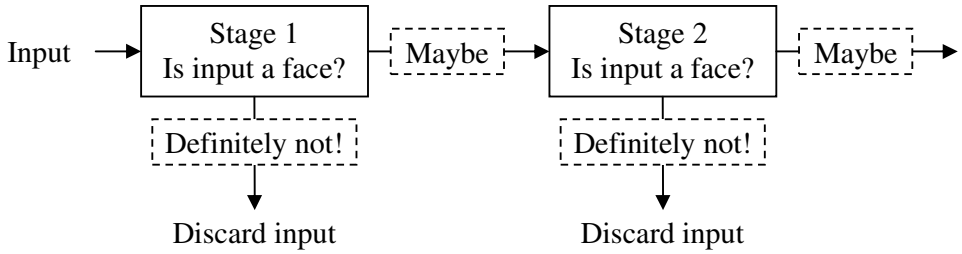


Figure 6 - The cascaded classifier.

In a single stage classifier one would normally accept false negatives in order to reduce the false positive rate. However, for the first stages in the staged classifier false positives are not considered to be a problem since the succeeding stages are expected to sort them out. Therefore Viola-Jones prescribe the acceptance of many false positives in the initial stages. Consequently the amount of false negatives in the final staged classifier is expected to be very small.

Viola-Jones also refer to the cascaded classifier as an attentional cascade. This name implies that more attention (computing power) is directed towards the regions of the image suspected to contain faces.

It follows that when training a given stage, say *n*, the negative examples should of course be falsenegatives generated by stage *n*-1.

The majority of thoughts presented in the 'Methods' section are taken from the original Viola-Jones paper [1].

Implementation & Results

This section presents some of the considerations regarding the implementation of the Viola-Jones face detector along with intermediate and final results. The algorithms and results are presented almost in the same order as they were developed and recorded.

Generating positive examples

In order to train the different stages of the cascaded classifier the AdaBoost algorithm requires to be fed with positive examples – that is, images of faces. The faces used in this project were taken from two different databases:

FERET The Facial Recognition Technology Database. A database administered and distributed by the American National Institute of Standards and Technology. Contains 2413 facial images taken under controlled conditions [6].

LFW Labeled Faces in the Wild. A database made and distributed by The University of Massachusetts with the intend of studying the problem of unconstrained face recognition. Contains more than 13.000 facial images found on the internet. All faces in the database were detected by a Viola-Jones face detector [7].

A Matlab script called “algo01.m” capable of conditioning the database images into positive examples was constructed. The script does the following:

1) Opens a facial image and transforms to greyscale if needed.
2) Displays the image and lets the user place a bounding box around the face.
3) Rescales the face to 24*24 pixels.
4) Saves the rescaled face as an image and as a variance normalized data file.

The variance normalization is suggested by Viola-Jones as a mean of reducing the effect of different lighting conditions. In Figure 7 an image from the LFW database is shown after the user has defined the face.

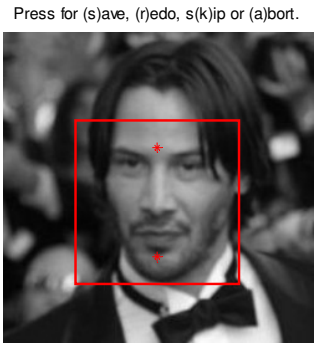


Figure 7 - Generating a positive example.

A total of 5000 positive examples should be sufficient to train the individual stages of the staged classifier, but these examples should be chosen with care. The final detector will only detect faces similar to those in the training set so the training set should represent the most common facial variation.

For the first stage in the cascade Viola-Jones advice constructing a simple pre-screening filter containing only two features. Since these two features are the first in the entire cascade they encode the most fundamental differences between the faces in the training set and the non-faces used for training. Consequently the two features also make it possible to draw conclusions regarding primarily the positive training set and secondly also the negative training set.

For training of the cascade a total of three slightly different data sets were constructed using the above-mentioned databases. Together with the same one negative data set the three data sets were used to train three different first stages. In the following the three data sets are presented together with a graphical representation of their respective two first features:

Data set A
Contains 5500 positive examples.
No. 1 – 1343 are taken from FERET.
No. 1344 – 5500 are taken from LFW.
From the FERET database only the frontal upright head shots were used.
The positive examples were made by having a user place a point just above the eyebrows and between the mouth and the chin. These two points were used to create a rectangle which was enlarged by 50 %. This is what is shown in Figure 7.
The first two features are shown in Figure 8.

Figure 8 - The first two features for data set A.

Data set B
Contains 5234 positive examples.
No. 1 – 1243 are taken from FERET.
No. 1244 – 5234 are the ‘good ones’ from LFW.
In this training set only frontal upright headshots from both databases were used.
The user placed a point between the eyes and a point just below the chin. The mark below the chin was the bottom line in a square box where the eyes were 1/3 from the top. The first two features are shown in Figure 9.

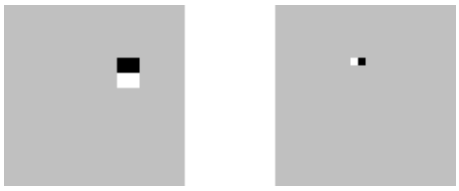


Figure 9 - The first two features for data set B.

Data set C
Contains 6000 positive examples.
No. 1 – 6000 are randomly taken from LFW.
The examples were generated as in data set A. The first two features are shown in Figure 10.

Figure 10 - The first two features for data set C.

The three figures shown above allow for some conclusions to be made. Most noticeable is that set A and C generate the same first feature. This feature represents the fact that the eye region is generally darker than the (light) upper cheek region on the training subjects. Also, the relatively large width of the feature is explained by variations in face sizes and proportions. This variation is suspected to stem primarily from the LFW images.
The size of the first feature in data set B is in stark contrast to size of the first feature in set A and C. A very probable reason is that set B is the most homogeneous set and apparently can be discriminated from the negative training set utilizing a small feature representing the left eye of the training subjects.

Looking at the second feature for data set A the conclusion is that it represents a contrast between the light bridge of the nose and the dark region of the training subjects' left eye.
For data set B two plausible explanations for the size and position of the second feature arise. The first observes that the position and type of the feature is similar to the second feature in data set A. Furthermore the smaller size should be attributed to the very homogeneous nature of the data set. That is, the position of the light bridge of the nose and the dark eye region are almost constant in the examples. The second explanation takes as a starting point that a feature at minimum size (1*2 pixels) doesn't encode much meaningful information. As a result of perhaps a very good discrimination by the first feature it should then be seen as a coincidence that the second features becomes so small.
Lastly the second feature of data set C represents the fact that there usually is a contrast between the training subjects' light forehead and the generally darker background.

For the training of the succeeding stages data set A is chosen. This data set seems to offer a good compromise between being too specific – like set B – and being too 'loose' – like set C. In addition data set A is also the only one where both features encode prominent facial characteristics. Of the 5500 positive examples contained in the data set 5000 are used for training and the last 500 as evaluation set.

The Matlab script that generated data set A can be found on the enclosed DVD.

Generating negative examples

For the training of the different stages in the cascade negative examples are also required. A negative example is basically just an image not containing a face. This criterion may at first seem easy to fulfil, but ideally the negative examples should represent all sorts of non-facial textures that the detector can be expected to meet.

The raw material for the generation of negative examples is images guaranteed not to contain faces. Some of the images used in this implementation are shown in Figure 11.

Figure 11 - Some of the images used for the generation of negative examples.

In order to make the most optimum training of the final detector the negative examples should be generated in the same manner as the detector scans through an image. This means that negative examples should be generated at every size and then rescaled to the base resolution of 24*24 pixels.

The Matlab script “algo02.mat” does the following:

1) Opens an image and transforms to greyscale if needed.
2) Runs a sub-window through the image.
2a) Rescales the content of the sub-window to 24*24 pixels (if needed) and saves it as a variance normalized data file.
2b) Enlarges the sub-window by a given factor a goes back to 2).
2c) Continues until the sub-window size is equal to the least dimension of the image.

As stated earlier the false positives from stage *n* should be used to train stage *n*+1. This means that part 2a) instead should be:

Rescale the content of the sub-window to 24*24 pixels (if needed) and save it as a variance normalized data file **if** it survives the existing stages of the cascade.

Thus, it is to be expected that the better the existing cascade becomes in discriminating between faces and non-faces the harder it will be to generate new negative training examples. This is very much confirmed by the numbers recorded during the training of the cascade. See Table 1.

Stage no.	Full resolution images used	Survived from previous stage	Sub-windows generated	Sub-windows discarded	Examples used for training
1	1	-	23560	0	23560
2	1	8989	23560	12389	20160
3	1	8380	92989	80742	20627
4	1	10146	207184	194318	23012
5	1	11194	207184	199341	19037
6	2	4325	1681710	1665630	20405
7	1	6776	1404126	1385371	25531
8	3	10137	4791630	4777938	23829
9	5	8431	7020630	7010422	18639
10	5	7349	7020630	7012594	15385
11	4	6358	6195756	6190739	11375
12	5	3189	9916890	9914894	5185
13	25	1583	50388976	50384667	5892
14	21	2553	52783905	52781430	5028
15	9	2454	99234130	99231148	5436
16	16	1286	172314864	172311119	5031
Total	-	-	413307724	413152742	-

Table 1 - Generated negative examples.

The amount of evaluated sub-windows in a given full resolution image is of course dependent on the resolution but also on the distance that the sub-window is moved. For the first and second stage the sub-window was moved a distance equalling its width which resulted in no overlap between sub-windows of the same size. For higher stages this distance was reduced in order to extract more negative examples from the same image. This is seen at stage 15 where almost twice the amount of sub-windows were generated using only half the amount of images compared to stage 14.

At the beginning of the implementation of the entire system the AdaBoost algorithm was believed to be the most computationally heavy part. This was also true for the first couple of stages, but when stage numbers got above 10 the generation of negative examples turned out to be the most time consuming task. The explanation simply being the vast amount of sub-windows the script has to generate, rescale and test. The rescaling is performed by the build in Matlab function "imresize" using bicubic interpolation which adds to the overall slow execution.

Due to time constraints an effective parallelized version of the script "algo02.m" was not constructed and thus the time required for generation of negative examples ended up determining the amount of stages in the cascade. At the time of writing negatives for the 16^{th} stage is being generated and it is estimated that this will take some 10 days.

For testing whether a rescaled sub-window survives the existing stages a mex function called "EvalSamp.dll" was programmed in C [8]. Also the script "algo03.m" which collects all the negative examples into one Matlab data file was made. The Matlab scripts "algo02.m", "algo03.m" and the "EvalSamp.c" source code can be found on the enclosed DVD.

Training a stage in the cascade

In the cascade a stage is actually just a strong classifier trained by the modified AdaBoost algorithm. During the course of this project two major different implementations of the modified AdaBoost algorithm were constructed:

The sequential version
Made as a mex function. Programmed in C.
This is more or less a straight forward implementation of the modified AdaBoost algorithm described in Figure 5, Figure 5. The first version loaded the examples from the hard disk drive. This proved to be very inefficient because of the many times each example has to be evaluated. The second version uploaded all examples to memory and instantly proved more efficient albeit more memory consuming.
As described earlier the best performing feature is determined by a brute force search through all possible features. This approach requires a sorting of all examples for every feature evaluated. This sorting was initially done by a classic bubble sort, but was soon replaced by a build in quick sort, which also contributed to a faster execution.

The parallel version
Made as a Matlab script that calls four mex functions. The mex functions are programmed in C.
In the sequential version a for-loop runs through all features and for each feature all examples are evaluated. Since the iterations in the for-loop are independent of each other the algorithm can be easily parallelized. This was done using the Matlab R2006b Parallel Computing Toolbox.

The sequential version was executed on a Zepto laptop containing 1 GB RAM and a 2 GHz Intel Centrino Dothan CPU. The parallel version was executed on IMM's Linux cluster consisting of 30+ servers each running a 64 bit version of Linux on a 1 GHz dual core AMD CPU. Recorded performance results are listed in Table 2.

System	Features [F]	Positives [P]	Negatives [N]	Total time [T]	Parallel jobs [J]	$\frac{T}{F \cdot (P+N)}$	$\frac{J \cdot T}{F \cdot (P+N)}$
Zepto	3	5000	23560	13334 s	1	0.1556	0.1556
Zepto	10	5000	20160	39143 s	1	0.1556	0.1556
Zepto	10	5000	23829	45121 s	1	0.1565	0.1565
Zepto	10	5000	23829	45466 s	1	0.1577	0.1577
Cluster	10	5000	20627	4464 s	16	0.0174	0.2787
Cluster	100	5000	5185	24999 s	16	0.0245	0.3927
Cluster	140	5000	5892	28970 s	18	0.0190	0.3420
Cluster	260	5000	5028	56700 s	18	0.0217	0.3914
Cluster	320	5000	5436	62025 s	18	0.0186	0.3343

Table 2 - Training performace.

The first simple observation that the figures in Table 2 warrants is that the parallel implementation as expected is much faster than the sequential. It is seen that given the same number of features and roughly the same number of examples the parallel implementation is almost 10 times faster.
Secondly it is worth noting that the execution time is directly proportional to the product of features and examples. This is because each new feature to be determined requires exactly the same operations to be carried out and these operations have to evaluate every submitted example. It is thus possible to calculate how much time is used to evaluate just one example. The interesting point is not that the cluster is faster than the laptop, but rather that the laptop produces a fairly constant figure (0.15) whereas the cluster fluctuates a bit. The explanation of course being that the cluster has many users and therefore a varying workload.
Lastly to give the laptop a little credit it should be noted that it works approximately twice as fast as the individual nodes in the cluster. The explanation could again be that the cluster nodes are not fully committed to the task and also that the parallel computing toolbox adds a little overhead to the calculations.

The mentioned Matlab script and source code can be found on the enclosed DVD.

Training the cascade

Each stage in the cascade was trained using a positive set, a negative set and for performance measurement an evaluation set. For each stage the positive set and the evaluation set was the same (data set A) while the negative set was especially designed for exactly that stage.

As described earlier false positives are preferred over false negatives and since the AdaBoost algorithm aims at minimizing false negatives it needs a little tweaking. In their paper Viola-Jones describe a procedure that gradually lowers the classifier threshold of a stage until a given performance requirement is met.

Since the evaluation set in this project only consists of positive examples the true positive rate became the key measurement for a stage's performance. Secondary the false positive rate was estimated by letting the existing cascade evaluate the current negative examples. Normally it is not advisable to use the training data as evaluation data, but due to the cascaded structure it can be allowed in this case. Especially for the higher stages the negative training sets were generated by running through literally millions of examples and thus ensuring training sets almost identical to the real world data the cascade is expected to encounter.

For the lowering of a given stage's threshold an alternative version of the procedure described by Viola-Jones was used. This procedure is presented in pseudo code in Figure 12.

1) Gradually lower the threshold in appropriate steps until a given true positive rate is achieved.
2) Run the negative examples through the cascade using the threshold determined in 1).
If false positive rate is above 50 %:
3a) Increase the amount of features by an appropriate number and go back to 1).
Else:
3b) Use the current threshold and amount of features for this stage. Generate a new negative training set and a new stage containing the same amount of features as the current stage. Now focusing on the new stage go back to 1).

Figure 12 - Procedure for training the cascade.

The term 'appropriate' may seem a bit informal, but the presented pseudo code was never implemented - it was rather used as a rule of thumb during the training of the cascade. For the first stages an appropriate step was a 5 % reduction and for the later stages this figure was down to 0.1 %. Correspondingly five features were added in 3a) for the first stages and later this was increased to twenty features at a time.

For the first stage 93.4 % of the positive examples in the evaluation set were correctly classified when no tweaking was applied. This number was raised to 97.4 % using a tweak factor of 0.85. Among the negative examples used to train the first stage the figures were 81.8 % correctly classified with no tweaking and 61.9 % with tweaking. This clearly proves that the price of an improved true positive rate is a decreased true negative rate (or equivalently: an increased false positive rate).

It should be noted that it had no effect to further lower the tweak factor of the first stage unless it was lowered to 0 – which resulted in the absurd situation that all positive and negative examples were classified as positives. The training results of each individual stage are shown in Table 3.

Stage	Features	Tweak factor	Positive evaluation set		Negative training set	
			TPR	FNR	FPR	TNR
1	2	0.8500	0.9740	0.0260	0.3815	0.6185
2	10	0.5500	0.9740	0.0260	0.4157	0.5843
3	20	0.6000	0.9740	0.0260	0.4919	0.5081
4	20	0.6500	0.9740	0.0260	0.4864	0.5136
5	40	0.8000	0.9740	0.0260	0.2272	0.7728
6	40	0.8000	0.9740	0.0260	0.3321	0.6679
7	50	0.8000	0.9740	0.0260	0.3970	0.6030
8	60	0.8500	**0.9720**	0.0280	0.3538	0.6462
9	60	0.8500	**0.9700**	0.0300	0.3943	0.6057
10	80	0.8500	0.9700	0.0300	0.4133	0.5867
11	80	0.9000	**0.9680**	0.0320	0.2804	0.7196
12	140	0.9000	0.9680	0.0320	0.3053	0.6947
13	180	0.9000	0.9680	0.0320	0.4333	0.5667
14	220	0.9000	0.9680	0.0320	0.4881	0.5119
15	220	0.9300	0.9680	0.0320	0.2366	0.7634
16	400	0.9192	0.9680	0.0320	0.4393	0.5607
Total	1622	-	-	-	-	-

Table 3 - Stage evaluation results.

In Table 3 the boldfaced figures indicate where a slight decrease in the true positive rate was accepted. This was done only if a doubling of the amount of features from one stage to another couldn't reproduce the same true positive rate. Since more features take more time to compute this was a way of sacrificing a little bit of sensitivity in order to keep the amount of features down.

Assuming the evaluation set and the various negative training sets are trustworthy representations of real world faces and non-faces, the hope is that each stage will remove approximately half of the non-faces while preserving almost all the faces.

The Matlab script "algo04.m" runs either the evaluation set or the negative set through the existing stages of the classifier and determines the numbers plotted in Table 3. The script can be found on the enclosed DVD.

The final face detector

The face detector is implemented as a Matlab mex function that requires four input arguments in addition to the input image. These input arguments are the cascade containing the features and tweak factors for each stage, a starting scale, a starting delta and a scale increment. The base resolution of the detector is 24*24 pixels, but using a starting scale of e.g. 2 the first detector sub-window will have a size of 48*48 pixels. The starting delta is also multiplied with the starting scale and the result equals the step size used to move the detector sub-window through the input image. Once the first pass is completed both the size of the sub-window and the step size is incremented by a factor equal to scale increment. This procedure is repeated until the size of the detector sub-window exceeds the least dimension of the input image.

By varying the input parameters it is possible to decide how thorough the detector shall be and it is also possible to take any existing a priori knowledge regarding the input image into account. However, if no a priori knowledge is present the detector should use a small starting scale (1 or 2) and a step size of 1 pixel. Finally Viola-Jones advise using a minimum scale increment of 1.25.

Since the faces in the positive training data are not perfectly aligned the final detector is relatively insensible to small variations in position. As a consequence a (detected) face normally generates a dense swarm of positive detections which can be quite confusing to look at and work with. To counter this problem the face detector is followed by two algorithms that merge overlapping detections. In the first algorithm two detections are merged if they have equal size and they overlap with 25 % or more. In the second algorithm two detections are merged if their centres coincide. As long as the detector is not yet done this merging affects the performance figures in a negative direction since the amount of visible true positives are more heavily reduced than the amount of visible false positives.

Due to the time-consuming task of generating negatives the detector developed in this project ends up having only 16 stages. This is less than half the 38 stages in the detector described by Viola-Jones in their paper. Despite this big difference in stages and the fact that the 16-stage detector is not yet done it already shows remarkable good performance results. In fact the performance is so promising that some 20 stages might end up being sufficient.
The explanation for this is that the individual stages of the 16-stage detector are trained more intensively than the stages in Viola-Jones' detector. In their paper Viola-Jones mention that 10000 negatives are used for the first stage and 5000 are used for the subsequent stages. This should be compared to the approximately 20000 negatives used for the first ten stages in this project. This high number of negatives was used to make sure a sufficient amount of negatives were discarded by the first stages. In hindsight this decision seems to be wrong. The speed of the detector is not determined by the number of stages used to discard a negative, but rather by the average number of features evaluated in order to discard a negative. With this in mind it seems highly probable that the simpler stages of the Viola-Jones classifier is a more efficient approach.

The amount of features used further support the idea of the intensively trained classifier. In their classifier Viola-Jones use a total of 6060 features distributed over the 38 stages. In comparison the 16 stages of the developed detector contain 1622 features and 400 of these are in the 16^{th} stage alone. As the discrimination between faces and non-faces becomes increasingly more difficult more features are needed and a future stage 17 will most likely contain at least some 500 features. Given

this rapid increase some 6000 features will be reached around stage 20 and this supports the notion that the developed detector discards non-faces more aggressively than the detector described by Viola-Jones.

As the 16-stage detector is not yet done final classification performance can't be compared to other systems. Nevertheless the detector's ability to efficiently discard non-faces while still preserving faces can easily be demonstrated. For this purpose four images of equal resolution (600*800) are run through the cascade and the classification results are recorded after each stage. The results are recorded before merging and can be seen in Table 4.

Stage	Image 1 Contains 1 face		Image 2 Contains 2 faces		Image 3 Contains 1 face		Image 4 Contains no faces	
	#	%	#	%	#	%	#	%
1	851241	33.79	753507	41.39	840247	34.64	836548	34.93
2	431886	49.26	385987	48.77	319908	61.93	500966	40.12
3	189424	56.14	217009	43.78	144153	54.94	312076	37.71
4	66381	64.96	110264	49.19	63382	56.03	173503	44.40
5	12235	**81.57**	28865	**73.82**	12681	**79.99**	64749	**62.68**
6	6915	43.48	15686	45.66	7224	43.04	40310	37.74
7	5088	26.42	12389	21.02	5807	19.62	30370	24.66
8	2619	48.53	6187	50.06	2975	48.77	13181	56.60
9	980	62.58	2249	63.65	1132	61.95	4393	66.67
10	265	**72.96**	688	**69.41**	446	**60.60**	1540	**64.94**
11	147	44.53	428	37.79	249	44.17	567	63.18
12	106	27.89	328	23.36	176	29.32	325	42.68
13	62	41.51	168	48.78	105	40.34	148	54.46
14	49	20.96	133	20.83	78	25.71	90	39.19
15	27	44.90	91	31.58	45	42.31	37	58.89
16	26	3.70	86	5.49	45	0.00	30	18.92

Table 4 - Reduction of non-faces.

The four images are analysed using identical input parameters which results in 1285621 sub-windows being processed for each image. As Table 4 shows some 800000 sub-windows survive the first stage meaning it only manages to sort out some 33 %. This figure should be compared to the 50 % mentioned by Viola-Jones. The most likely explanation for this difference is that the negative examples used to train the first stage simply aren't good enough.
The boldfaced figures in Table 4 indicate the cases where huge reductions can be attributed to significant increases in the amount of features. Stage 5 contains twice as many features as stage 4 and between stage 9 and 10 the amount of features are increased by a factor one and a half.
It is also worth noting that even though the amount of features is almost doubled between stage 15 and 16 no noticeable reductions are seen. The most likely explanation for this is that the non-face sub-windows encountered have a pattern so unique that the classifier needs even more features to make a successful discrimination.

In Figure 13 the result of running image 1 through the existing stages of the classifier can be seen. A green rectangle indicates a detection consisting of 5 or more instances and a red rectangle consists of 4 or less.

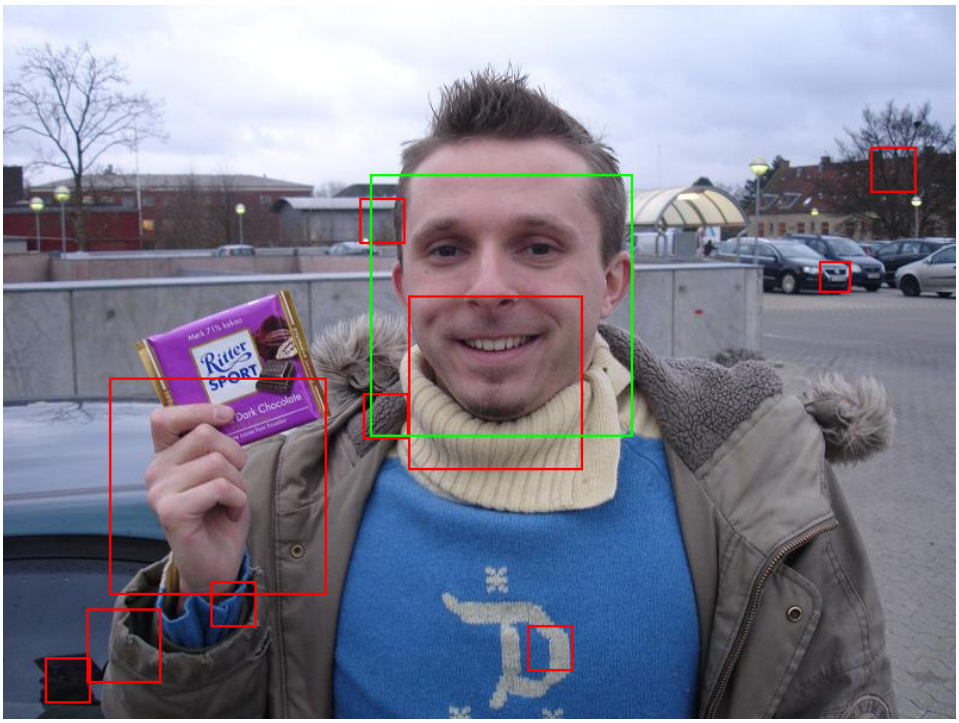


Figure 13 - Image 1 with merged detections.

Out of the 26 positives mentioned in Table 4 the amount of false positives are 16 and it follows that 10 must be true positives. It can be seen that many of the false positives are relatively small and consequently better performance can be obtained by selecting a higher starting scale. Had the image been the starting frame of a video stream this knowledge could have been utilized in the next frame thus reducing the time required for processing.

As the detector is not yet done better performance could be expected after the addition of more stages. Unfortunately the very small reduction made by stage 16 has the potential to undermine this approach. As an alternative way of reducing the amount of false positives Viola-Jones suggest using three differently trained parallel detectors together with a simple voting scheme. This scheme being that two detectors have to agree before a detection is recorded. Due to time constraints this scheme was not implemented.

The result of the three other test images with merged detections can be seen in Appendix 3, page 35.

The face detector is implemented in a mex file called “ViolaJones.dll” and is called from the Matlab script “algo05.m”. The script and the source code can be found on the enclosed DVD.

A simple comparison

Because of the incomplete detector no thorough comparison of e.g. ROC curves can be done. However Viola-Jones show some images overlaid with detections in their paper. These images stem from the MIT + CMU test set and they have also been subjected to analysis using the 16-stage detector. This enables for a simple performance comparison. It is strongly suggested that the images shown in Figure 14 are compared to page 152 in [1]. (The starting scale is 2 and no tweaking is applied to the threshold of the final stage.)

Figure 14 – Test images from the MIT + CMU test set.

The statistics for the images shown in Figure 14 are plotted in Table 5. The image in the upper left corner is 1 and the image in the lower right corner is 9.

Image	True positives		False positives	
	#	%	#	%
1	6	66.67	3	33.33
2	78	98.73	1	1.27
3	98	88.29	13	11.71
4	4	33.33	8	66.67
5	40	90.91	4	9.09
6	5	100.00	0	0.00
7	42	97.67	1	2.33
8	24	92.31	2	7.69
9	8	88.89	1	11.11

Table 5 – Statistics for the MIT + CMU images.

Some of the figures plotted in Table 5 should be taken with a pinch of salt. For instance image 9 shows a high amount of true positives, but these should be compared to the amount of false negatives – which unfortunately are not available. Despite this a visual inspection quickly reveals that three faces have been discarded. Looking at the images in Figure 14 some comments can be made:

- The bad performance in image 4, 5 and 9 is primarily caused by the starting scale. A starting scale of 1 actually yields a better true positive to false positive ratio, but this is only due to the low resolution of the faces and their large numbers.
- Image 2, 3 and 7 confirms that the current 16-stage detector works better on high resolution images. The true positive to false positive ratio drops for these images if the starting scale is chosen lower.

It can be concluded that the current 16-stage detector works best on images of a certain resolution. Furthermore it is to be expected that a tweaking of the threshold will result in a higher detection rate, but also in more false positives. All in all this only points to that the cascade needs more stages and that the training is not yet done.

Discussion

During the design of a stage the final task is to lower the classifier threshold. As stated earlier this is done in order to minimize the total amount of false negatives and the price is consequently an increase in the false positives rate. This is fully acceptable as long as the stage is succeeded by another stage. However, false positives are not desirable in the final output and therefore the threshold of the final stage should not be lowered. Given that a sufficient amount of positive instances stemming from the same one face survive until the last stage, it seems quite reasonable not to lower the threshold since at least one instance is expected to survive.

Assuming the positive evaluation set is a trustworthy representation of real world faces the current 16-stage detector is so sensible that some 97 % of all (frontal upright) faces is detected. Without any lowering of the threshold this figure is reduced to some 95 % which is still really good. In addition the amount of false positives is reduced and thus better performance achieved.

Experiments show that often more than 50 % percent of the false positives surviving the current 16-stage detector are relatively small. That is, their sizes are between 1 and 4 times the base resolution of the detector sub-window. The reason for this is in part that more small sub-windows exist and as a consequence more small sub-windows survive. Furthermore JPEG-artefacts and the erratic pattern seen at pixel scale caused by white noise in the image recording hardware also seems to generate extra false positives. It follows directly that the mentioned false positives will never be generated if the starting scale is chosen higher, but is this sensible?
Today most non-enthusiast digital cameras easily have a minimum of 5 megapixel and new cameras are often sold with resolutions approaching 10 megapixel. Given these extreme figures it seems very safe to use a higher starting scale. Add to this, that if the face detector is to be succeeded by a recognition system a certain minimum face size is required in order to make a successful extraction of facial features. All in all they only arguments for the small base resolution seem to be that it is sufficient for detection and that it generates a manageable amount of features to be evaluated by the AdaBoost algorithm. In Figure 15 an image processed using no tweaking of the final stage and a starting scale of 4 is shown. (Each detection contains 15+ instances.)

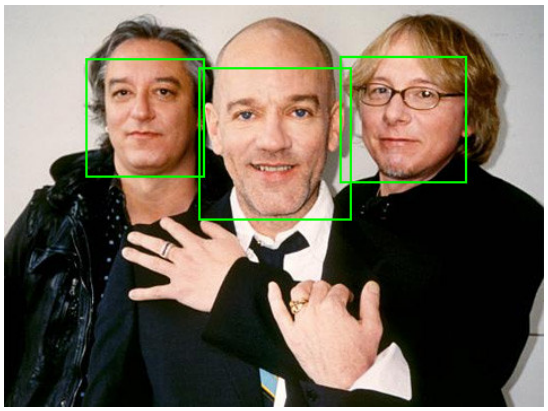


Figure 15 – R.E.M. through the face detector.

At the present time of writing negative examples are being generated for the training of the 17th stage. On an average some 50 examples are generated each day and since 2210 negative examples survived the 16th stage, it will take some 50 days to generate the 2790 negative examples needed to reach a total of 5000. This process is clearly becoming too cumbersome and given the promising performance of the 16-stage detector it seems likely that the 17th stage should be the last one. In order to improve performance the last stage could be upgraded to a kind of final 'expert stage'. So far only a quarter of the 6060 features used by Viola-Jones are used in the 16-stage detector. With this in mind the expert stage could be designed to contain at least some 1000 features hopefully ensuring a high true positive rate and a very small false positive rate. The introduction of this final stage is expected to have minimum impact on execution time since only a very small amount of sub-windows come this far in the cascade.

Should the expert stage prove to be a good idea the face detector developed in this project will end up using less than half the amount of features used by Viola-Jones. As a consequence the average time used to correctly classify a face will be shorter. This however should be compared to the fact that even an image containing many faces still contains vastly more non-faces and thus the average time it takes to discard a non-face is the most influential on total performance.
As mentioned earlier the average amount of features evaluated in order to discard a non-face is the key figure and consequently the cascade design emphasizing many, but simpler stages turns out to be the optimum one.

Finally the image from Figure 1 is run trough the detector and the result is shown in Figure 16. (The starting scale is 2 and no tweaking is applied to the threshold of the final stage.)

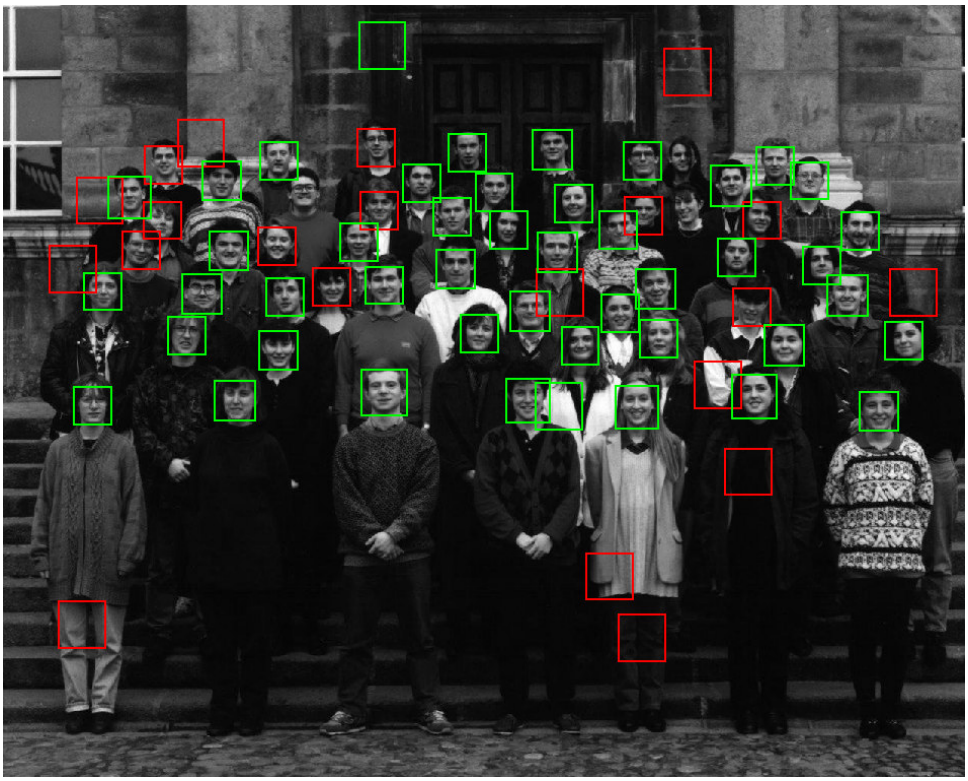


Figure 16 - The typical input image through the detector.

Future Work

As mentioned in the beginning the Viola-Jones algorithm has already been modified in order to detect rotated and profile view faces. This section will therefore not comment on the algorithm, but rather present some of the lessons learned during the implementation.

1) Don't use the FERET faces for training. They are too ideal and therefore not a trustworthy representation of faces encountered in an unconstrained environment. Use faces like the ones in the LFW database.
2) Carefully choose the negative examples for training of the first stage. Generate examples from more than one full resolution image and make sure a broad host of texture types are represented.
3) Use an evaluation set containing more than 500 faces. Also the images should be different sizes and not all rescaled to 24*24 pixels.
4) When generating positive and negative training examples use a method for scaling that reminds of how the detector works. There is a twofold rationale for this. Firstly the generation of negatives will hopefully be speeded up. Secondly the threshold values calculated this way will be better matched to the feature values calculated during detection.
5) Make sure the base resolution actually is 24*24 pixels. An integral image generated from a 24*24 pixels sub-window only allows for the calculation of feature values in a 23*23 patch. If the existing examples are to be used again add a first row and a first column of zeros.
6) Let the AdaBoost algorithm tell how many and which examples are classified correctly after each new feature is added.
7) Use a trained cascade from the internet and spend the time on the face recognition part...

Even though this project could be improved on the mentioned points it is still worth noting that the current 16-stage classifier works pretty well. Furthermore it could be really interesting to examine the performance after the addition of an expert stage.

Conclusion

The purpose of this project was to implement the Viola-Jones face detection algorithm and obtain performance similar to the figures posted by Viola-Jones. While the implementation of the face detector itself proved pretty straightforward the training of the cascade which is used by the detector turned out to be very time consuming. This was in part expected and an algorithm intended for parallel execution was designed to cut down training time. However and surprisingly enough the most time consuming part ended up being the generation of negative examples for training of the steps comprising the cascade.

A large amount of minor problems were solved during the course of this project. Different facial databases were procured and scripts for conditioning of the faces were developed. Three different facial data sets containing more than 15000 faces were constructed. A script for generation and conditioning of negatives was developed. A total of four different implementations of the modified AdaBoost algorithm were also developed. The first implementation worked so slow it did not manage to calculate just one feature and the last implementation being the very efficient parallel implementation. And finally the face detection algorithm itself was implemented as an easy-to-use Matlab function.

Unfortunately the cascade was never completed, but the tendency demonstrated by Table 4, page 25 seems to indicate that the training process is working since non-faces are discarded while faces are preserved. Add to this that the preliminary results of the developed 16-stage detector already are very promising. All in all the different parts of the algorithm is working and it is only a matter of time before the cascade is completed.

Appendix 1 - Literature List and references

[1] Paul Viola, Michael J. Jones, *Robust Real-Time Face Detection*, International Journal of Cumputer Vision 57(2), 2004.

[2] Paul Viola, Michael J. Jones, *Fast Multi-view Face Detection*, Mitsubishi Electric Research Laboratories, TR2003-096, August 2003.

[3] Henry A. Rowley, Shumeet Baluja, Takeo Kanade, *Neural Network-Based Face Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 20, Issue 1, Jan 1998. Digital Object Identifier 10.1109/34.655647.

[4] Henry Schneiderman, Takeo Kanade, *A Statistical Method for 3D Object Detection Applied To Faces and Cars*, IEEE Conference on Computer Vision and Pattern Recognition 2000 proceedings, Volume 1. Digital Object Identifier 10.1109/CVPR.2000.855895.

[5] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, first edition, Springer 2006.

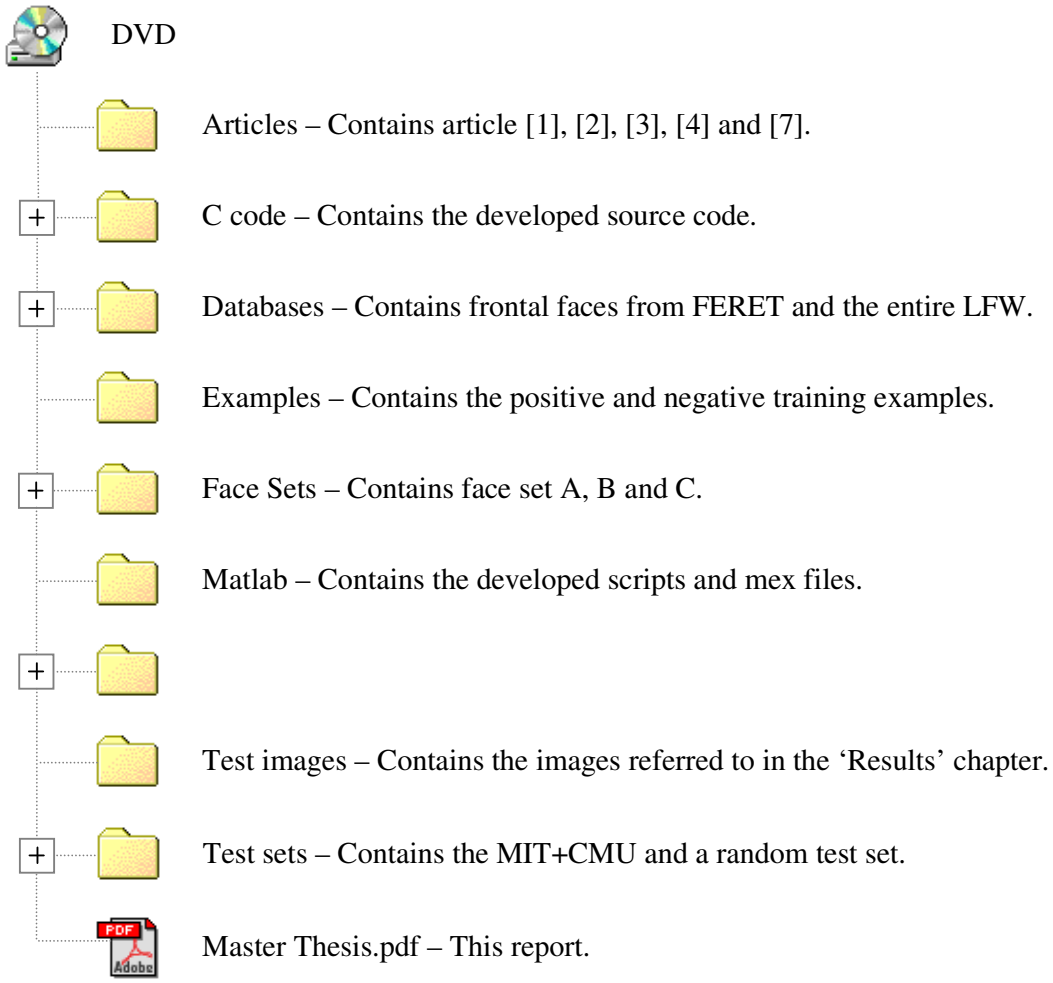
[6] P. J. Phillips, H. Moon, *The Facial Recognition Technology (FERET) Database*. http://www.itl.nist.gov/iad/humanid/feret/feret_master.html

[7] Gary B. Huang, Manu Ramesh, Tamara Berg, Erik Learned-Miller, *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*, University of Massachusetts, Amherst, Technical Report 07-49, October 2007. http://vis-www.cs.umass.edu/lfw/index.html

[8] Brian W. Kernighan, Dennis M. Ritchie, *The C Programming Language*, second edition, Prentice Hall Software Series, 1988.

Appendix 2 - Contents of the enclosed DVD

The enclosed DVD contains:



Appendix 3 - Image 2, 3 and 4 after detection.

Here image 2, 3 and 4 is shown after detection. The developed 16-stage classifier is used with tweaking of the last stage. Detections are merged.

Image 2:

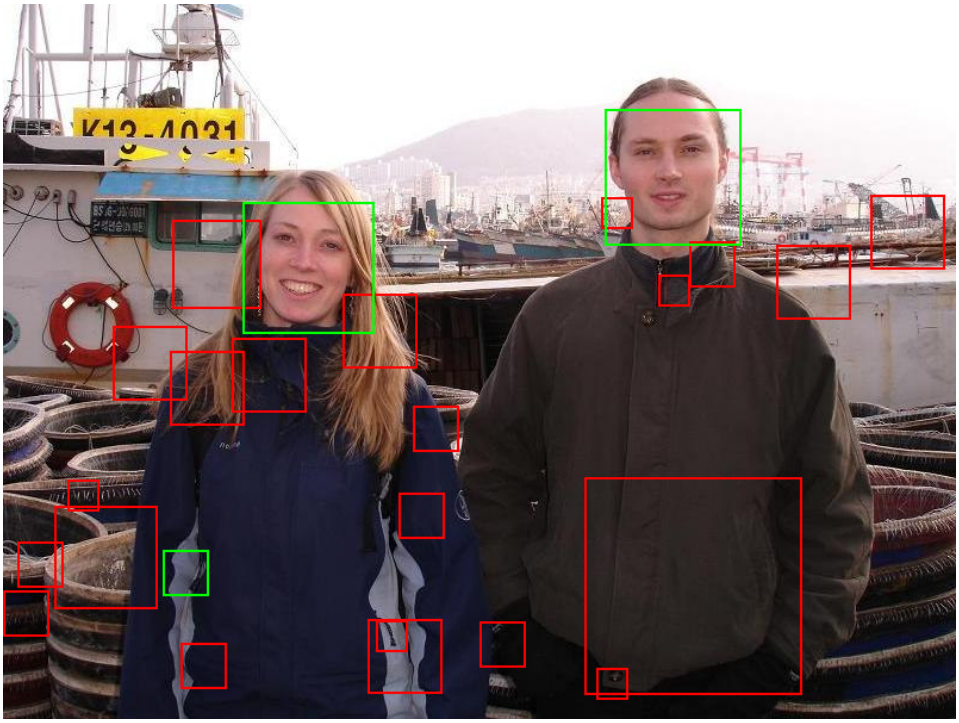


Image 3:

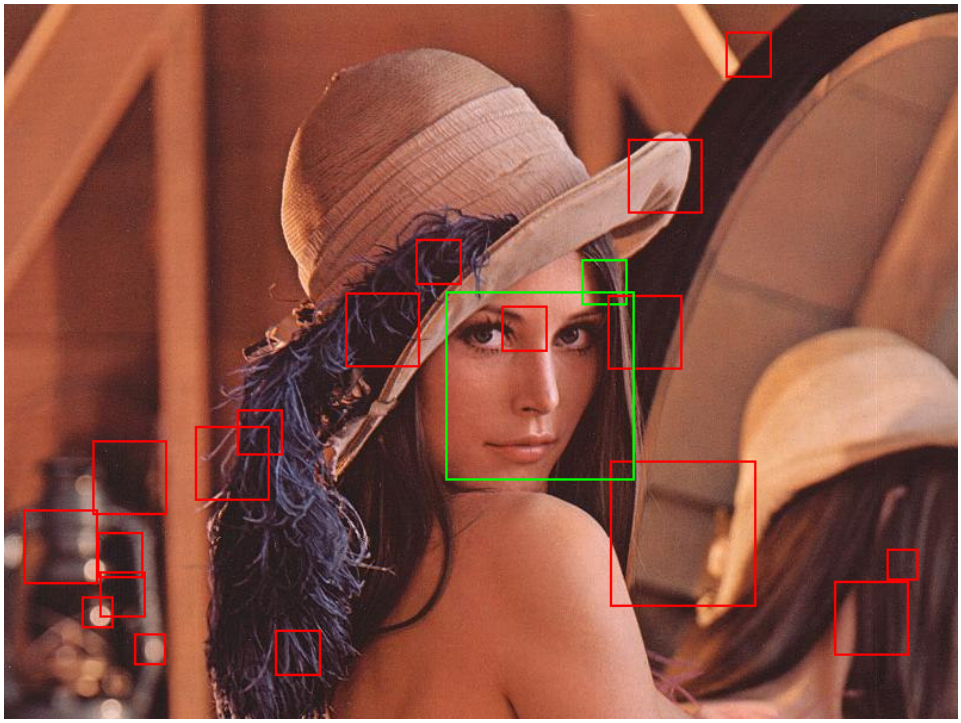


Image 4: