Diploma thesis
Face detection with Waldboost algorithm

Zdeněk Kálal
zdenek.kalal@gmail.com

January 18, 2007

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 19. 1. 2007

Zdeněk Kálal

Acknowledgement

I would like to thank my supervisor, doc. Dr. Ing. Jiří Matas, for the guidance and advice he has provided throughout the research. I would also like to thank Jan Šochman for helping me especially in the technical issues.

Abstract

Face detection algorithms based on the work of Viola and Jones [11] train the classifier by processing training examples of face and non-face patterns. A general effort is to process a large number of training examples and hence describe the problem accurately. Current approaches are based on the assumption that non-face patterns can be easily obtained contrary to faces. While the faces remain in the training the whole time, the easy non-face patterns are stepwise replaced with more difficult ones. In the case when a large number of faces is at disposal, such conventional techniques are not able to process the whole positive training set effectively.

The main contribution of this thesis is the improvement of the Waldboost algorithm [12] by which the positive training patterns are treated in a similar way as non-face patterns, called symmetric bootstrap. In the symmetric bootstrap, both the easy positive- and easy negative- patterns are stepwise replaced with more difficult ones and hence bigger space of faces is explored in the training.

In order to show the effect of symmetric bootstrap, sufficiently large pool of positive training patterns had to be found and the non-face distribution appropriately modeled. A new bounding box alignment, which does not require any feature points, was found. This alignment enables us to easily collect training data from different sources (e.g. on-line face detectors) and simplifies the training task, which leads to higher detection rates of the classifier. The space of the positive training patterns was further enlarged by the technique, which is able to generate new faces from already seen ones, called face synthesis. Using synthesis, we are able to generate a large positive training set with user defined parameters controlling in-plane rotation, bounding box shift/scale and wight to height ratio of faces. The non-face patterns are in the symmetric bootstrap modeled by parts of faces, resulting in lower false positive rates.

Except the above mentioned contributions, the original implementation of the Waldboost learning was accelerated by 60% enabling more effective testing of new algorithms. And next, it was shown that features used for classification can be correlated with other useful information such as head pose. This conclusion is elaborated in a new algorithm proposal trying to exploit the entire sequence of feature responses for classification.

According to the experiments, the symmetric bootstrap can improves the performance of the Waldboost in the cases when the whole set of positive training examples can not be processed at once. The improvement is, however, limited by the distinguishing power of the weak features. As in the non-symmetric version we reach the possibilities of the weak learner after some time of training and further improvement is not possible. This break point is reached by the symmetric bootstrap sooner suggesting development of stronger features.

Abstrakt

Učení současných algoritmů detekce obličeje je založeno na zpracování trénovacích příkladů obličejů a pozadí. Obecnou snahou je projít co největší množství příkladů a tím popsat daný problém co nejpřesněji. Algoritmy vycházejí z předpokladu, že negativní příklady můžeme snadno získat skenováním, zatímco obličeje nikoliv. K oběma třídám tedy přistupujeme při učení odlišně. Zatímco obličeje zůstávají v tréninku po celou dobu, prostor pozadí je efektivně zpracován bootstrapováním (jednoduché příklady pozadí jsou postupně nahrazovány složitějšími). V současné době je stále jednodušší získat ohromné množství příkladů obličejů (např. pomocí detektorů), které ovšem tyto standardní algoritmy nejsou schopny zpracovat efektivně.

Základním přínosem této práce je rozšíření učení klasifikačního algoritmu Waldboost [12], pomocí kterého je možné k třídě obličejů přistupovat téměř stejným způsobem jako k třídě pozadí, t.j. bootstrapování obličejů (symetrický bootstrap). Symetrický bootstrap umožňuje zpracovat mnohem větší prostor obličejů což v určitých případech může vést na lepší klasifikátor.

Aby bylo možné demonstrovat efekt symetrického bootstrapu, bylo nutné najít velký zdroj pozitivních trénovacích dat a vhodně modelovat distribuci pozadí. Byl nalezen vhodný způsob zarovnání bounding boxu, který pro svou definici nevyžaduje polohu význačných bodů v obličeji. Tento způsob zarovnání umožňuje snadno získávat nové příklady obličejů z různých zdrojů (např. z detektorů) a vede na rychlejší a kvalitnější klasifikátor. Prostor obličejů byl dále rozšířen syntézou. Syntéza dovoluje generovat libovolné množství pozitivních příkladů s parametry, které specifikuje uživatel (in-plane rotace, posun a zvětšení bounding boxu, změna identity). Negativní příklady jsou v symetrickém bootstrapu modelovány novým způsobem a to, skenováním částí obličeje což vede na menší chybu klasifikace.

Dále bylo ukázáno, že odezvy příznaků, které se typicky využívají pouze pro klasifikaci, mohou být korelovány i s jinou užitečnou informací jako je 3D rotace hlavy. Toto pozorování bylo rozvinuto v návrhu algoritmu, který pro klasifikaci využívá celou sekvenci příznaků. V neposlední řadě byl původní algoritmus urychlen o 60%.

Podle provedených experimentů, symetrický bootstrap může zlepšit klasifikační schopnosti detektoru v případě, kdy nejsme schopni zpracovat celý soubor pozitivních příkladů najednou. Jeho efekt je však do značné míry limitován klasifikačními schopnostmi slabých klasifikátorů. Stejně jako v nesymetrické verzi, trénink se po určitém čase zastavuje a další vylepšení již není možné. Tento bod je ovšem pro symetrický bootstrap dosažen dříve, což naznačuje nutnost vyvinout lepší příznaky.

České vysoké učení technické v Praze - Fakulta elektrotechnická

Katedra kybernetiky ***Školní rok:*** 2005/2006

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Zdeněk K á l a l

Obor: Technická kybernetika

Název tématu: Detekce obličeje pomocí algoritmu WaldBoost

Zásady pro vypracování:

1. Seznamte se s klasifikačním algoritmem WaldBoost vyvíjeného v Centru strojového vnímání. Urychlete učení stávajícího algoritmu.
2. Prozkoumejte možnosti bootstrapu na pozitivní příklady.
3. Navrhněte a implementujte změny směřující k efektivnějšímu klasifikátoru s menší chybou.
4. Výsledný detektor porovnejte s původní verzí na úloze detekce obličeje.

Seznam odborné literatury:

[1] Wald, A.: *Sequential Analysis*. Dover, New York, 1947

[2] Freund, Y.; Schapire, R.E.: *A Decision - Theoretic Generalization of On-Line Learning and an Application to Boosting*. Journal of Computer and System Sciences. 1997

[3] Šochman, J.; Matas, J.: *WaldBoost - Learning for Time Constrained Sequential Detection*. Proceedings of Conference on Computer Vision and Pattern Recognition. 2005

Vedoucí diplomové práce: doc. Dr. Ing. Jiří Matas

Termín zadání diplomové práce: zimní semestr 2005/2006 (změna zadání: 28.11.2006)

Termín odevzdání diplomové práce: leden 2007

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická

prof. Ing. Zbyněk Škvor, CSc.
děkan

V Praze dne 11.12.2006

Contents

Chapter 1

Introduction

Face detection is a computer technology capable of localizing a face in an input image regardless of different head orientation, scale, light conditions and facial expression. It detects facial features and ignores anything else, such as bodies, trees and buildings. It can be understood as a special case of object detection whose objective is to localize any object that falls into a given class (pedestrians, cars, traffic signs). Figure 1.1 illustrates the face detection technology. The yellow squares mark the places which were labeled as faces.

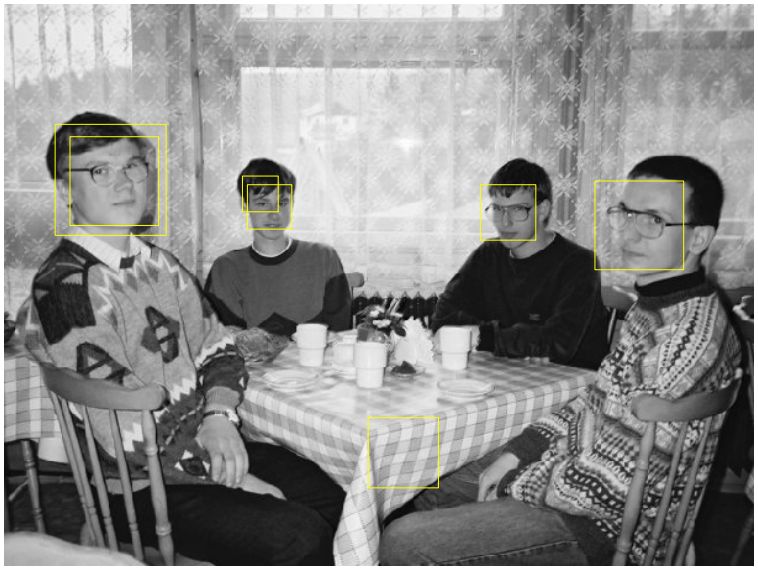


Figure 1.1: Output of a face detector. The yellow squares mark the places which were labeled as faces.

The face is important for identification of others and conveys significant social information. Psychological processes involved in face perception are known to be present from birth and involve large and widely distributed areas in the brain. Hence, it can be easily understood that people are able to localize a face effortlessly and with high accuracy. That is why we can see immediately, which detections in the figure 1.1 are right and which are false.

The ability to quickly localize a human face is not unlimited, though. There exist images, such as figure 1.2, in which the face can not be seen immediately. Even human have to search for the face by brute-force, i.e. scan the image for every position, scale and rotation which is an approach often adopted by face detectors.

Figure 1.2: Can you see the face? Example of an image, where a face is highly unlikely. Even human have to search the face by brute-force, i.e. for every position, scale and rotation. The hidden face is upside down in the top left corner of the image. The original image was downloaded from Worth1000.com

Area	Specific applications
Biometrics	Drivers' Licenses, Passports
Information Security	Desktop Login, Internet security, Secure terminals
Surveillance	CCTV Control, Post-event analysis, Suspect Tracking
Access Control	Vehicular access, Facility Access
Entertainment	Film Annotation

Table 1.1: Possible applications of face recognition technology

1.1 Face detection applications

Face detection is often considered as a part of the *face recognition* technology, which aim is to recognize the identity. Face recognition applications are listed in the table 1.1.

Direct applications of face detection exist and some of them are listed in the table 1.2. This list is not enclosed but still growing due to the dynamic progress in the field.

Area	Specific applications
Surveillance	Secure ATM terminals
Entertainment	Auto-focus in a camera, Web cam focus
Mobile	Video phone

Table 1.2: Possible applications of face detection technology

1.2 Related work

Face detection is one of the fastest growing technologies in the computer vision community. There are tens of research teams all over the world and thousands of papers published in the resent years. There is a growing number of vendors applying the technology into real application. There are open source initiatives developing algorithms for various computer vision problems including face detection. And hence, it is not easy to give a short overview the entire problem.

Face detection has been investigated for more than 30 years. The main approaches of solving this problem are summarized in the following list. It is, however, difficult to classify current approaches to any of these categories.

Ad hoc methods This top-down approach represents the face by a set of human-coded rules. The search can be done by multi-resolution focus-of-attention approach. This method works well for simple tasks with uniform background but fails when cluttered background or pose change is considered, because it is implausible to enumerate all possible cases [Yang and Huang 94].

Feature-based methods This bottom-up approach aims to find structural features of a face that exist even when the pose, viewpoint, or lighting conditions vary. The found features are grouped into candidates which geometric arrangement is verified [Leung et al. 95].

Template matching methods Several standard patterns stored to describe the face as a whole or the facial features separately. Templates has to be initialized near to the images [Sinha 94].

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

Our approach can be classified as a combination of the feature- and appearance-based method and further separated according to the figure 1.3.

In the field of appearance-based face detection, boosting [1] is one of the most important recent classification methodologies. Boosting works by sequentially applying a classification algorithm to reweighted versions of the training data, and then taking a weighted majority vote of the sequence of classifiers thus produced. For many classification algorithms, this simple strategy results in dramatic improvements in performance.

Several generalizations to the discrete AdaBoost algorithm were proposed in [8]. Particularly in a setting in which hypotheses may assign confidences to each of their predictions, i.e., real AdaBoost.

A principal paper [11] in the field of rapid face detection was published by Viola and Jones in 2001. AdaBoost algorithm is used here for selecting a small number of critical visual features from a large set. The computation of these features is very quick due to the concept of Integral Image. The selected features are combined in increasingly more complex classifiers which are combined in a cascade. The cascade can be viewed as an object specific focus-of-attention mechanism which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions.

The detection framework proposed by Viola and Jones 2001 is extended in [4] to handle profile views and rotated faces. As in the work of Schneiderman et al. 2000

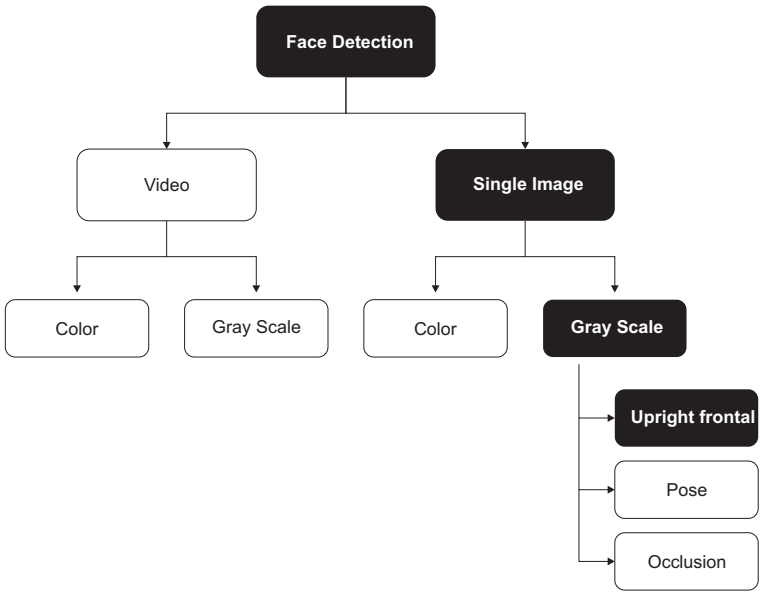


Figure 1.3: One possible classification of the face detection task. The direction addressed by our research is denoted by the black color.

[9], different detectors for different views of the face are constructed. A decision tree is then trained to determine the viewpoint class. There are a few other systems which explicitly address non-frontal, or non-upright face detection such as [5].

A frontal face detection method based on combining the results of individual face parts classifiers is proposed in [3]. This approach increases the robustness of the detector again occlusion.

Mainly off-line training have been used in all of the above mentioned methods, which implies that all training data has to be a priori given. The on-line training of the classifier was introduced in [2], which opens new areas of application for boosting in computer vision.

A sequential classification algorithm called Waldboost [12] is proposed by Šochman and Matas in 2004. It integrates the AdaBoost algorithm for measurement selection and ordering with Wald's sequential decision-making [13]. This paper represents a fundamental material for this thesis.

Face detection algorithms based on the work of Viola and Jones [11] train the classifier with a large number of non-face patterns and much lower number of faces. Such an approach is based on the assumption that non-face patterns can be easily obtained contrary to faces and hence both classes are treated differently. While the faces remain in the training the whole time, the easy non-face patterns are stepwise replaced with more difficult ones. In the case when a large number of faces is at disposal, conventional techniques are not able to process the whole set effectively.

1.3 Organization of the thesis

The main aim of this thesis is to develop an improvement to the Waldboost algorithm able to process effectively a large set of positive training examples. The remaining chapters of this thesis are organized as follows.

Current Implementation of Waldboost Classifier This chapter is going to describe the implementation of the Waldboost face detector that is being devel-

oped at Center for Machine Perception (CMP). In order to fully understand it's function a short introduction to the basic concepts in face detection will be given. Further the sequential probability ratio test (SPRT), will be explained. The SPRT together with real AdaBoost comprise the core of the Waldboost algorithm that will be introduced at the end.

Algorithm Profiling The acceleration of the original Waldboost algorithm by 60% is introduced. The output of this section is an accelerated code, with rather essential changes in the way the training examples are represented.

Training Data Preparation In order to demonstrate the symmetric bootstrap, sufficiently large positive training set is required. Sice the original version of Waldboost works only with fixed amount of faces, new sources of training examples are investigates in the section *Enlargement of the positive training set by face synthesis*. Except the new databases obtained from 3rd party face detectors, face synthesis is introduced, which enables arbitrarily enlargement of training set with user-defined distribution. A new approach to bounding box alignment is suggested in the section *Unification of face databases using proper bounding box alignment*. The suggested alignment enables exploiting different sources of training examples and simplifies the task by finding the alignment that is the easiest for some previously learned detector. Proper modeling of the background is discussed in the section *Background generated from parts of the face*. Such background modeling is important for correct function of the symmetric bootstrap and reduces the false positive alarms.

Bootstrapping of Positive Training Examples This chapter deals with the improvement of the Waldboost algorithm [12] by which the positive training patterns are replaced nearly the same way as negative, called symmetric bootstrap. In the first part, the distribution of example-weights during learning are discussed. Next, the proper estimation of the thresholds for classification and relative error of the estimated threshold are introduced.

Pose Estimation with Linear Regression A different direction in the research is discussed in this chapter, i.e. the correlation of the feature responses (used for classification) with some other useful information. A simple linear algorithm is implemented where the head pose is estimated using linear regression.

Further Work This chapter suggests further steps that can be taken in the research.

Experiments Only the most important experiments will be introduced, which results justify the conclusions done.

Conclusions Summary of the conclusions and observations.

Chapter 2

Current Implementation of Waldboost Classifier

This chapter is going to describe the implementation of the Waldboost face detector that is being developed at Center for Machine Perception (CMP). Our detector can be classified as a appearance-based sequential binary classifier. In order to fully understand it's function a short introduction to the basic concepts in face detection will be given in the firs section. Further the sequential probability ratio test (SPRT), will be explained. The SPRT together with real AdaBoost comprise the core of the Waldboost algorithm that will be introduced at the end of this chapter.

2.1 Face detection basics

Face detection proved to be a vary challenging task because of the large variability of human faces (both inter- and intra- personal) and the numerous sources of noise that influence the performance of the system. Another aspect that has to be taken into account is the fact that the face is not a static object but a dynamic one that evolves and interacts in a changing environment.

Addressing directly the task of face detection proved to be an unfeasible approach so it is usually divided into a number of subtasks. Our research aims at addressing the task of **frontal face detection from static gray scale images**. In other words we don't use any information about color of the skin, or motion extracted from a sequence of images. These sources of information can be, however, added to our system and further increase it's performance.

Figure 2.1 demonstrates the frontal upright face appearance variability, where all faces are aligned to fixed position of eyes. Similar images are used in the training of the classifier.

The appearance of the face is determined by a few noise factors that can be classified as follows:

Pose (out-of-plane rotation) Since the head is a 3D object, it's appearance vary according to it's pose. The task of face detection is significantly easier when only some subspace (frontal upright, profile, upside down) of rotation out-of-plane is considered. The illustration of the pose change is given in figure 2.2.

Orientation (in-plane rotation) Face appearance vary with rotation about the camera's optical axis. This rotation can be easily eliminated or added by geometric transformation. Figure 2.2 illustrates the in-plane rotation.

Figure 2.1: The appearance of a frontal upright face aligned to fixed position of eyes is determined by the identity, occlusions, lightning conditions, face expression and pose. All of the images represent one class which has to be completely described by the face detector.

Presence or absence of structural components Beards, mustaches or glasses change significantly the appearance of the face.

Facial expression face appearance is directly affected by a person's facial expression

Occlusion faces may be partially occluded by other objects

Imaging conditions lightning (spectra source distribution and intensity), camera characteristics and resolution of the image

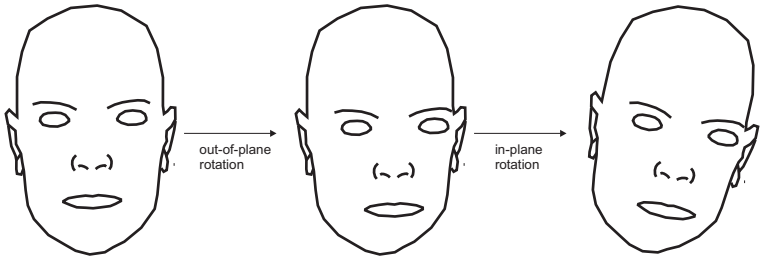


Figure 2.2: The rotation out-of-plane and rotation in-plane are often restricted in order to simplify the task of face detection. The in-plane rotation can be easily eliminated by applying upright detector to a row of rotated images. Other noise factors such as occlusions, facial expression and beards are often considered as noises which are ignored by the detector.

Let's take one more look at the figure 2.1 and try to find common features that distinguish the faces from the background. One could come up with many ides. En example would be a statement that faces are characterized by two eyes, one nose and a mouth and these features are in some specified layout. Such a rule is a typical

example of a knowledge-based system which might be suitable for some easy tasks. There are two problems, however, connected with this approach. First, these rules of thumb are often too abstract for a machine. And second, we are not able to generate sufficiently enough such rules when the task becomes more difficult.

The basic concept is to find such **features** that are easily measurable by a computer and that are capable of describing the face suitably. These features should be invariant again those noisy factors described earlier. The features used in this thesis are going to be discussed in the following section.

2.1.1 Measuring face features by Haar-like filters

A digital image can be understood as a multidimensional measurement of brightness rearranged into a matrix. In the following, the image will be denoted by x and a class (face, background) y. The dimension of the x depends on the size of the image. In our training a maximal size of the image is 40 pixels, in this case the vector x has dimension equals 1600.

The difficulty in modeling $P(x|y)$ is that we do not know the true statistical characteristics of appearance either for the face or for the background. For example, we do not know if the true distributions are Gaussian, Poisson, or multimodal. These properties are unknown since it is not tractable to analyze the joint statistics of large numbers of pixels.

To overcome this limitation, features can be measured on the image x. A feature $f_i(x)$ is a random variable describing some chosen visual characteristics such as low frequency content. In our task of face detection, we are looking for features which can distinguish the face from the background.

In face detection, the Haar-like features are often used because of very fast computation thanks to the Integral Image [11]. The types of features used in our implementation are illustrated in figure 2.3.

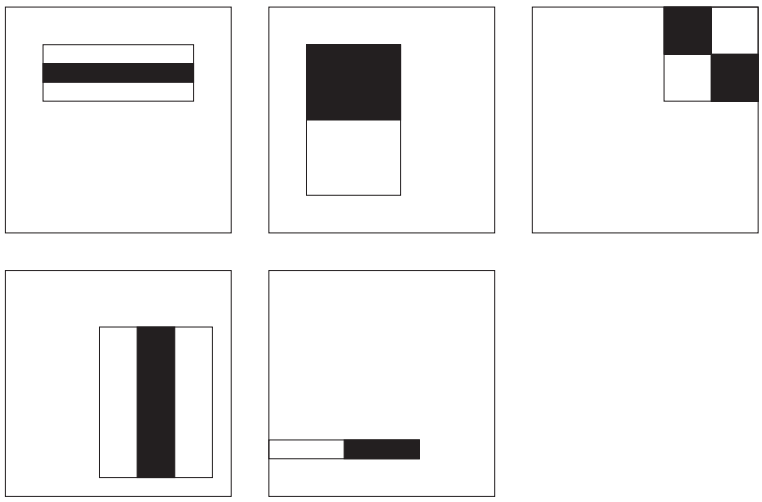


Figure 2.3: Haar-like features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the black rectangles. These features are able to detect some elementary visual attributes such as edges, lines or corners.

The number of all possible Haar-like features measurable on a image 40×40 is much larger than the dimension of the image. Every such feature describes some visual attribute. A question remains, how to choose the appropriate features that

are able to distinguish a face from background. This issue will be discussed in the section denoted to AdaBoost.

2.1.2 Face detection formulated as a binary classification task

Many algorithms implement the face-detection task as a binary pattern-classification task. That is, the content of a given part of an image is transformed into features, after which a classifier trained on example faces decides whether that particular region of the image is a face, or not.

Often, a window-sliding technique is employed. That is, the above-mentioned classifier is used to classify the (usually square or rectangular) portions of in image, at all locations and scales, as either faces or non-faces (background pattern).

There are many sub-windows that has to be tested when the window-sliding technique is employed. The need to evaluate completely every sub-window within the input image was a long time an obstacle for putting face detection technology into real-time applications. A breakthrough in fast face detection came with the work of Viola and Jones [11] that published an algorithm capable of eliminating a huge number of background sub-windows by early decision.

Speed of the classifier, i.e., average number of features measured on one sub-window, becomes a very important parameter. In the following the Waldboost algorithm will be described. This algorithm addresses directly the problem of number of measurements that have to be taken in order to meet some give error rates.

2.2 Waldboost

Waldboost is a meta-algorithm that can be used in conjunction with many other learning algorithms to improve their performance. In the task of face detection, it integrates the real AdaBoost for measurement selection and ordering similarly as in [11]. The number of measurement is, however, formalized in the framework of sequential decision-making [13]; only such number of measurements is taken to meet the given false positive and false negative error rates.

In the following, a short introduction to sequential decision making will be given which comprises the theoretical background for Waldboost sequential classifier explained later on. The learning of the classifier comprises two major parts. First, measurement selection by means of real Adaboost. Second, estimation of thresholds for sequential classification.

2.2.1 Sequential Probability Ratio Test (SPRT)

Sequential analysis refers to statistical analysis where the sample size is not fixed in advance. Instead data is evaluated as it is collected, and further sampling is stopped in accordance with a pre-defined stopping rule as soon as significant results are observed. A final conclusion may therefore sometimes be reached at a much earlier stage than would be possible with more classical hypothesis testing or estimation, at consequently lower financial and/or human cost.

Sequential analysis was first developed by Abraham Wald with Jacob Wolfowitz as a tool for more efficient industrial quality control during World War II.

Let x be characterized by a hidden state (class) $y \in \{-1, +1\}$. This hidden state is not observable and has to be determined based on successive measurements $x_1, x_2, \ldots$. Let the joint conditional density $p(x_1, \ldots, x_m|y = c)$ of the measurements $x_1, \ldots, x_m$ be known for $c \in \{-1, +1\}$ for all m.

SPTR [13] is a strategy S^*, which is defined as:

$$S^* = \begin{cases} +1, & R_m \leq B \\ -1, & R_m \geq A \\ \#, & B < R_m < A \end{cases} \tag{2.1}$$

Where # stands for "take one more measurement" and R_m is the likelihood ratio

$$R_m = \frac{p(x_1, \ldots, x_m | y = -1)}{p(x_1, \ldots, x_m | y = +1)}. \tag{2.2}$$

The constants A and B are set according to the required errors of the first kind α and second kind β. Optimal A and B are difficult to compute in practice, but tight bounds are easily derived. Wald suggests to set the thresholds A and B to their upper and lower bounds respectively

$$A' = \frac{1-\beta}{\alpha}, \quad B' = \frac{\beta}{1-\alpha}. \tag{2.3}$$

In [12] the high dimensional density estimation of $R_m(x)$ is approximated so that this task simplifies to a one dimensional likelihood ratio estimation. The m-dimensional space is projected into a one dimensional space by the strong classifier function H_m

$$\hat{R}_m = \frac{p(H_m(x) | y = -1)}{p(H_m(x) | y = +1)}. \tag{2.4}$$

2.2.2 Sequential binary classifier

The binary classifier was till now understood as a black box with an image at the input and decision about the class at the output. The internal structure of the sequential classifier is illustrated in the figure 2.4. The classifier of length T, which will be denoted H_T, comprises of a row of weak classifiers $h_t(x)$ where $t \in (1 \ldots T)$. Each weak classifiers represents the measurement taken on the input image.

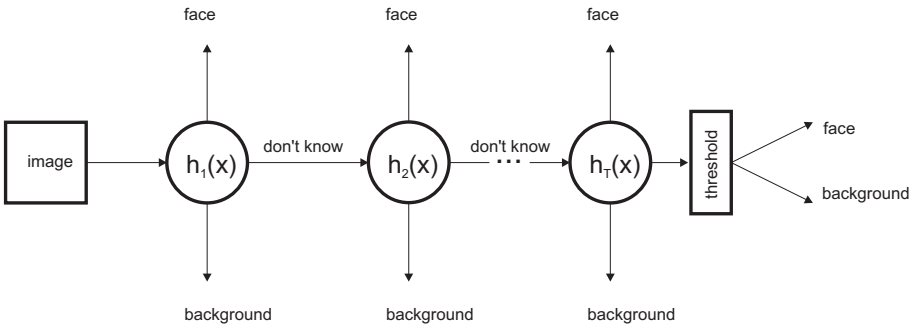


Figure 2.4: Internal structure of sequential binary classifier. The input image is evaluated by the first weak classifier h_1 which output is either decision about the class or the decision is not done and next weak classifier h_2 is evaluated. If the image reaches the end of the classifier the class is decided according to a threshold.

The input image can be understood as a huge set of all possible sub-windows, where only a negligible amount represent faces, and the others represent background. The classifier sequentially classifies background and faces as soon as enough evidence is collected. If the sub-window reaches the end of the classifier, the class is decided according to a threshold.

The principle of the strong classifier should be obvious right now, but it wasn't told yet, how does the weak classifier works. The figure 2.5 illustrates the internal structure of a Waldboost weak classifier.

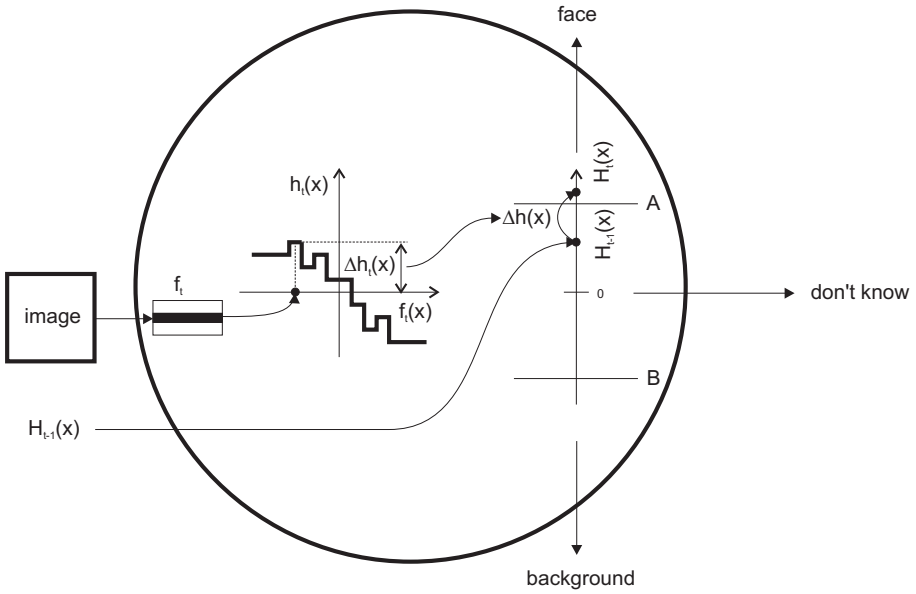


Figure 2.5: Internal structure of a weak classifier used in Waldboost. The input image x and the previous evaluation $H_{t-1}(x)$ enters the weak classifier. A feature f_t is measured on the input image. The feature value $f_t(x)$ is mapped to the output of a weak classifier $h_t(x)$ whose form is given by real AdaBoost. If the previous $H_{t-1}(x)$ and current $h_t(x)$ evaluation together get over any threshold, a decision about the class will be done. Otherwise next weak classifier will be evaluated.

The Waldboost weak classifier consists of three parts. First, the *Haar-like feature* which is measured on the image. Second, the feature response is mapped to form of *real AdaBoost weak classifier* which minimizes the upper bound of training error, having the form:

$$h^{t+1} = \frac{1}{2} log \frac{P(y=+1|x,w^t)}{P(y=-1|x,w^t)} \tag{2.5}$$

And third, the confidence of the weak classifier is summed with the previous response and compared with *thresholds defined by SPRT*. If the enough evidence is collected, the decision is made.

In the rest of this section the learning of the sequential classifier will be discussed, which comprises the search for the best weak classifier and SPRT thresholds estimation.

2.2.3 Real Adaboost for feature selection

There is a huge number of Haar-like features that are measurable at the image. The conditional distribution of two of them is illustrated in the figure 2.6. The question is, which of these are important and which are not. And next, we don't know how to order these features into the sequential classifier.

Discrete Adaboost was successfully used for feature selection and ordering by Viola and Jones in [11]. The basic idea of their algorithm can be stated as: evaluate all features that are at disposal on current distribution of positive and negative

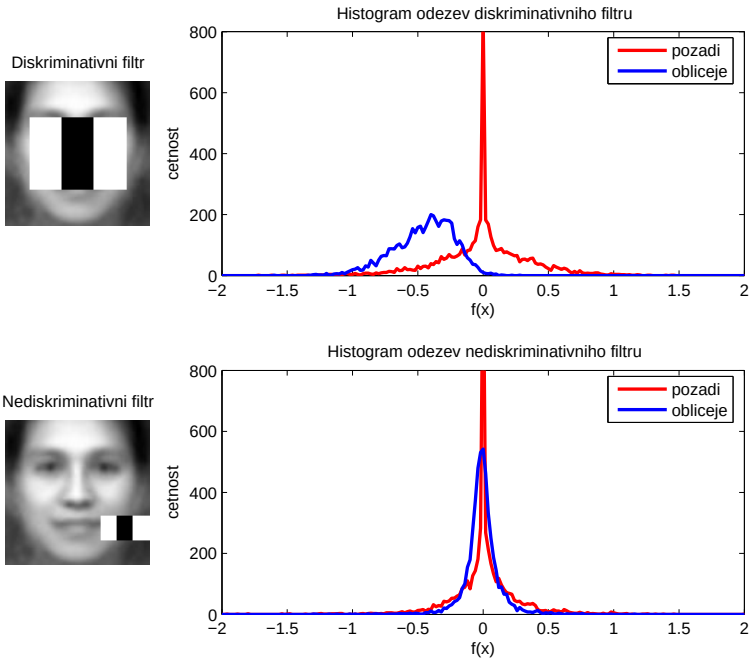


Figure 2.6: Histogram of feature response for two Haar-like features. The first feature is very discriminative for this distribution, i.e., the classes can be separated based on the response of the features. The second feature doesn't bring any information about the class for the underlying distribution. The situation can be completely different for different weights which is changing due to AdaBoost.

examples and take the one which performs the best. Reweigh the distribution so that easy examples get smaller weights and difficult examples get bigger weights. Repeat these steps till some criterion is reached.

In the setting where weak classifiers assign confidences, i.e. real AdaBoost [8], the feature selection is described by the following algorithm.

Algorithm 1 Real Adaboost for feature selection

Require: Training examples $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ where $(x_i, y_i) \in \mathcal{X} \times \{+1, -1\}$ feature set $\mathcal{F}$, number of weak classifiers T.

1: Distribution initialization, $D_1(i) = \frac{1}{m}$
2: **for** $t = 1, 2, \ldots T$ **do**
3: **for** for every feature $f \in \mathcal{F}$ **do**
4: Compute $p(f(x), y)$ and separate $\mathcal{R}(f)$ into several bins $X_1, \ldots, X_n$.
5: Under the distribution D_t calculate

$$W_+^j = P(x \in X_j, y = +1) = \sum_{i: x_i \in X_j \wedge y_i = +1} D_t(i)$$

$$W_-^j = P(x \in X_j, y = -1) = \sum_{i: x_i \in X_j \wedge y_i = -1} D_t(i)$$

.
6: Set the output of h_t on each X_J

$$\forall x \in X_j, h(x) = \frac{1}{2} log \frac{W_+^j + \varepsilon}{W_-^j + \varepsilon} = \frac{1}{2} log \frac{P(y = +1|x, D_t)}{P(y = -1|x, D_t)},$$

where ε is a small number.
7: Compute the normalization factor $Z = 2 \sum_j \sqrt{W_+^j W_-^j}$.
8: **end for**
9: Select f_t minimizing the upper bound on error, $f_t = \arg\min_{f \in \mathcal{F}} Z$.
10: Actualize the distribution

$$D_{t+1}(i) = D_t(i) \exp[-y_i h_t(x_i)]$$

and normalize D_{t+1} to p.d.f.
11: **end for**
12: resulting classifier H has the form

$$H(x) = sign(\sum_{t=1}^{T} h_t(x))$$

and the confidence is given by

$$Conf_H(x) = \Big| \sum_{t=1}^{T} h_t(x) \Big|$$

2.2.4 Threshold estimation

The Waldboost classifier makes a decision about the class if the sample reaches some predefined confidence-level given by the thresholds. This can be expressed as

$$y = \begin{cases} +1, & H_t(x) \geq \theta_B^t \\ -1, & H_t(x) \leq \theta_A^t \end{cases} \tag{2.6}$$

where θ_A^t and θ_B^t are thresholds in step t.

The thresholds are set according the Sequential Probability Ratio Test (SPRT) described earlier. It was shown in [12] that the ratio in SPRT can be estimated by

$$\hat{R}_t = \frac{p(H_t(x)|y=-1)}{p(H_t(x)|y=+1)}, \quad t \in (1,\ldots,T). \tag{2.7}$$

This equation simplifies significantly the estimation of the ratio given by equation 2.2, because it project originally T-dimensional density estimation into one-dimensional density estimation.

For now, we only need to know that the threshold is estimated according the estimated ratio defined by the equation 2.7. The correct estimation of the thresholds is one of the topics that were addressed in the thesis and hence will be discussed later on.

2.2.5 Waldboost learning

We have already discussed the to major parts of the Waldboost learning, i.e., learning of a weak classifier $h_t(x)$ with real AdaBoost and computation of thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$. The final Waldboost learning can be now easily expressed by the following algorithm.

Algorithm 2 Waldboost learning

Require: Training examples $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ where $(x_i, y_i) \in \mathcal{X} \times \{+1, -1\}$ desired final false negative rate α and final false positive rage β

1: Distribution initialization, $D_1(i) = \frac{1}{m}$
2: Set
$$A = \frac{1-\beta}{\alpha}, \quad B = \frac{\beta}{1-\alpha}$$
3: **for** $t = 1, 2, \ldots T$ **do**
4: Find the best weak classifier h_t by AdaBoost algorithm.
5: Estimate the likelihood ratio R_t according to
$$\hat{R}_m = \frac{p(H_m(x)|y=-1)}{p(H_m(x)|y=+1)}.$$
6: Find thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$
7: Throw away samples from training set which $H_t \geq \theta_B^{(t)}$ or $H_t \leq \theta_A^{(t)}$
8: Sample new data into training set
9: **end for**
10: Output: strong classifier H_t and thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$

2.3 Summary

There exist many different approaches how to address the face detection task. Our approach can be considered as a combination of feature- and appearance-based method, where a binary classifier, based on Haar-like features, is learned from a set of training examples. The order and number of these features is explicitly formulated by means of sequential decision making. The resulting classifier takes only such number of measurement that are essential to make a decision with given error of classification. In other words, the classifier is the fastest possible. The Waldboost algorithm is not limited only to the face detection task, it can be used with any other learning algorithm to boost it's performance.

Chapter 3

Algorithm Profiling

The learning of the face classifier is very time-consuming process. It is partially caused by the fact that the entire information about faces is obtained from many training images, which have to be analyzed during training. The current implementation uses about 10000 faces and about 160 mil. background samples and the time needed to train a classifier is counted in days/weeks.

The time of learning becomes an important parameter especially when new algorithms are to be developed. Slow learning significantly restrict the number of experiments that can be accomplished and hence lowers the effectiveness of the research.

The following section briefly describes the most time-consuming parts in the original implementation of the Waldboost algorithm. The main changes leading to a faster training will be introduces later on. Finally, the comparison of the original and the accelerated version will be given. A referential experiment, specified by table 3.1, is used throughout this section in order to compare the achieved acceleration.

Parameter	Value
α	0.02
β	0
Number of examples	10000
Length of classifier	50
Number of features	20000

Table 3.1: Referential experiment on which the speed up was tested.

3.1 Original version

The original code is composed of many functions written in different languages (Matlab, MEX files, C, C++). The main learning environment is implemented in Matlab but the most time-consuming parts are implemented in MEX files.

Since the learning algorithm is rather complicated a specialized tool for code profiling (Matlab Profiler) was used in order to uncover the most time-consuming parts. According to the profiler, the time is spent especially on the training data manipulation and on the search for the best weak classifier.

3.1.1 Data manipulation

The training examples are in the original version represented by their feature-responses. Every image isn't stored as a matrix of pixels but as a vector, where ith entry represents response of ith feature. For the referential experiment, there are about

$$|data| \cdot |features| = 10000 \cdot 20000 = 2 \cdot 10^8$$

values, which are stored in external files. The manipulation with these files is very time-consuming and doesn't provide an easy survey. Even though, this representation is more effective than direct manipulation with the raw images, which distribution is shown in figure 4.12.

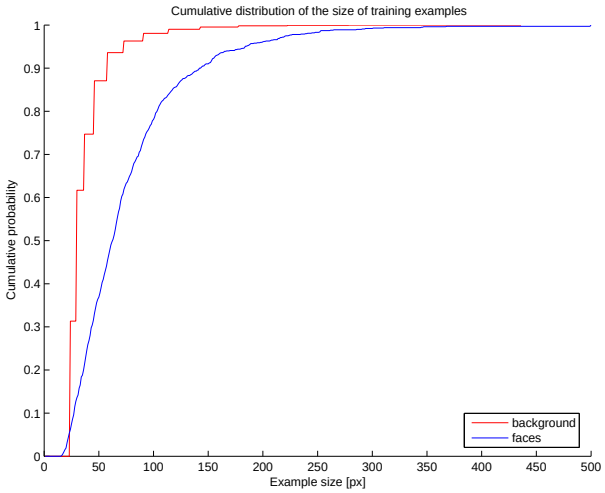


Figure 3.1: Cumulative distribution of size of training examples. The positive examples were manually collected and hence it's size is a random variable. The negative examples are obtained by scanning the images with certain step in scale-space. Such a distribution doesn't allow storing all training examples in the memory.

3.1.2 Weak classifier learning

The second, most time-consuming part, is the search for the weak classifier. The search is done by brute force, i.e., every possible classifier is evaluated and the best one is chosen. In the process of learning a weak classifier, the original algorithm uses sorting several times. In fact the sorting is done only because of the 5% and 95% quantiles, which can be computed much more effectively due to histograms.

3.2 Speed up version

The main difference between the accelerated and the original version is the way the training data are manipulated. The training examples are no more stored in the external files but directly as images in the memory. Moving the training data from hard drive to memory speed's up the learning significantly. The different way of manipulation brings other advantage, such us more flexible and more transparent code.

The problem of too big training examples was overcome by setting the upper bound on size of the training examples. Bigger examples were resized to the maximum allowed size, using bilinear interpolation. It was proven by experiments that

Original version	17.0h
Speed up version	7.0h

Table 3.2: The comparison of the original and the accelerated version on the referential experiment.

the resampling doesn't cause any loss of information. The classifiers learned by original resp. accelerated code are identical.

The next important step was optimization of the weak learner. The figure 3.2 depicts the histogram of features, which follow the Laplace distribution. This fact was exploited and the search for quantiles was implemented more effectively by histograms.

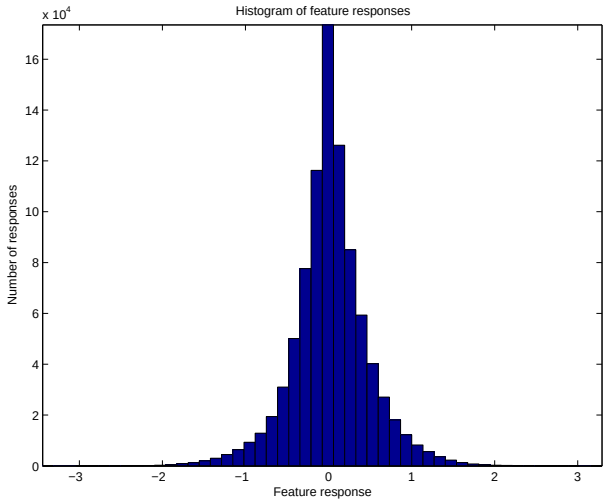


Figure 3.2: Histogram of feature responses on training data. Histograms are used for effective search for quantiles in the accelerated implementation of the Waldboost learning.

3.3 Summary

The implementation of the Waldboost learning was carefully analyzed by the Matlab Profiler, by which the main drawbacks were uncovered. In order to optimize the code, rather essential changes had to be done. The training examples were moved from the hard drive into the memory thanks to a suitable compression, which doesn't cause any loss of information in this particular case. Next, the weak learner was optimized by finding more efficient solution for quantile computation.

Except the above mentioned improvements, many other drawback were found in the code. In general, the most critical parts of the code were implemented as MEX files and the number of hard drive operation was minimized. The accelerated version is compared to the original version in the table 3.2. The resulting code is by 60% faster. The time line of the optimization can be found in the appendix and the profile reports are accessible at `http://cmp.felk.cvut.cz/~kalalz1/`

Chapter 4

Training Data Preparation

In order to demonstrate the symmetric bootstrap, sufficiently large positive training set is required. Sice the original version of Waldboost works only with fixed amount of faces, new sources of training examples are investigates in the section *Enlargement of the positive training set by face synthesis*. Except the new databases obtained from 3rd party face detectors, face synthesis is introduced, which enables arbitrarily enlargement of training set with user-defined distribution.

A new approach to bounding box alignment is suggested in the section *Unification of face databases using proper bounding box alignment*. The suggested alignment to maximal confidence enables unifying different sources of training examples and simplifies the task by finding the alignment that is the easiest for some previously learned detector.

Proper modeling of the background is discussed in the section *Background generated from parts of the face*. Such background modeling is important for correct function of the symmetric bootstrap and reduces the false positive alarms.

4.1 Enlargement of the positive training set by face synthesis

Till now a database called Schneiderman was used exclusively in the training. The main disadvantage of the database is, however, it's limited size, which doesn't allow to make experiments with symmetric bootstrap. In this chapter, different sources of training data will be discussed. The first source collects images from on-line face detector called BetaFace, the second synthesizes new faces from already seen ones. Both of the sources enlarge the set of training examples and allow demonstrating the effect of symmetric Waldboost.

4.1.1 Schneiderman database

The database Schneiderman contains 5556 frontal upright faces witch various background. The source images were collected from on-line face detector earlier published by Henry W. Schneiderman on Robotics Institute at Carnegie Mellon University.

Seven feature points were manually marked on every face. These feature points enable arbitrary bounding box to be defined. Figure 4.1 shows the average image generated from Schneiderman database for two different ways of bounding box alignment.

Number of faces	5556
Pose	frontal upright
Source of images	random selection from CMU face detector
Ground truth	left/right eye/nostril, left/right mouth corner, mouth center
Alignment	no alignment

Table 4.1: Schneiderman database details

b) alignment to eyes b) alignment to nose

Figure 4.1: Average image of Schneiderman database for two possible alignments.

4.1.2 BetaFace database

There are increasing number of on-line face detectors which might be considered as sources of new faces. The advantage of using detectors is the ease of acquiring new training examples, the disadvantage is lack of ground truth and often different way of the bounding box definition.

The database called BetaFace was acquired from on-line face detector (`www.betaface.com`). It contains about 13000 frontal upright faces without any feature points coordinates. All the faces are, however, aligned to eyes and have the same bounding box size.

Based on this, a new bounding box was defined, which requires only left and right eye coordinates. This information is implicitly known for all the detections gathered from the on-line detector.

Number of faces	13000
Pose	frontal upright
Source of images	random selection from BetaFace face detector
Ground truth	constant position of eyes, fixed size
Alignment	Fixed position of eyes

Table 4.2: BetaFace database details

Figure 4.2 shows the average image obtained from the raw BetaFace detections. The eyes of the average image are very sharp, due to the way the detections are aligned.

4.1.3 Face synthesis

Face synthesis, an alternative approach of acquiring new training images, will be discussed in this section. There are many ways how to create a new face from one already seen. The space of all synthesis is, however, restricted by the need to create only those faces that can be found in nature. Also the distribution of the synthetic

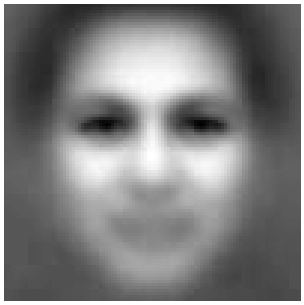


Figure 4.2: Average image obtained from BetaFace detections (faces are aligned to fixed position of eyes)

faces should follow the natural distribution.

The main motivation for the synthesis is the expansion of the training set. The synthesis can be understood, however, in different way as well. Suppose, we are able to transform every image into a normalized form (fixed rotation, scale and shift), from which new instances are synthesized according to the user-defined distribution. Since we can control the distribution of the synthetic images we can also control the scope of the final detector. In other words, the rotation-in-plane, for examples, is not given by the database distribution but it can be adjusted by the user.

In general the whole synthesis is composed of a subsequent transformations that will be described later on. Every such transformation is parameterized by one parameter which value is either estimated from the training data or directly defined by user.

Face normalization

Face normalization maps the input image x to a normalized form x_n. The subsequent transformations, which create particular instances of synthetic faces, are applied directly to the normalized form. The figure 4.3 illustrates the basic concept of normalization.

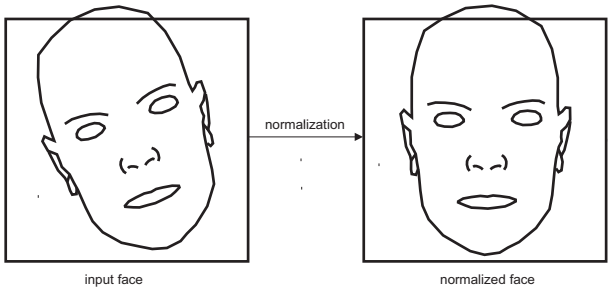


Figure 4.3: Normalization transforms the input face to the normalized form by means of geometric transform. In this case the normalized face is defined by rotation to upright position.

In general, all normalized images are distinguished from other images by certain properties. En example would be a brightness normalization, by means of linear histogram equalization. In that particular case we know for sure that all the normalized images will have the brightness values in a given interval.

Two ways of face normalization were defined and tested in face-synthesis. Both of them are strictly geometric transformations an will be described in the following

text.

First normalization is called **Affine transformation to average face**. Figure 4.4a) shows the average image from Schneiderman database and the feature points marked with yellow crosses. These feature points define a normalization grid, in which the input image is mapped by means of affine transform.

An affine transformation or affine map between two vector spaces consists of a linear transformation followed by a translation:

$$x_n = Ax + b$$

what can be in homogenous coordinates expressed as

$$\begin{bmatrix} x_n \\ 1 \end{bmatrix} = \begin{bmatrix} A & b \\ 0 \dots 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix}$$

where x_n is a vector $[2\times1]$ representing pixel-coordinates in the normalized image, x is a vector $[2 \times 1]$ representing the same pixel in the input image, A is a matrix $[2 \times 2]$ of linear map and b is $[2 \times 1]$ translation vector. The affine transformation is defined by six parameters. These parameters can be solved by SVD decomposition.

The second normalization is called **In-plane rotation to upright position**. We don't need to compute the entire affine transform but only the in-plane rotation given by the form:

$$x_n = Rx = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} x$$

where R is a rotation matrix $[2 \times 2]$. The transformation is fully defined by one parameter θ which can be easily determined from the eyes position of the input image.

Figure 4.4 compares the different approaches of face-normalization by means of an average images.

a) average image b) affine normalization c) rotation normalization

Figure 4.4: Comparison of average faces from the Schneiderman database: a) raw faces aligned to nose, yellow crosses mark the position of feature points, which define the normalization grid, b) normalized by means of affine map to average feature points, c) normalized by means of rotation in-plane to upright position. The sharper is a particular region in the average image, the smaller variance is observed in that region; more blurred parts indicate higher variance. The affine normalization reduces the variance of eyes, nose and mouth. Normalization by means of in-plane rotation reduces only variance around the eyes.

Change of identity

Every human face can be characterized by its ratio between width and height, which is not unique but relates partially with the identity. We use this relation when mapping one identity to another, as illustrated in figure 4.5.

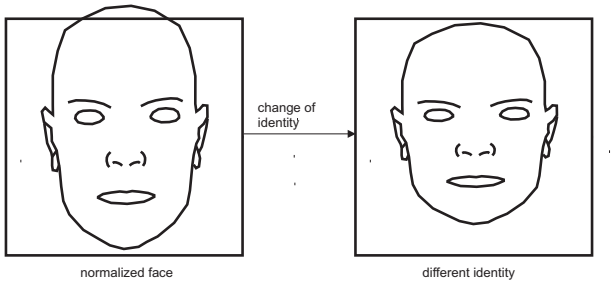


Figure 4.5: The appearance of the normalized face is mapped to a different identity by means of scale change along vertical axis.

The figure 4.6 defines the face width w and height h as used in our synthesis. We suppose that these variables have normal distribution in the real world. However, this distribution doesn't hold in case when every face is at different scale. A new scale invariant variable, has to be defined:

$$\sigma = \frac{w}{h}$$

where the random variable σ has log-normal distribution as illustrated in figure 4.7.

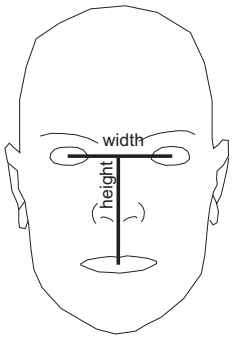


Figure 4.6: Definition of the face width and face height as used in the synthesis. These variables have normal distribution in real world.

The transformation "Change of identity" is realized as follows:

$$\begin{bmatrix} x' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{\sigma}{\sigma_0} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} = T_1 \begin{bmatrix} x \\ 1 \end{bmatrix}$$

where σ is the requested ratio and σ_0 is the ratio of the input image. As we can see, the transformation is nothing else but the change of scale along the vertical axis, which is possible due to the normalized image.

The σ parameter has either user-defined distribution, or the values are given by previously observed ratios. The identity of every face can thus be changed to another one in the sense of width/height ratio.

The figure 4.8 illustrates average faces for different values of the parameter σ.

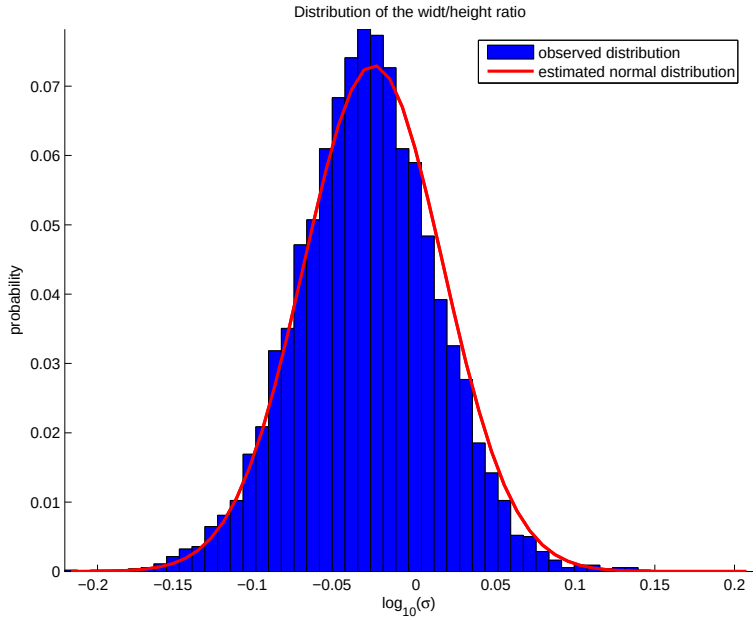


Figure 4.7: The width/height ratio is a scale-invariant random variable directly measured on training examples. The histogram fits the log-normal distribution, which estimation is marked by the red line.

Figure 4.8: The average faces for different width/height ratio given by the parameter σ. Smaller value can be interpreted as change of pose upward resp. downward and hence enable extension the range of the final detector.

Rotation in-plane (RIP)

The in-plane rotation, which was in the normalization step removed, can be in the synthesis further adjusted. Figure 4.9 illustrates the transformation.

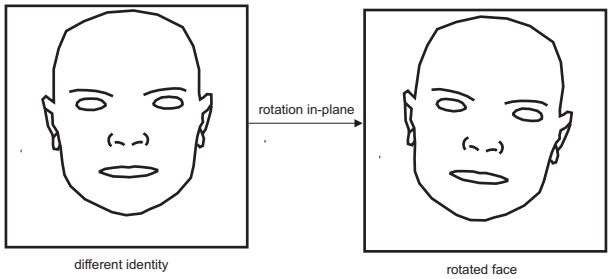


Figure 4.9: In-plane rotation

The rotation is done around the face middle. The face middle is defined by the

bounding box used. In the case of alignment to eyes, the face middle lyes between eyes. In the case of alignment to nose, the face middle lyes near the tip of the nose.

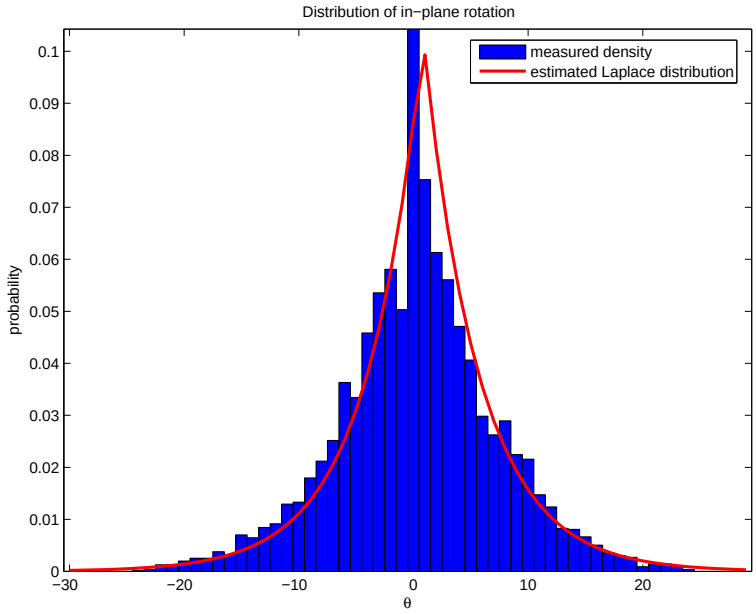


Figure 4.10: The histogram of the in-plane rotation from Schneiderman database. The parameters of the underlying Laplace distribution were estimated and the distribution was plotted by red line.

The explicit form for rotation of an image around a shifted middle can be expressed in homogenous coordinates with the following equation

$$\begin{bmatrix} x' \\ 1 \end{bmatrix} = T_2 \begin{bmatrix} x \\ 1 \end{bmatrix} = \begin{bmatrix} I & \mathbf{s} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & \mathbf{-s} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix}$$

where x' is a vector $[2 \times 1]$ representing pixel coordinates in the rotated image, x is a vector $[2 \times 1]$ representing the same pixel in the original image, R is a rotation matrix and $\mathbf{s}$ is $[2 \times 1]$ vector representing the middle of the rotation.

The distribution of the rotation obtained from the Schneiderman database is illustrated in the figure 4.10. In the synthesis the rotation can be either set according to this distribution or defined by the user according to a particular application. The figure 4.11 shows the average images corresponding to different rotation in-plane.

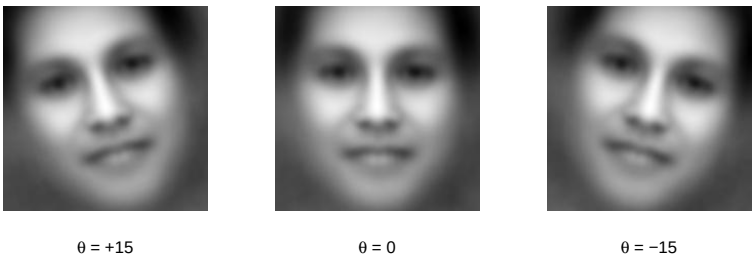


Figure 4.11: Average faces for different rotation in-plane given by the parameter θ.

Change of scale

The face detector implemented as a binary classifier uses a window-sliding technique when detecting a face. The image is scanned with different window sizes. Due to the speed issues, we don't scan the input image with every possible scale but only with a certain step in the scale-space. The figure 4.12 illustrates the sampling of the scale-space during scanning the input image.

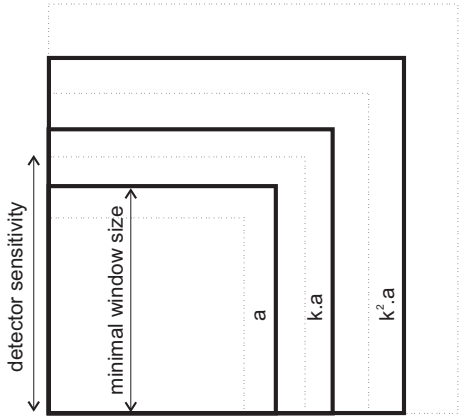


Figure 4.12: The scale-space of all sub-windows is sampled with fixed step specified by parameter k. In order not to miss any face, the detector have to be sensitive to scales between the dotted lines.

The sampling illustrated in the figure 4.12 can by written in the following equation:

$$a_i = ka_{i-1}, a_0 = a$$

where a stands for the minimal window size.

Due to the jumping in the scale-space with the step k, there is a risk of missing those faces that lie out of our grid. There are two possible approaches how to detect these faces. The first way is to decrease the step size, which leads to bigger number of windows that has to be scanned. The second approach is to learn a classifier capable of detecting faces in a specified scope of scale.

Such a classifier can be learned on suitably generated data, which can be synthesized by the transformation "Change of scale" illustrated in the figure 4.13.

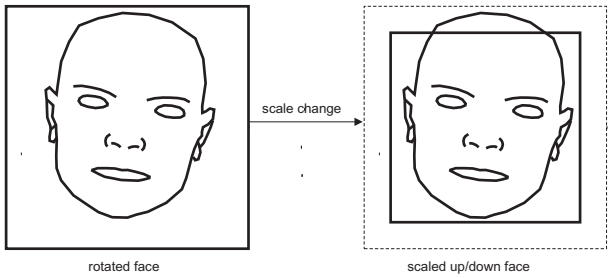


Figure 4.13: Transformation "Change of scale" scales up/down the bounding box and enables to define a area in which the final detector is invariant to scale change.

Till now an explicit mapping from feature points to bounding box was used. The transformation "Change of scale" scales up/down the bounding box according to uniformly distributed random parameter. The more random is the scale of the

bounding box, the more scale-invariant is the final detector but the more difficult becomes the task.

Bounding box shift

The face detector implemented as a binary classifier uses a window-sliding technique when detecting a face. The image is scanned by a window that moves across the image with some specified step. The shift of the window is illustrated in figure 4.14.

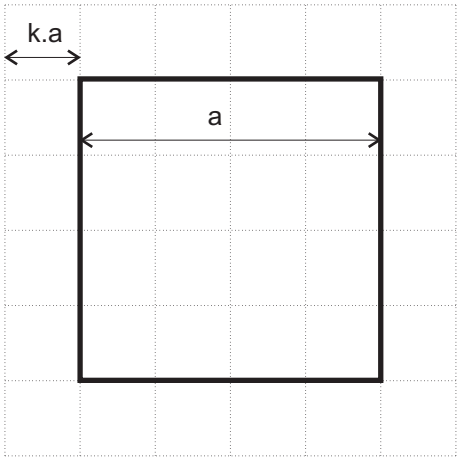


Figure 4.14: A grid on which a detector is tested. The density of the grid is given by the size of the sub-window a and the shift factor k.

The bigger is the shift factor k, the bigger is the step of the shift and the higher is the probability of missing a face occurring out of our grid. There are two possible approaches how to detect these faces. The fist is to increase the density of the grid which leads to more sub-windows that have to be tested. The second approach is to learn a classifier, which is invariant to the bounding box shift to some extend.

The transformation "Bounding box shift" enables to define such an area as illustrated in the figure 4.15.

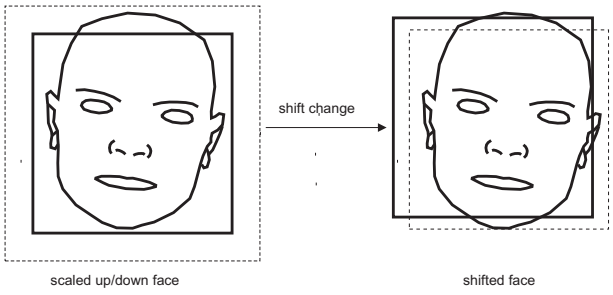


Figure 4.15: Bounding box shift is a transformation which randomly shifts the bounding box according to a random parameter and enables to define a area in which the final detector is invariant to shift.

Till now an explicit mapping from feature points to bounding box was used. The transformation "Bounding box shift" moves the bounding box according to uniformly distributed random parameter. The more random is the position of the bounding box, the more shift-invariant is the final detector but the more difficult becomes the task.

4.1.4 Summary

Three sources of positive training data were discussed in this section. The first source is the Schneiderman database and the second is the BetaFace database, which together contain more than 18.000 of frontal upright faces. The third source, synthesis, enables arbitrary enlargement of the training set.

The appearance and distribution of the synthesized faces are controlled by a couple of parameters. Every face can be mapped to another identity or arbitrarily rotated in-plane. The bounding box can be scaled up/down and shifted which is a tool to control the final database difficulty.

The synthesis was partially motivated by the need to enlarge the database Schneiderman under constraint to remain it's distribution. Such enlargement would enable demonstrating the effect of the symmetric Waldboost on frontal upright faces. Even with the most conservative setting of the synthesis, it wasn't able to increase the size and at the same time remain the original distribution. Different approaches to demonstrated the effect of symmetric Waldboost had to be considered.

Synthesis is not only a tool for generating more examples; probably even more important is its natural way to control the distribution of the training set. It seems that proper setting of the independence between training and ratio data sets has a big influence on the stopping of the algorithm during learning. This aspect will be discussed later on.

4.2 Unification of face databases using proper bounding box alignment

A bounding box defines a sub-windows of an image, in which appears some object of the interest, e.g., a human face. There are many possible ways how the bounding box can be defined. These definition differ in the shape, size and the way the face is aligned. Three different alignments will be discussed in this chapter and the experiments will show what alignment/size might be proffered.

4.2.1 Alignment to nose

The bounding box aligned to nose is defined by four feature points depicted by crosses in figure 4.16. These points determine the position of the face-center represented by the black dot. Aligning faces to this central point minimizes the overall scatter of all features in the face.

The feature points required by the bounding box definition have to be marked manually on all training examples. This step is very time consuming and doesn't allow quick increase of training data.

Pros Minimal scatter of all feature points.

Cons Manual labeling of feature points.

4.2.2 Alignment to eyes

The bounding box aligned to eyes is defined by two feature points marked by crosses in the figure 4.17. This alignment minimizes the scatter of eyes but the scatter of other feature points increases. The definition of this bounding box was partially

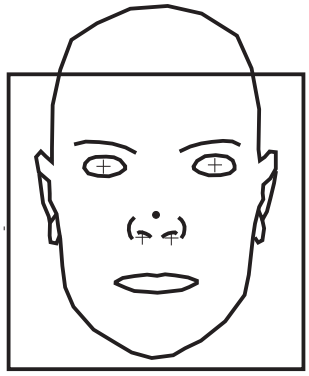


Figure 4.16: Bounding box aligned to nose requires four feature points to be defined completely. These feature points determine the face center, marked by the black dot.

motivated by the ease of acquiring large training data from on-line face detectors that use this alignment.

Pros Performs better when eyes are well defined. Ease of obtaining new training data aligned to eyes from on-line face detectors.

Cons Less robust when eyes are covered by glasses or when the image has low resolution.

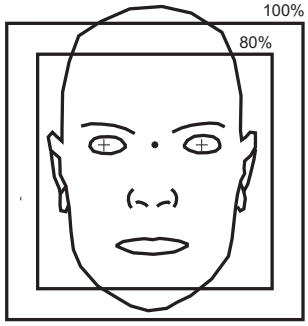


Figure 4.17: Bounding box aligned to eyes requires two feature points, marked by crosses, to be defined completely. The face-center lies between eyes and is denoted by the black dot. The 100% size refers to the definition used by on-line face detector `www.betaface.com`

4.2.3 Alignment to maximal confidence

The following bounding box combines both advantage of the previous definitions, i.e. ease of acquiring new training examples and optimally aligned face for arbitrary rotation out-of-plane.

Unlike the previous feature-points-based definitions, no features are required here. The only inputs are the image x and some previously learned classifier H.

Let M be a set of all bounding boxes that we consider as possible for given image x. Our task is to choose the one that has the highes confidence. The situation is illustrated in figure 4.18 where M is denoted by gray square. The bounding box with the highest confidence is selected according to the following equation:

$$bb_{max} = \arg\max_{bb_i \in M} H(bb_i(x))$$

where $bb_i(x)$ denotes the sub-window of image x defined by bounding box bb_i.

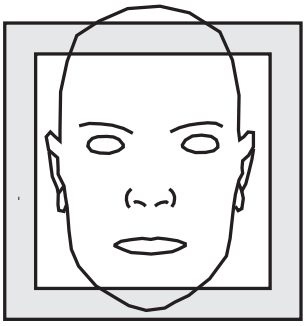


Figure 4.18: Bounding box aligned to maximal confidence. The gray sector represents the set of all possible bounding boxes from which the one with maximal confidence is selected.

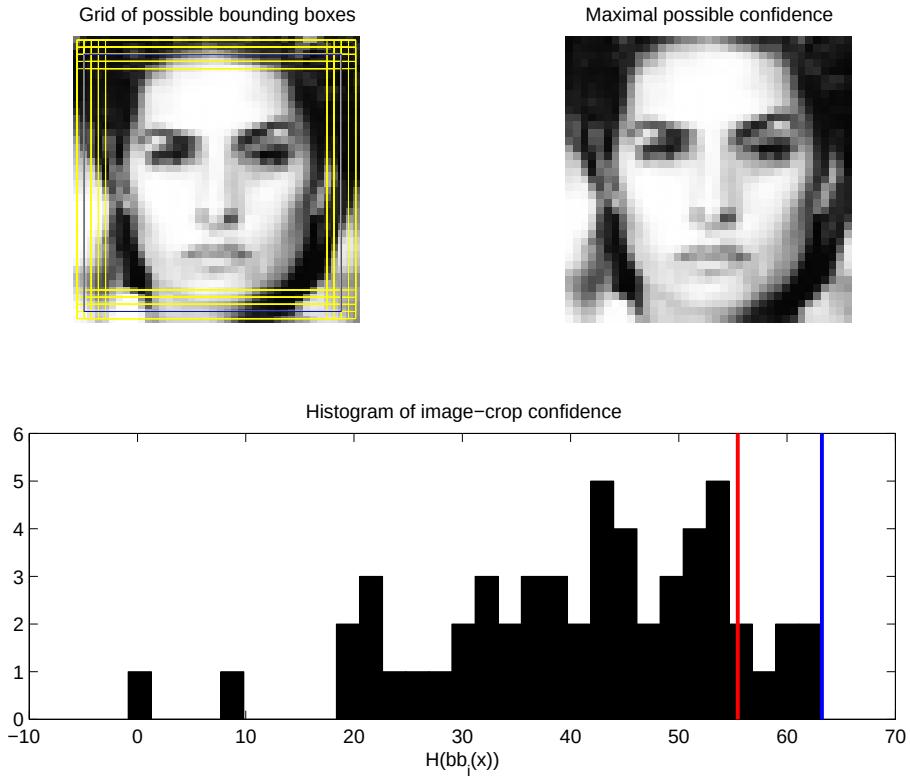


Figure 4.19: Selection of bounding box with maximal confidence. The top left subplot represents the input image from which the easiest sub-window, in the sense of some classifier, is to be selected. A set of all considered bounding boxes is marked by yellow color, the easiest one is marked by blue color. The bottom subplot depicts the histogram of confidence. The red vertical line represents the confidence of the initial image (no crop). The blue vertical line represents the selected bounding box, which sub-window is displayed in the top right corner.

The selection of the easiest sub-window, illustrated in figure 4.19, requires a

previously learned classifier. The selection depends on the way how the classifier was learned. In other words, if the classifier was learned by examples aligned to eyes it will probably prefer these bounding boxes.

The big advantage of this approach is the possibility to gather easily a huge number of training data with only a little human supervision. One possible way, how to collect new images, is to run a previously learned face detector on new images and collect true positives. This way we can collect a lot of "easy" examples that our detektor already knows.

Another approach is to use a 3rd party detectors[1] and collect detections that our detector is not able to see. This way we can easily extend the scope of our training data. The fact that we use differen definition of the bounding box than the 3rd party detector has no importance here.

Another advantage of this approach is the way of controlling the difficulty of the training set. Using maximal confidence bounding box enables us to control the difficulty by one simple parameter theta.

$$H(bb_{max}(x)) \begin{cases} \geq \theta & \text{use image} \\ < \theta & \text{don't use image} \end{cases} \tag{4.1}$$

This way we can easily remove the outliers which might cause overfitting of the AdaBoost algorithm.

4.2.4 Scaling down/up the bounding box

The scale of the bounding box is an important factor and must be considered when comparing different ways of alignment. The scale defines not only the square that is considered as face but also defines the density of features, in some cases.

In our implementation, a fixed number of features is used. By scaling down the bounding box, the smaller part of the face is covered but the denser are the features. By scaling up the bounding box, the larger surrounding with more background is covered but the features are not able to distinguish small details. This situation is illustrated on figure 4.20.

As we can see, there is a trade-off between feature density and the surrounding that can be considered in learning.

Figure 4.21 illustrates the spatial density of information on a face used by classifiers for different alignments. Most of the features are superimposed directly on the face and only a few affect the cluttered image background. This fact suggests to reduce the bounding box size and hence increase the sampling of the useful information on the face.

4.3 Background generated from parts of the face

Faces appear usually in human-made environments such as houses or streets, but it is very unlikely to find a face inside a pile of beans or on Mars surface. This information can be exploited when modeling the background. The following sections will describe the difference in contextual and non-contextual approach in background modeling and compare their performance.

[1] There is a growing number of on-line face detectors and off-line detector able to detect faces in movies

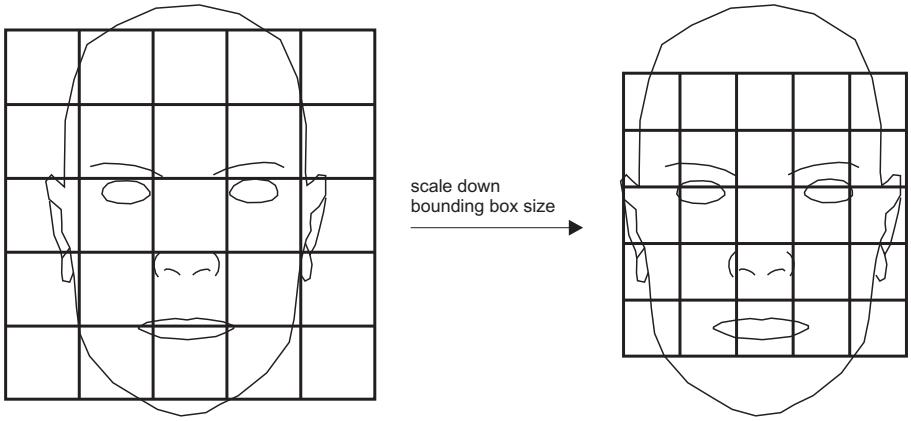


Figure 4.20: The effect of scaling down/up the bounding box on feature density. The smaller bounding box is considered the smaller surrounding is used in learning but the higher is the density of features. The bigger bounding box, the bigger is the surrounding but the sparser are the features.

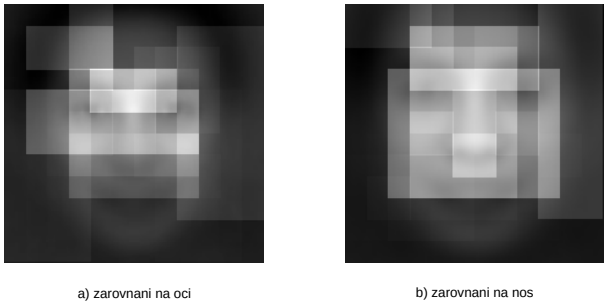


Figure 4.21: Spatial density of information exploited by the classifier for differen bounding box alignments. Two classifiers were learned under the same training conditions but different bounding alignments. The resulting features are superimposed on an average face. The brighter color, the more features cover that particular place and the more information that is used for classification.

4.3.1 Non-contextual background

In the recent approach to Waldboost learning the background was modeled by scanning random images which didn't contain any faces. These background images were gathered from the Internet and subsequently cleaned up from all images containing the human face. This corresponds to the approached published by Viola and Jones in [11].

Many of these images depict sceneries where a human face is highly unlikely to by found (microscopic images, details). Learning from these images causes the background distribution to get biased. In other words, the resulting cascade pays more/less attention to visual features in background that are less/more probable in the real world.

According to our observation, there is a big number of false positives detected directly on a face. The typical locations are illustrated in figure 4.22. This type of false positives is caused by the biased background distribution obtained from non-contextual images.

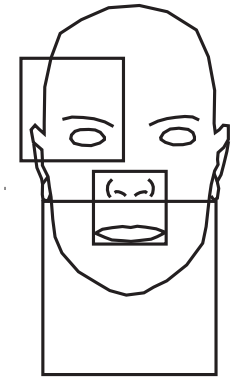


Figure 4.22: Typical false positives on a face when non-contextual background is uses in training.

4.3.2 Contextual background

A new approach of background modeling is suggested, where the contextual information is exploited. Unlike the non-contextual modeling, the true negatives are collected by scanning images that initially contained faces. This guarantees that only relevant background is used.

The face-surrounding that is removed from the images is illustrated in figure 4.23. Only the image-parts outside faces are considered in searching for background examples.

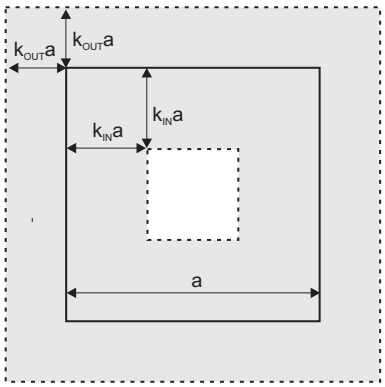


Figure 4.23: The surrounding of the face that is removed from the background images is defined by two parameters k_{IN} and k_{OUT}.

4.3.3 Summary

A simple method of modeling contextual background in face detection was suggested. The background examples are collected by scanning only those images that originally contained faces, which were removed with some specified surrounding. The size of the surrounding directly influences the difficulty of the task. In the extreme case, when only the bare faces are removed, the task may becomes too complicated to be solved. Several non-extreme settings of the surrounding-size were tested and all of them performed better, in the sense of false positive rates, than classifier learned on the non-contextual background.

Chapter 5

Bootstrapping of Positive Training Examples

The current implementation of the Waldboost algorithm constructs a sequential classifier able to make early decisions on background patterns. Positive samples have to go through the entire classifier and the final label is decided at the end by a threshold.

The main idea of this section is to make early decisions about face-patterns too. We call this approach symmetric bootstrap. The following sections are going to describe the particular steps leading to the symmetric Waldboost algorithm.

5.1 Distribution of example-weights during learning

The distribution of training examples used in Waldboost changes over time due to Adaboost and bootstrapping. Adaboost focuses on difficult examples by increasing their weights and ignores easy examples which become smaller weighted. Bootstrapping replaces easy examples with appropriately difficult ones. This proces is not symmetric in the original implementation of Waldboost, i.e., the negative examples are bootstrapped but the positive examples are not.

The effect of bootstrap on distribution of negative examples is illustrated in figure 5.1. In the first iteration of training, all the weights are equal, as the learning continues more and more attention is payed on difficult examples by increasing their weights. This proces is, however, restricted by the bootstrap which removes too easy examples and replaces them with more difficult ones. That's why the distribution of the positive training samples is kept almost equal.

On the other hand, the positive examples are not bootstrapped, their weights are changed with Adaboost only. As the learning continues, smaller and smaller portions of all the faces represent the distribution, as illustrated in figure 5.2. And hence only a few examples actively participate on selection of the weak classifier, which selection is then more random.

If we manage to sequentially replace easy faces with accurately difficult ones, we should be abe to explore the space of positive examples more effectively. In other words we would focus on positive examples not only by the means of Adaboost weights but also by proper bootstrapping. The side effect of the bootstrap would be an early decision about faces which don't need to go up to the end of the cascade. The experiments with weights suggest that proper bootstrapping of faces is a promising direction.

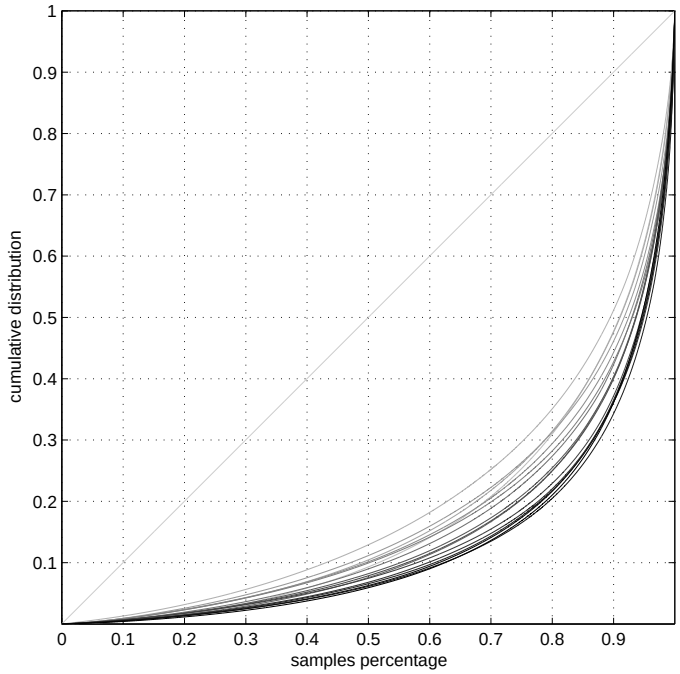


Figure 5.1: Cumulative distribution of negative examples (background) during first 80 iterations. The distribution changes due to Adaboost and bootstrap. The darker line, the higher iteration.

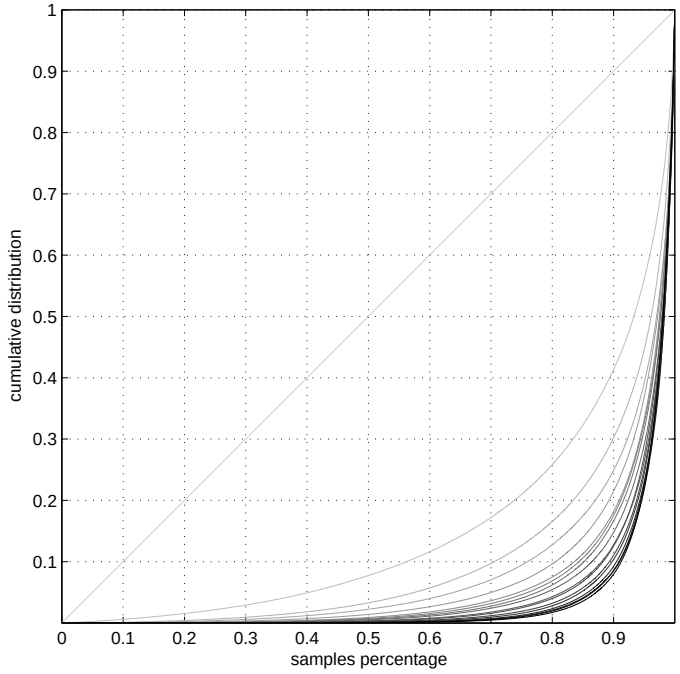


Figure 5.2: Cumulative distribution of positive examples (faces) during first 80 iterations. The distribution changes only due to Adaboost. The darker line, the higher iteration.

5.2 Threshold estimation

The symmetric Waldboost classifier makes a decision about the class if the sample reaches some predefined confidence-level given by the thresholds θ_A and θ_B, as

expressed by equation 2.6.

Only the threshold θ_A, which enables background-classification, was estimated in the non-symmetric implementation. The second threshold θ_B was set to infinity and hence no sample was classified as a face before it got at the end of the classifier. In the following, we are going to discuss the process of searching for the threshold θ_B.

5.2.1 Estimation of θ_B

The correct thresholds for sequential classification are set according to SPRT [13], as expressed by the equation **??**. The essential role in SPRT plays the computation of the ratio of classes. It was shown in [12] that the ratio can be estimated by

$$\hat{R}_t = \frac{p(H_t(x)|y=-1)}{p(H_t(x)|y=+1)}, \quad t \in (1,\ldots,T).$$

This form isn't, however, used in the implementation of the algorithm due to the stability issues. Two different ratios are computed from the cumulative conditional probabilities of classes. The ratio for estimation of the threshold θ_A has the form

$$R_A(\theta) = \frac{p(H(x) < \theta|y=-1)}{p(H(x) < \theta|y=+1)}, \tag{5.1}$$

and the ratio for estimation of the threshold θ_B

$$R_B(\theta) = \frac{p(H(x) > \theta|y=-1)}{p(H(x) > \theta|y=+1)}, \tag{5.2}$$

where $p(H(x) < \theta|y)$ denotes the cumulative conditional probability computed from "left" and $p(H(x) > \theta|y)$ cumulative conditional probability computed from "right". The thresholds are then estimated by

$$\theta_A = \arg\max_{\theta} \{R_A(\theta) > A\} \tag{5.3}$$

$$\theta_B = \arg\min_{\theta} \{R_B(\theta) < B\} \tag{5.4}$$

The figure 5.3 illustrates the estimation of the thresholds.

It seems that the estimation of the θ_B is only a symmetric analogy to the estimation of the θ_A. This would be truth if the a priory probabilities of classes and allowed errors of classification would be at least similar. In the face detection task, however, it holds that

$$P(y=-1) \gg P(y=+1).$$

A typical setting of the classification errors are expressed by the table 5.2.1.

$\alpha = 0.02$	2% of faces can be rejected in training
$\beta = 10^{-6}$	0.0001% of background can be classified as faces

Table 5.1: Typical setting of the classification errors

Since we allow only 0.0001% of background to be classified as faces, we need to set the threshold θ_B very accurately. As shown earlier, in order to estimate the threshold, some cumulative probabilities have to be estimated. These probabilities are very small, and hence their accurate estimation on a fixed set is a bit problematic.

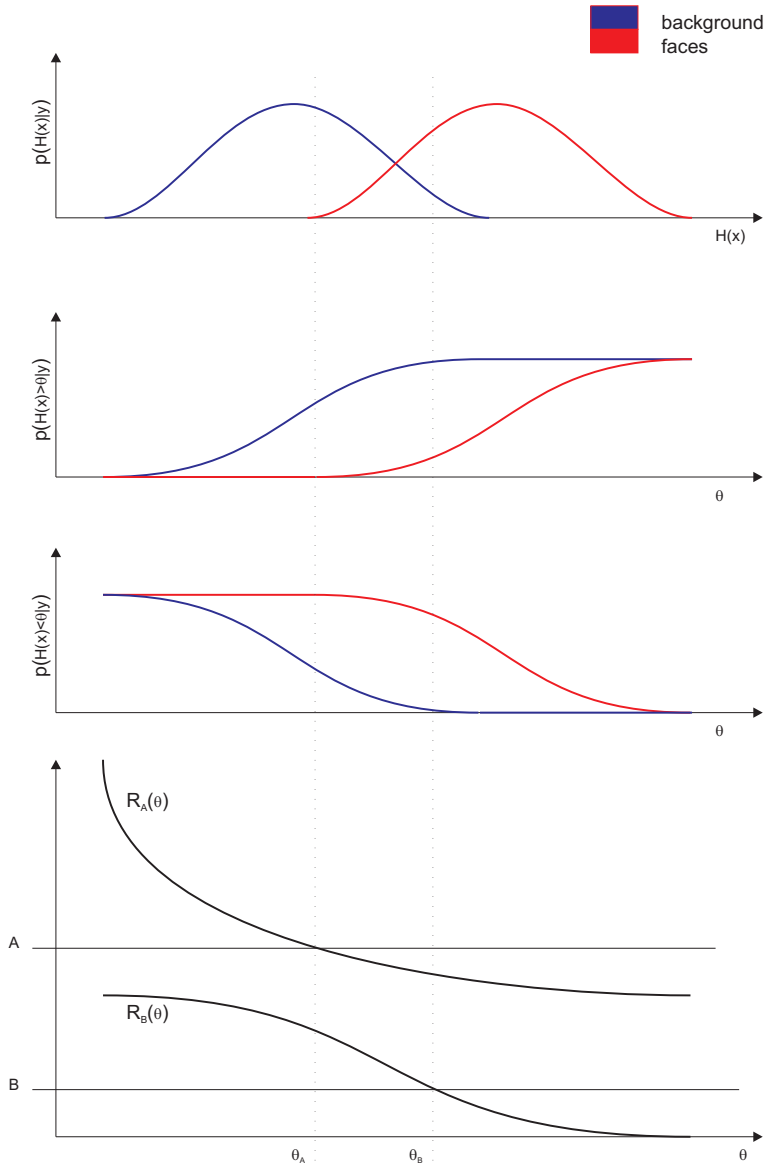


Figure 5.3: Estimation of the thresholds. The top most figure denotes the conditional probabilities of the classes, from which the cumulative distributions can be computed as illustrated in the second and third figure. The ratios estimated according to the equations 5.1 and 5.2 are illustrated in the last figure. The points A and B are given by the SPRT. The thresholds estimated according to the equations 5.3 and 5.4 are represented by the dotted vertical lines.

In the following subsection, the exact form of the relative error can be expressed. The form will help us to understand why the estimation of the thresholds can't be treated as completely symmetric.

5.2.2 Relative error of the estimated threshold

Suppose we are estimating a unknown probability p from n yes/no trials. The probability p is estimated according to

$$\hat{p} = \frac{k}{n},$$

where k is the number of "yes" observations. In general, the random variable k follows the binomial distribution

$$k \sim B(n,p), \quad \mu_k = np, \quad \sigma_k^2 = np(1-p),$$

and hence the $\hat{p}$ is also a random variable with

$$\mu_{\hat{p}} = p, \quad \sigma_{\hat{p}}^2 = \frac{np(1-p)}{n^2}.$$

The probability p is estimated with the error

$$\sigma_{\hat{p}} = \sqrt{\frac{p(1-p)}{n}}.$$

and the relative error of the estimate

$$\sigma_{\hat{p}}^{rel} = \frac{\sqrt{\frac{p(1-p)}{n}}}{p} = \sqrt{\frac{1-p}{pn}}.$$

From the equation follows that the relative error of the probability estimation depends on the estimated probability and number of trials. If the probability p is high enough (i.e., $p \sim 0.02$) we don't need that much trials to estimate is relatively accurate. This is the case of estimating θ_A. If, however, the estimated probability is very small (i.e., $p \sim 10^{-6}$) we need much more trials, otherwise the relative error is too high.

The error of the ratio can be expressed as follows.

$$R_A(\theta) = \frac{p(H(x) < \theta | y = -1)}{p(H(x) < \theta | y = +1)} = \frac{P_1}{P_2}$$

The gradient of the ratio has the form

$$\nabla R_A = (\frac{1}{P_2}, -\frac{P_1}{P_2^2})^T$$

and the error of the ratio can be expressed as

$$\sigma_{R_A} = \frac{1}{P_2}\sigma_{P_1} + \frac{P_1}{P_2^2}\sigma_{P_2}$$

where σ_{P_1} resp. σ_{P_2} are the errors of estimation the probabilities P_1 resp. P_2. Analogous equations hold for $R_B(\theta)$.

The problem with the high relative error of ratio estimation appeared only when the threshold θ_B was estimated. The early versions of the symmetric classifiers didn't hold the required errors of classification, i.e., they produced too many false positives. Two possible solutions were suggested and implemented:

Increase of the set on which the ratio is computed In the original implementation, all of the available training samples were separated equally to training set (TS) and validation set (VS). Where the size of TS and VS were equal (~ 5000). The threshold estimation was done on the VS. The number of samples in VS is absolutely insufficient for accurate θ_B estimation and hence it's size was increased. In the current implementation, the number of negative samples in set to 20M, all the available faces participate on the ratio estimation. It was shown by experiment that increasing further the size of negative samples doesn't bring much effect.

Bayesian estimate of the probabilities The ratio R_B is estimated conservatively. Partially the probabilities in the ratio are estimated as

$$p = \frac{k+1}{n+2}.$$

The conservative estimate of the θ_B significantly reduces the number of false positive alarms.

5.3 Summary

In this section, the generalization of the original Waldboost implementation, partially the search for the threshold θ_B, was discussed. The threshold θ_B was first estimated the same way as the threshold θ_A. This approach was soon rejected due to the high relative error of estimation, which was caused by small set on which the estimation was done.

The error of the estimation was explicitly expressed and two possible solutions were suggested and implemented. First, the size of the set on which the ratio is computed was increased. Second, the ratio R_B is computed conservatively.

It was shown by experiments that we can't use absolutely symmetric approach of treating the negative and positive samples. This is caused by the fact that the prior probabilities are very different. The conservative estimation of the threshold θ_B on a bigger set significantly reduces the false positive alarms. The correct estimation of the threshold is also connected with contextual and non-contextual background modeling.

Chapter 6

Pose Estimation with Linear Regression

The Waldboost face detector is a binary pattern-classifier. That is, the content of a given part of an image is transformed into features, after which a classifier decides whether that particular region of the image is a face, or not.

The features are selected according their ability to separate the background from faces. Suppose that the features are also correlated with some other useful information such as head pose. Under this assumption the aim of this chapter can be formulated as follows: Using the feature responses, estimate the head pose with linear regression. The computation of the pose would be very efficient, and could help to increase the robustness of the face detractor.

The following sections will describe the basic concepts of linear regression, data collection, implementation and the final experiments.

6.1 Linear regression

Linear regression is a regression method of modeling the conditional expected value of one variable y given the values of some other variable or variables x. Both of the variables can be scalar or vector. Linear regression is called "linear" because the relation of the response to the explanatory variables is assumed to be a linear function of some parameters.

The regression model has the form

$$y_i = \alpha + \beta x_i + \varepsilon_i.$$

By recognizing that the regression model is a system of linear equations we can express the model using data matrix X, target vector Y and parameter vector δ. The ith row of X and Y will contain the x and y value for the ith data sample. Then the model can be written as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \alpha & \beta \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

which when using pure matrix notation becomes

$$Y = X\delta + \varepsilon,$$

where ε is normally distributed with expected value 0 (i.e., a column vector of 0s) and variance $\sigma^2 I_n$, where I_n is the $n \times n$ identity matrix. The estimated parameters δ can be obtained by least squares

$$\widehat{\delta} = (X'X)^{-1} X'Y \tag{6.1}$$

where X' is the transpose of X.

6.2 Implementation

In our task, the measurement is a vector of feature responses denoted by

$$x = [f_1, f_2, \ldots, f_T],$$

where f_i represents ith feature response. The estimated variable is the rotation of the head denoted by y. For simplicity reasons, only left to right rotation, illustrated by figure 6.1, will be considered here.

Figure 6.1: The head pose considered in the estimation is limited to left-right rotation in the range ±20 degrees. The range is limited by the detection possibilities of the classifier.

6.2.1 Data preparation

The training data were acquired from the databases UMIST and PIE [10], which contain face images with varying head rotation. The database UMIST contains sequences of faces, with rotate out-of-plane in the range from profile to frontal with step about 2 degrees. This database doesn't contain any ground truth information about the head orientation. Thus the outside positions were estimated manually and the intermediate points were set by linear interpolation.

Database PIE contains ground truth data about the face orientation. The face rotation in the database covers the interval from left to right profile with fixed step 22.5 degrees.

Since the detector is sensitive to faces with rotation out-of-plane in the range ±25, we could only choose a certain portion of all faces that were available. About 500 raw examples were chosen from both databases and the number were doubled by mirroring. The training example were aligned to maximum confidence.

6.2.2 Parameter estimation

The data acquired in the previous step can be directly substituted to the equation (6.1). The resulting model would consider all features measured on the face. Such a model is, however, too complicated and it's generalization is poor. A feature analysis had to be done in order to reduce the number of features used in the model.

According to the accomplished experiments, the most of the features is not correlated with the head rotation. This corresponds with the basic function of the features, i.e., detect faces and not estimate the head pose.

The figure 6.2 illustrates typical features which are not correlated with the head rotation. Such features doesn't bring any useful information and their response is rather a source of inaccuracy and over-fitting.

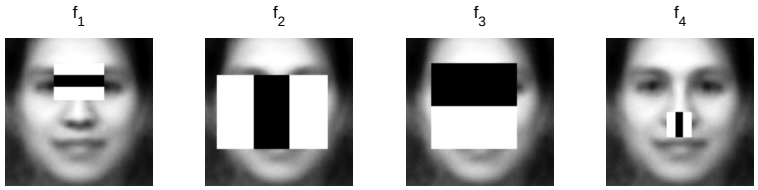


Figure 6.2: Example of features which are not correlated with head rotation.

In order to simplify the final model, a minimal threshold θ on correlation between the feature response and head rotation was defined. All the features considered in the simplified model have to meet the following equation:

$$corr(y, f_i) > \theta$$

The figure 6.3 illustrates the features that are correlated with the pose more than some threshold θ, i.e., contain some information about head rotation. These features were subsequently used for parameter estimation.

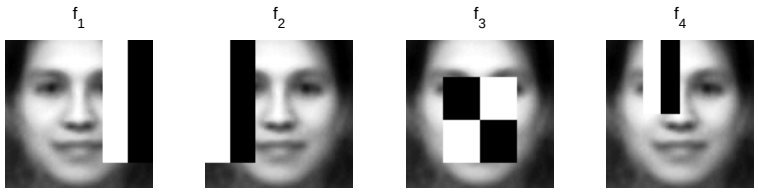


Figure 6.3: Example of features which are correlated with the head rotation.

The final pruned model contains about 5 features, it is easier and more generalizing. The scatter plot 6.4 shows the error of the pose estimation on training and validation set.

6.3 Summary

A simple linear method for pose estimation based on feature responses was proposed. The figure 6.5 illustrates the output from the detector + pose estimator. The accuracy of the method is about 8 degrees and the range is ± 25 degrees. The estimation is very time-efficient, because it only adds up the weighted responses of already computed features.

The main problem is the relatively high error. It is caused by the lack of enough accurately labeled examples. The databases, which were used in training suffered from two kinds of problems. The first is the inaccuracy of the ground truth and the second is the constant background. Precise estimation would require larger database with cluttered background and with accurately labeled head rotations, which is unfortunately not available.

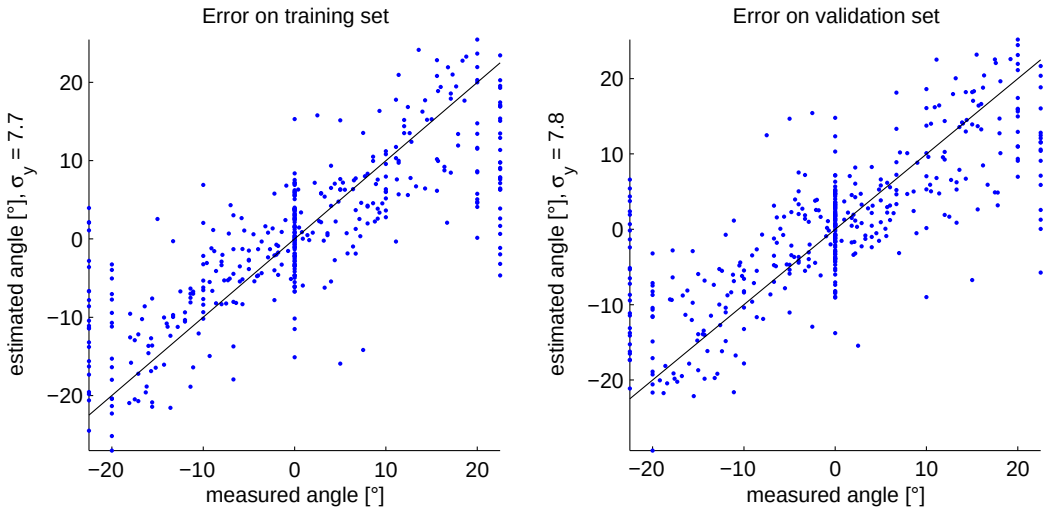


Figure 6.4: Error of angle estimation on training resp. validation set

Figure 6.5: The output from the detector + pose estimator.

The main contribution of this section can be stated as follows. It was shown that features contain not only information about the class but also other useful information can be exploited. This is a promising way for further research in the context of fast sequential detection. One possible improvement based on this idea, is to exploit the whole sequence of features (not only one at the time) that are measured and based on their probabilities classify difficult samples. An algorithm proposal is enclosed in the appendix.

Chapter 7

Further Work

The effect of the symmetric bootstrap is in the current Waldboost implementation probably limited by the discriminating power of the weak features used. After some time of learning, we are not able to find any feature that lowers the upper bound of error on both training- and ratio- set. This problem was addressed in several papers, in which more discriminant features are proposed, e.g. [14] [6]. The further research should be aimed in developing such stronger features that are able to separate the classes even in later stages of the training.

Except the stronger features, other further steps in the research can be considered. It was shown that the feature-responses can be correlated with some other useful information such as head pose. This observation is promising for further research in fast sequential decision-making. The algorithm *Classification based on the sequence of features* aims to exploit the entire sequence of features for classification, not one at the time. The second algorithm suggested in this chapter generalizes discrete on-line Adaboost to real on-line Adaboost.

7.1 Classification based on the sequence of features

In the current implementation of the Waldboost weak classifier h_i determine the response of the strong classifier according to the following equation:

$$H_T(x) = \sum_{i=1}^{T} h_i(x)$$

Suppose that there exist some typical sequences of feature responses (path) for faces and background. Figure 7.1 illustrates the situation.

The probability of the path can by under certain conditions written in the following form:

$$P(f_1(x), f_2(x), \ldots, f_T(x)|y) = P(\tilde{x}|y) = P(f_1(x)|y)P(f_2(x)|f_1(x), y) \ldots P(f_T(x)|f_{T-1}(x), y),$$

where the probabilities $P(f_1(x)|y)$ and $P(f_i(x)|f_{i-1}(x), y)$ can be easily obtained during Waldboost learning. So it is possible to enumerate the probability $P(\tilde{x}|+1)$ and $P(\tilde{x}|-1)$.

The response of the strong classifier can be than adjusted using the information gained out of the path as follows

$$H'_T(x) = H_T(x) + \frac{1}{2}\log\frac{P(\tilde{x}|+1)}{P(\tilde{x}|-1)} + \frac{1}{2}\log\frac{P(+1)}{P(-1)}$$

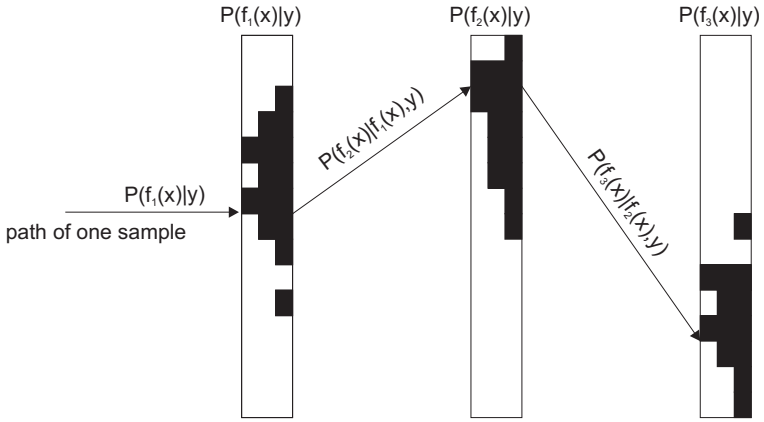


Figure 7.1: One possible path through three feature responses.

As the learning continues, the faces and background examples are less separable by simple Haar-like features and the learning slows down. The faces and background seems identical after the projection of the strong classifier. The samples can, however, differ in the way how they reached the final value. If there exist some typical paths for faces, it would be easy to eliminate false positives by analyzing their paths.

7.2 On-line real Adaboost

On-line discrete AdaBoost algorithm is suggested by Oza in [7]. This approach is used by Grabner in [2] for selector selection which enable on-line updates of the weak classifiers.

The following algorithm aims to generalize the algorithm proposed in [2] for real AdaBoost. The main difference is the way of actualizing the weights λ and introduction of new variables $\lambda^{corr}_{n,m,j}, \lambda^{wrong}_{n,m,j}$ which contain information about correctly/wrongly classified examples in every bin of the weak classifier.

Algorithm 3 On-line Real AdaBoost

Require: training example $\langle x, y\rangle, y \in \{-1, 1\}$
strong classifier $H(x) = \sum_{n=1}^{N} h_n^{sel}(x)$
pool of weak classifiers $h_{n,m}(x) = \frac{1}{2}\log\frac{P(+1|x,w)}{P(-1|x,w)}$ with K bins and corresponding bin-weights of correctly/wrongly classified examples seen so far $\lambda_{n,m,j}^{corr}, \lambda_{n,m,j}^{wrong}$ (initialized with 1)

$\lambda = 1$
for $n = 1, 2, \ldots N$ **do**
  for $m = 1, 2, \ldots M$ **do**
    $h_{n,m} = update(h_{n,m}, \langle x, y\rangle, \lambda)$
    if $y = sign(h_{n,m}(x))$ **then**
      $\lambda_{n,m,j}^{corr} \leftarrow \lambda_{n,m,j}^{corr} + \lambda$
    else
      $\lambda_{n,m,j}^{wrong} \leftarrow \lambda_{n,m,j}^{wrong} + \lambda$
    end if
    $\lambda_{n,m}^{corr} = \sum_{j=1}^{K} \lambda_{n,m,j}^{corr}$
    $\lambda_{n,m}^{wrong} = \sum_{j=1}^{K} \lambda_{n,m,j}^{wrong}$
  end for
  $h_n^{sel} = \arg\min_{h_{n,m}} \frac{\lambda_{n,m}^{wrong}}{\lambda_{n,m}^{wrong} + \lambda_{n,m}^{corr}}$
  $\lambda \leftarrow \lambda \exp(-y h_n^{sel}(x))$
end for

The weak classifier has the form

$$h(x) = \frac{1}{2}\log\frac{P(+1|x,w)}{P(-1|x,w)} = \frac{1}{2}\log\frac{P(x|+1,w)}{P(x|-1,w)} - T$$

where the distributions can be modeled by:

$$P(x|+1,w) = N(\mu^+, \sigma^+), P(x|-1,w) = N(\mu^-, \sigma^-)$$

The parameters $\mu^+, \mu^-, \sigma^+, \sigma^-$ can be sequentially estimated due to Kalman filter.

Chapter 8

Experiments

Many experiments were accomplished during the work. The complete list can be found at `cmp.felk.cvut.cz/~kalalz1` and a portion of them is listed in the Appendix. In this section, only the most important experiments will be introduced. In the first experiment, a comparison of symmetric and non-symmetric bootstrap will be given. The second shows a different way of understanding the ratio set in the Waldboost training. The third compares different alignment and sizes of bounding boxes. And the forth discusses the contextual and non-contextual background modeling.

8.1 Comparison of symmetric and non-symmetric bootstrap

Several experiments demonstrating the symmetric bootstrap were done, which structure is illustrated in figure 8.1. Suppose, we have a huge number of faces that can be used in the Waldboost training. The number doesn't allow processing all of the training data at once, i.e., take one half to training set and the second half to ratio set. Only some portion (smaller than half) is taken into the training set and the rest is left for ratio estimation.

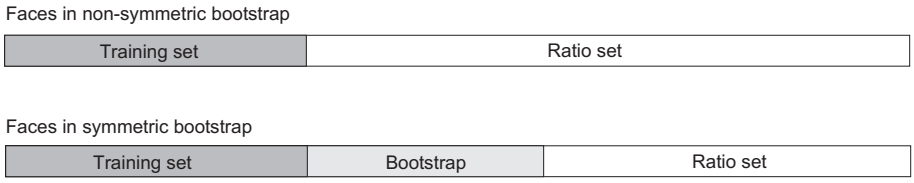


Figure 8.1: Structure of the experiments. A set of positive training examples is separated into training set (TS) and ratio set (RS). The size of the positive training set is in the non-symmetric Waldboost fixed, but the symmetric version explores more training examples by bootstrapping.

The non-symmetric bootstrap selects the best weak classifier on the fixed training set which effective size decreases during learning due to Adaboost weights. On the other hand, the symmetric bootstrap actualizes the training set by bootstrap and hence bigger space of faces participate on training weak classifier.

The details of this experiment are listed in the table 8.1. Positive examples were generated from Schneiderman database by synthesis (41.000, shift of bounding box, mirror) and mixed with the BetaFace database (26.000, mirror). The resulting set

was separated into training set (20.000) and ratio set (47.000).

Positive examples	Synthetic Schneiderman (shift, no RIP, mirror), BetaFace (upright, mirror), Total=67.000
Negative examples	Contextual background from Schneiderman images, Total=250.000.000
TS size	20.000
α	0.02
β	$0\|10^{-7}$

Table 8.1: Experiment parameters

The non-symmetric Waldboost selected the weak classifiers on the fixed training set of size 20.000, on the other hand, symmetric bootstrap explored more than 49.000 of positive examples during training. The effect of bigger training set can be seen o the ROC curve shown in the figure 8.2.

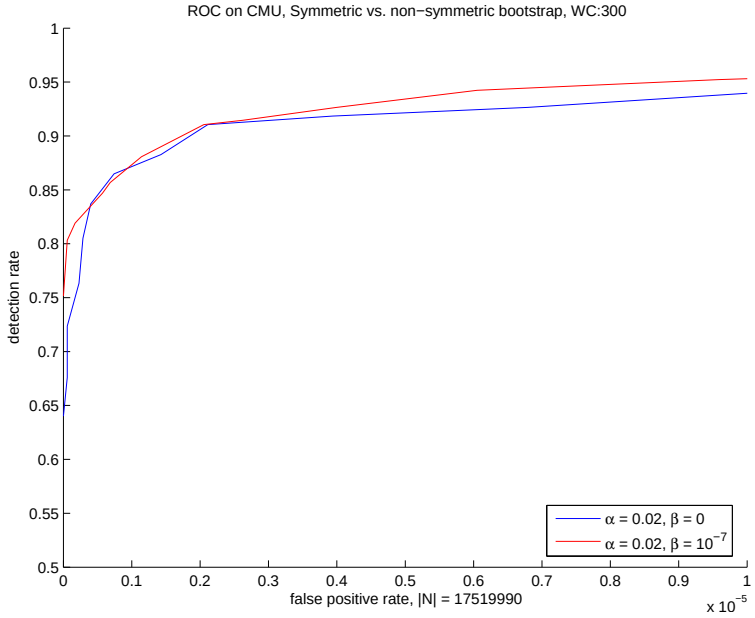


Figure 8.2: Comparison of symmetric and non-symmetric bootstrap in Waldboost.

This experiment shows the effect of symmetric bootstrap. By exploring bigger space of positive examples, the quality of the final classifier increases.

8.2 Ratio estimation on slightly shifted training set

In the original approach to Waldboost training, the set of all available positive examples is separated into training set (TS) and ratio set (RS). Using independent sets for training and ratio estimation is essential for keeping the errors of classification in the specified limits. The drawback of this approach is that only a half of all possible faces is used for weak classifier selection.

According to previous experiments, the upper bound on error on training set Z_{TS} decreases exponentially during training, but the upper bound on error on ratio set Z_{RS} start to increase around 100-200th step. The increase of Z_{RS} causes the

learning to slow down, particulary bootstrapping of negative patterns. In can be interpreted that the sets are too independent in the later iterations.

By splitting the original set of faces into TS and RS, we guarantee the errors of classification to be in the limits, but we loose the control about the independence of the sets. The independence can be, however, controlled by synthesis.

Suppose we put all of the faces available into the TS and crate the RS by shifting slightly the examples in TS. The shift of the examples is a random variable with some distribution that controls the independence of training and ratio set. Using this approach, all of the faces can actively participate on selection of the weak classifier and on the other hand, their shifted versions set the thresholds for classification.

The question remains, if the shift of the training examples guarantee that the final classifier follows the required errors. In the following experiment the whole Schneiderman database was used in the TS. The RS was generated by shifting the TS by a random variable bb_{shift} having uniform distribution in some specified range that was tested. Table 8.2 shows the details of the experiment.

Positive examples	Synthetic Schneiderman (shift, no RIP, mirror)
Negative examples	non-Contextual background
TS size	10.000
α	0.02
β	0
bb_{shift}	U(-1,1)px, U(0,0)px

Table 8.2: Experiment parameters

In the first run, the shift of the RS was set to zero, i.e., the TS was equal to RS up to the negative samples. The errors of this classifier are higher than allowed. In the second run, the RS was shifted by 1px. This classifier follows the specified errors. The achieved results on the CMU database are listed in the table 8.2.

#WC	Experiment	FN	TN	Speed
100	$bb_{shift} = 0px$	8.8	99.97118	4.2
150		10.3	99.99303	4.2
500		-	-	-
100	$bb_{shift} = 1px$	2.7	99.36511	5.7
150		2.7	99.52345	6.0
500		2.9	99.78943	6.9

Table 8.3: Comparison of classifiers learned on slightly shifted training set, where the shift-coefficient is given by parameter bb_{shift}. FN and TP refer to the false negative and true positive rates tested on the CMU database.

It was shown that the RS can be generated by shifting the training set. This approach has two advantages compared to the original understanding of learning. First, all of the positive training examples are used for weak classifier selection. Second, the independence of the TS and RS can be controlled by one simple parameter bb_{shift}. The question remains, how this parameter should be set, in order to achieve the best results possible.

8.3 Bounding box alignment and scale

A set of classifiers were learned on a fixed training data set, using the same learning parameters but different bounding box definition. Three different alignment in different scales were tested. The final classifiers were compared due to ROC curve on an independent databases (CMU, MS).

In the first experiment, different bounding box sizes with the same alignment were tested. Figure 8.3 shows the resulting ROC curve.

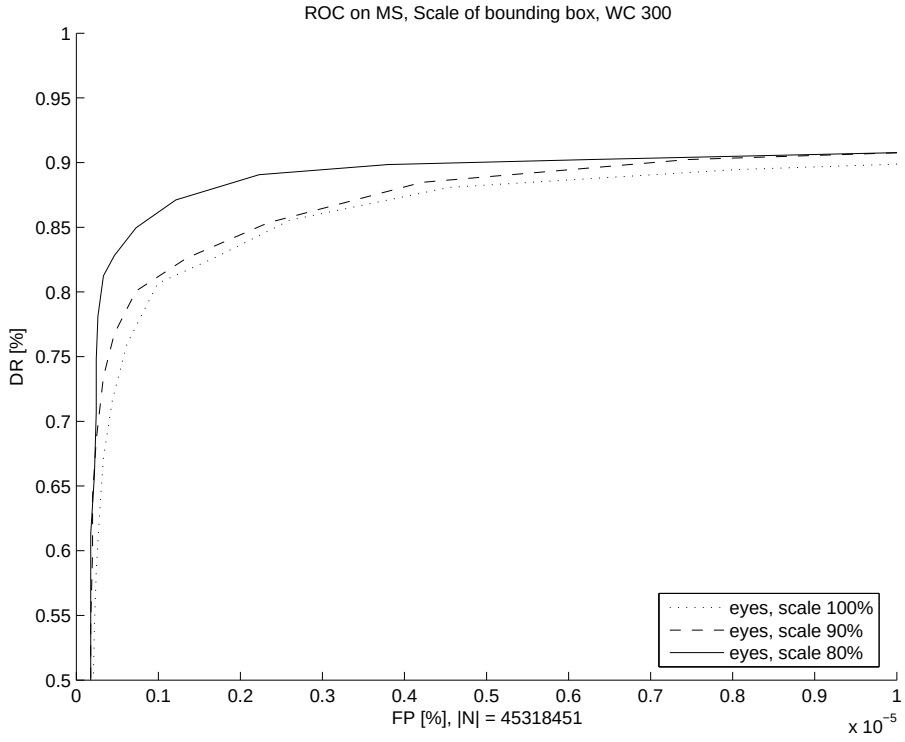


Figure 8.3: Influence of the bounding box size to the classifier performance.

The second experiment compares different alignment subject to a given bounding box size. Figure 8.4 depicts the resulting ROC curve.

In the case when the number of features is limited, the performance of the classifier increases with reducing the bounding box size. The information gained out of a face by denser sampling is higher than the information obtained by sparser sampling of a bigger image-region. This conclusion doesn't hold always. In the case of low resolution images, there is not much information that can be obtained by denser sampling and hence this approach fails. Even with this limitation, smaller bounding boxes seem to perform better for all types of alignments.

The test on bounding box alignment doesn't show any important difference between the alignment to eyes and alignment to nose when identical size is considered. On the other hand the alignment to maximal confidence significantly shifts the ROC curve upward for both cascade-alignments. The shift of the ROC curve can be explained by it's construction. A hit is considered, when at least one true positive is detected in some surrounding of the face. By aligning the training examples to maximal confidence, we force the classifier to detect only the easiest sub-window in the surrounding. This approach, however, results in more false negatives when tested on image crops. In other words, when the window-sliding technique is considered, the alignment to maximal confidence increases the performance of the classifier.

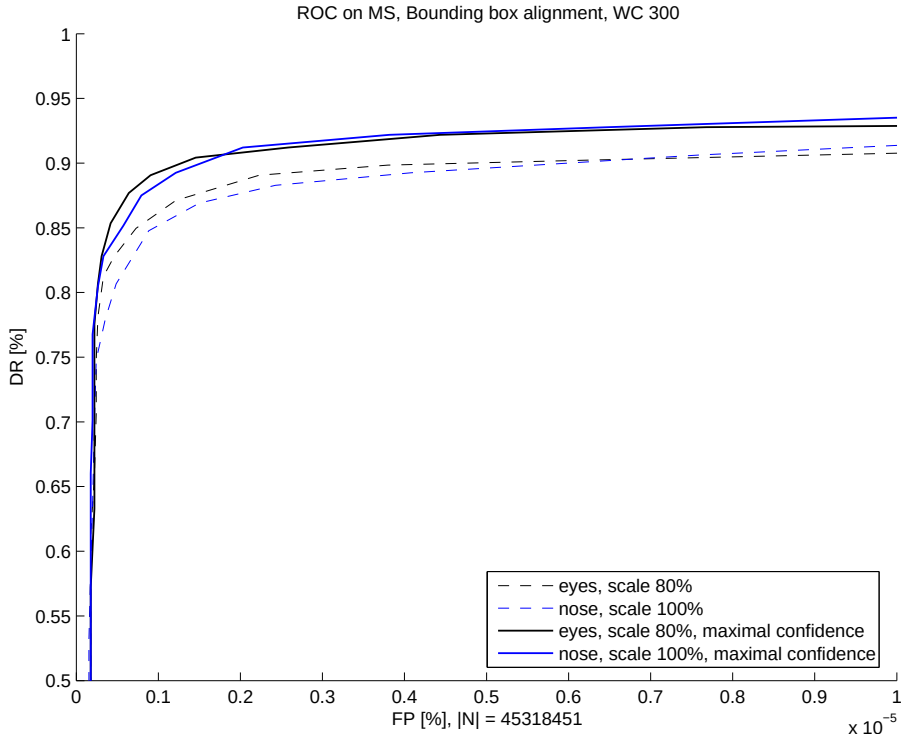


Figure 8.4: Influence of the bounding box alignment to the classifier performance.

8.4 Contextual vs. non-contextual background

Several classifiers differing only in the way how background examples were collected were trained and compared on a CMU database. The resulting ROC curves are shown in figure 8.5.

The contextual background reduces the number of false positives. Proper setting of the surrounding that has to be removedness from the source-images is important.

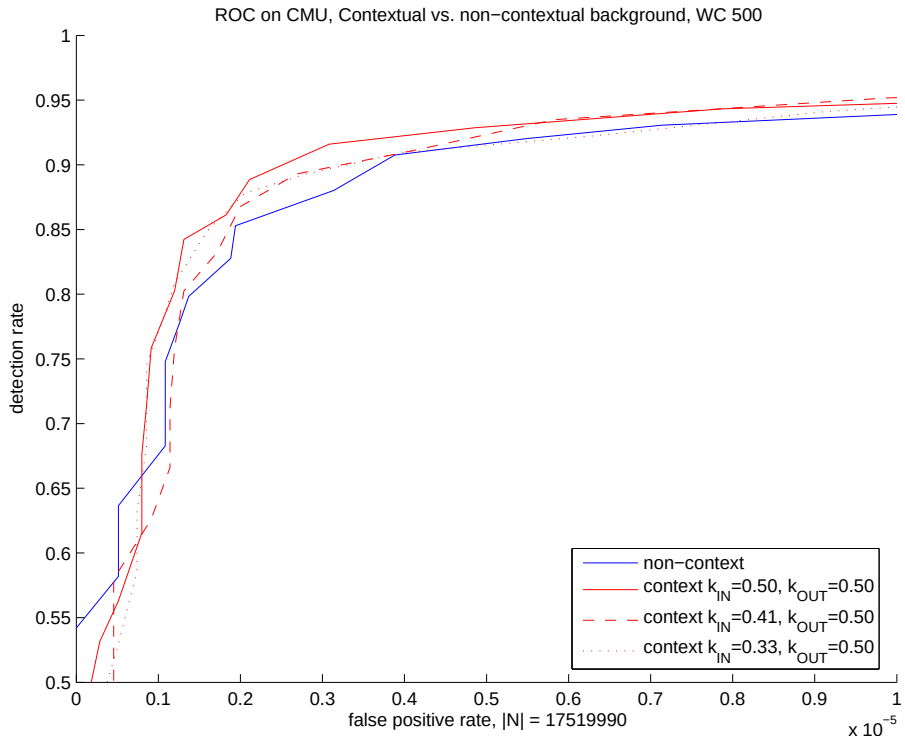


Figure 8.5: Comparison of contextual and non-contextual background modeling.

Chapter 9

Conclusions

In the chapter *Algorithm Profiling* the original algorithm was analyzed and accelerated. In order to optimize the code, rather essential changes had to be done. The training examples were moved from the hard drive into the memory thanks to a suitable compression, which does not cause any loss of information in this particular case. Next, the weak learner was optimized by finding more efficient solution for quantile computation. The most critical parts of the code were implemented as MEX files and the number of hard drive operation was minimized. The resulting code is by 60% faster.

In order to demonstrate the effect of symmetric bootstrap, the problem of lack of positive examples was addressed in chapter *Training Data Preparation*. Two new sources of training images were found. First, a database collected from on-line face detector (BetaFace) with more than 13.000 of frontal upright faces. Second, a face synthesis which enables arbitrary enlargement of the training set. The appearance and distribution of the synthesized faces are controlled by a couple of parameters. Every face can be mapped to another identity or arbitrarily rotated in-plane. The bounding box can be scaled up/down and shifted which is a tool to control the final database difficulty. Synthesis is not only a tool for generating more examples; probably even more important is its natural way to control the distribution of the training set.

The performance of the classifier depends largely on the bounding box that defines the face. In the case when the size of the feature-pool is fixed and these features are uniformly distributed on area under bounding box, the performance of the classifier increases with reducing the bounding box size, i.e. experiment *Bounding box alignment and scale*. It seems like the information gained out of a face by denser sampling is higher than the information obtained by sparser sampling of a bigger face-region.

A new type of bounding box alignment which does not require any labeled feature points, i.e. alignment to maximal confidence was proposed. This alignment enables easy collection of new training data from on-line face detectors and simplifies the training significantly. It was shown in the experiment *Bounding box alignment and scale* that the classifiers trained with examples aligned to maximal confidence have higher detection rates if tested by means of ROC curve. By aligning the training examples to maximal confidence, we force the classifier to detect only the easiest sub-window in the face's surrounding. This approach leads naturally in more false negatives if tested on cropped faces.

A simple method of modeling contextual background in face detection was suggested. The background examples are collected by scanning only those images that

originally contained faces, which were removed with some specified surrounding. It was shown in the experiment *Contextual vs. non-contextual background* that the classifiers trained with contextual background have lower false positive rates.

The sufficiently large positive training set and correctly modeled non-face examples are the prerequisites for the main part of this thesis, i.e. bootstrap on positive examples which we call symmetric bootstrap. In the section *Estimation of* θ_B it was shown that we can not use absolutely symmetric approach of treating the negative and positive samples, particularly in the way the thresholds are estimated. Contrary to the non-symmetric bootstrap, conservative estimation of the thresholds on larger data set is essential for keeping the required false positive alarms in required limits.

Many experiments were performed directed to show the difference between symmetric and non-symmetric bootstrap. The results can be stated as follows:

- The symmetric bootstrap increase the performance of the classifier if the number of faces used for feature selection is relatively low ($\sim$1000) and these faces are bootstrapped. As the number of training faces increases (>1000), the effect of symmetric bootstrap decreases and it is becoming more and more difficult to correctly interpret the results due to a the randomness of the experiments.

- It wasn't observed that replacing already classified faces with different instances generated by synthesis would lead to a better classifier as reported in the experiments *Learning without VS, Blowup effect* enclosed in the Appendix.

- The upper bound on error on testing set decreases up to certain point and then grows. This break point, which in fact represents the possibilities of the weak learner, is reached by symmetric bootstrap sooner than in the non-symmetric version as reported in *Symmetric and non-symmetric bootstrap* enclosed in the Appendix.

The last point of the above-mentioned list suggests that the effect of the symmetric bootstrap is degraded by too weak features. In the further research stronger and more generalizing features should be considered in later stages of learning.

Except the main research line leading to the symmetric bootstrap, other investigations were done. It was shown that the ratio set can be generated by shifting the training set as reported in the experiment *Ratio estimation on slightly shifted training set.* This approach has two advantages compared to the original understanding of learning. First, all of the positive training examples are used for weak classifier selection. Second, the independence of the training set and ratio set can be controlled by one simple parameter.

A simple linear method for pose estimation based on feature responses was proposed in the chapter *Pose Estimation with Linear Regression.* It was shown that features contain not only information about the class but also other useful information can be exploited. This might be considered as a promising direction for further research in the context of fast sequential detection-making.

Bibliography

[1] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[2] Grabner Helmut and Bischof Horst. On-line boosting and vision. 2006.

[3] Kiyoto Ichikawa, Takeshi Mita, and Osamu Hori. Component-based robust face detection using adaboost and decision tree. *fgr*, 0:413–420, 2006.

[4] M. Jones and P. Viola. Fast multi-view face detection, 2003.

[5] Stan Z. Li, Long Zhu, ZhenQiu Zhang, Andrew Blake, HongJiang Zhang, and Harry Shum. Statistical learning of multi-view face detection. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 67–81, London, UK, 2002. Springer-Verlag.

[6] Takeshi Mita, Toshimitsu Kaneko, and Osamu Hori. Joint haar-like features for face detection. *iccv*, 2:1619–1626, 2005.

[7] N. Oza and S. Russell. Online bagging and boosting. In *Artificial Intelligence and Statistics 2001*, pages 105–112. Morgan Kaufmann, 2001.

[8] Robert E. Schapire and Yoram Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[9] H. Schneiderman. A statistical approach to 3d object detection applied to faces and cars, 2000.

[10] Terence Sim, Simon Baker, and Maan Bsat. The cmu pose, illumination, and expression database. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1615 – 1618, December 2003.

[11] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features, 2001.

[12] Jan Šochman and Jiří Matas. Waldboost - learning for time constrained sequential detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 150–157, Los Alamitos, USA, June 2005. IEEE Computer Society.

[13] Abraham Wald. In *Sequential Analysis*, Dover, New York, 1947.

[14] Peng Wang and Qiang Ji. Learn discriminant features for multi-view face and eye detection. *cvpr*, 1:373–379, 2005.

Appendix A

Versions of the Code and Performed Experiments

More than 130 experiments were accomplished, and a row of improvements done during the research. The complete reports can be found in HTML form at `cmp.felk.cvut.cz/~kalalz1` with additional materials such as profile reports and demo videos.

Since it is not convenient to list all of the reports in printed form, only a portion of them is included in this appendix. The order of these experiments corresponds to the steps done during research. The corresponding versions of the code are listed in the following table.

Appendix B

Enclosed CD

Enclosed CD contains PDF version of the thesis, complete HTML reports from the performed experiments and the actual source code of the symmetric Waldboost version 3.6.1.

Symmetric Waldboost versions

The following table lists the versions of symmetric Waldboost that were developed during the work.

Version	Description
SB1.0	Initial implementation which enables symmetric classifier, thresholds are estimated on VS, Symmetric implementation of abFindFace_multiview, abClassEvalOnWin
SB1.2	Errors corrected: FP detections evaluated in every iteration, ev of each detection recomputed before inserting to data.X
SB1.3	A pool variable is introduce, from which TS|VS are sampled randomly. WC is selected on TS, ratio is computed on VS. TP resp. FN are replaced from the pool. The main disadvantage is the lack of enough positive examples.
SB1.4	The lack of positive examples solved by synthesis from Schneiderman database (affine transformation, in plane rotation, bb shift, bb scale). At the beginning TS and VS are generated as in SB1.3, new examples are added by synthesis. Version is not stable.
SB1.5	Ratio estimated on VS, new variables pTP, pFP, pTN, pFN are introduce which represent the history of the learning. Still not stable because pFN and pTP become higher than allowed by alpha and beta
SB1.6	A concept of BlowUp was introduced. Ratio estimated directly from all examples available in bootstrap (10M background, BlouUp times enlarged positive examples). The VS is still implemented but isn't used.
SB1.6.1	Scott rule for optimal bin-width in ratio estimation, 5556 pos images used instead of 5000, windows with std < 10 are added into the detections_P in the first run (unify with previous implementation which uses these windows)
SB1.6.2	Synthetic images based on observed parameters
SB1.7	Ratio estimated on all available negative examples (about 10M), and faces from VS
SB1.7.1	5556 pos images used instead of 5000 -> training and ratio data sets are from the same distribution, windows with std < 10 are added into the detections_P in the first run, Scott rule for optimal bin-width in ratio estimation, ratio displayed as cum sum and saved directly to public_html/exp_name directory - FN, TP are not updated in bootstrap (gen_pos_samples), this leads to stability of the algorithm - FN, TP estimated from both train and ratio dataset's - can be another source of instability for beta > 0 - error in findThetaAB corrected
SB1.7.2	Synthetic images based on observed parameters, synthetic images generated this way are easier - ratio estimated on both TS and VS; FN and TP NOT updated in bootstrap
SB1.7.3	Ratio estimated on both TS and VS; FN and TP updated in bootstrap
SB1.7.4	Ratio estimated on VS; FN and TP updated in bootstrap
SB1.7.5	Ratio estimated on VS; FN and TP NOT updated in bootstrap
SB2.0	No skew in synthetic images - data generated before training, RS is created by synthetic images, original images are used in TS
SB2.1	TS uses only some images available, synthetic images from all available images in RS
SB2.2	Detections removed if added to data, training data are in three states: detections, data, decided
SB2.3	Bootstrap will not be done if we don't have enough data, i.e., |detections.idxs| > |TS|
SB3.0	Contextual background, maximum confidence bbox
SB3.1	Ratio computation on TS, ROC test for original resp. B=Inf cascade
SB3.2	Generalization of the P/N a priory probabilities
SB3.3	Resampling of TS when the error on TS is smaller than some threshold. When new data are placed in TS they are evaluated by the current classifier.
SB3.4	Enables testing eye/nose bbox alignment, contextual/non contextual background
SB3.5	Z_PTS observed in a figure
SB3.6	CMU GT generated from original GT, Used 503 from 507 faces, only unlabeled faces and "ufo" removed Resampling REMOVED, VS contains shifted copy from TS, one Z_TS, Z_VS diagram in log, synthesis of BetaFace the same way as Schneiderman, similarity of TS and VS controlled by synthesis
SB3.6.1	Enables processing positive training examples coming without any GT

Initial experiments with symmetric bootstrap

The first implementation of symmetric bootstrap is tested in the following experiments. Schneiderman database is used exclusively in the training. Testing of the final classifier is the same as in the non-symmetric version, i.e., it doesn't allow to check correctly the FPs and TPs. It is expected however, that the speed of the classifier will increase when the symmetric bootstrap is used.

#exp	#wc	Train parameters	FN (%)		FP (%)			E[#wc/win]			Learning time	Details
			A	B	C	D	E	C	D	E		
11	50	060913_SB1.6_a0.02_b10-5_BlowUp2_noskew	5	5	18.3	17.5	20.6	18.5	19	20.5		
10	50	060914_SB1.6_a0.02_b10-5_BlowUp10	13	16	7.6	6.3	7.9	11	11	12		log profile
9	50	060913_SB1.6_a0.02_b0_BlowUp5	9	30	13.6	13	15.8	13.6	13.9	15.2		
8	50	060906_SB1.5_a0.02_b0_NoHitachi	18	113	1.94	1.61	2.3	5.3	5.2	5.7		
7	50	060906_SB1.4_a0.02_b10-4_NoHitachi	12	44	7.9	7.2	9.5	8.8	8.8	9.5		log
	50	060906_SB1.4_a0.02_b0_NoHitachi	11	37	2.1	2.0	2.9	5.5	5.7	6.3		log
6	50	060905_SB1.4_a0.02-0.05_b10-4_NoHitachi	24	129	0.97	0.7	1.1	5.1	5.1	5.4	11.5h	log
5	50	060905_SB1.4_a0.02_b10-3_NoHitachi	11	33	5.94	5.4	6.9	7.6	7.6	8.4	22.3h	log
	50	060905_SB1.4_a0.02_b10-3	11	22	5.75	4.9	6.4	7.7	7.7	8.5	25h	log
4	50	060828_SB1.3 α:0.02; β:0.00001; PTS:5000, NTS:5000; Pool: HS5398+mirror+Hit;	15	16	3.04	456925 (2.68)	685207 (3.93)	6.38	6.45	7.37	4h45m	
3	50	060828_SB1.3 PTS:5000, NTS:5000; Pool: HS5398+mirror+Hit;	18	20	167 (1.67)	284767 (1.67)	436250 (2.51)	5.65	5.80	6.45	11h	log profile
2	50	060825_SB1.2_b0_a0.02_HS5398+mirror+Hit TS/VS:HS5398+mirror+Hit; POOL: FN added from 'HS_frontal_5398_VS.txt'	15 (3.05)	15 (2.34)	232 (2.32)	340568 (2.00)	497965 (2.86)	5.69	5.77	6.37	12h30m	log
1	50	060821_SB1.0_b0_a0.02_HS5398+mirror+Hit TS/VS:HS5398+mirror+Hit; POOL: FN added from 'HS_frontal_5398_VS.txt'	15 (3.05)	15 (2.34)	238 (2.38)	374036 (2.20)	514080 (2.96)	5.80	5.82	6.29	13h	log

Conclusions

- Hitachi database removed from training, because it's distribution doesn't correspond with larger database Schneiderman.
- A concept of one Pool from which TS and VS are randomly sampled was proposed.
- Faces can't be synthesized on demand, the estimates of pTP and pFN change significantly during such synthesis and disable further threshold search.
- A new concept of ratio estimation was proposed: Estimate the ration on all data available in the bootstrap (10M background, all possible faces). Such an estimation is stable because we don't add any new positive examples which can cause the instability.
- **Any attempt to boost the performance of the classifier by symmetric bootstrap fails.** The resulting classifier is slower and at the same level doesn't throw that much TN as the non-symmetric one.

ID	Data set	# faces	# non-faces	File	Comment
A	MIT+CMU	491	0	GT file	Originally 507 faces but some partially out of image
B	Hitachi 640	640	0	GT file	
C	Non-skin 10,000	0	10000	GT file	
D	500 non face images	0	20,177,169	file list	All windows in 500 non face images (i.e. window scanning with parameters: min_win_size: 24x24, row/col_shift_factor: 2/24, scale_factor: 1.25, no FineSeekOut, no NMS, no SVM)
E	130 degraded CMU images	0	21,263,707	file list	All windows from modified dataset A (each face is upside down). Window scanning with parameters: min_win_size: 24x24, row/col_shift_factor: 2/24, scale_factor: 1.25, no FineSeekOut, no NMS, no SVM

Ratio estimated on Pool/VS/combination

Compare three different approaches to ratio estimation.
OLD - estimates the ratio on VS, 5k bootstrapped negative and 5k original positive examples
SB1.6 - estimates the ratio on all data available
SB1.7 - estimates the ratio on all negative examples available and bootstrapped positive VS
New testing data sets are introduces: Synthetic, MatejSmid. The classifier is tested as a binary typical classifier (four kinds of error + speed). No ROC curve is used in the testing.

#	wc	parameter	FN (%)				TP (%)				Speed P E(#wc/win)				FP (-)			TN (%)			Speed N E(#wc/win)			Details
			A	B	C	D	A	B	C	D	A	B	C	D	E	F	G	E	F	G	E	F	G	
30		061012_**SB1.6.2**_a0.02_**b10-5_BU1** cmpgrid-67	HYP: 30, 29 SHOULDN'T CHANGE (MISSTAKE) RES: OVERFITTING EXP: BU1, MOST OF FACES COPIED FROM DETECTIONS TO DATA DURING BOOTSTRAP -> OVERFITTING																					
	100		7.1	4.1	30.2	9.3	72.1	80.4	32.8	66.7	39.4	34.8	56.3	46.7	0	154	315	100.0	100.0	99.9	5.2	5.1	5.5	
	50		5.5	3.4	21.2	6.4	64.6	71.0	24.5	55.2	26.9	25.2	33.9	31.7	0	127	252	99.1	99.5	99.2	5.0	5.0	5.3	
29	100	061012_**SB1.6.2**_a0.02_**b10-5_BU10** cmpgrid-66	REF16: SYNTHETIC ON OBSERVED,NO SHIFT, NO SCALE vs. SYNTHETIC ON ESTIMATED DISTRIBUTION / FN, /TP, / SPEED P, =FP, /TN, /SPEED N RESULTS: SYNTHETIC IMAGES WITH OBSERVED PARAMETERS ARE EASIER																					
			2.9	1.7	16.2	4.1	71.3	80.3	29.1	65.6	41.6	36.4	65.1	50.3	0	145	281	98.6	99.0	98.4	7.7	7.3	8.2	
	50		2.6	1.7	15.8	4.0	65.0	71.8	23.8	54.6	27.7	25.8	36.4	33.2	0	123	226	96.7	97.3	96.4	6.6	6.5	7.0	
28		061007_**SB1.7.5**_a0.02_**b10-5** cmpgrid-67	REF27: / TN, / SPEED N, / SPEED P, - FN , REASON: BETTER WC FOUND DUE TO BOOTSTRAP ON FACES																					
	100		3.9	2.4	14.5	4.8	73.7	80.8	31.4	66.7	40.2	35.8	64.0	49.8	0	140	362	99.1	99.2	98.7	6.7	6.5	7.4	log
	50		3.5	2.3	13.9	4.6	65.2	72.1	26.2	54.4	26.9	25.6	35.8	32.9	0	113	277	97.8	98.1	97.3	6.0	5.9	6.5	
27		061007_**SB1.7.5**_a0.02_**b0** cmpgrid-65	RATIO ON VS; FN, TP NOT UPDATED IN BOOSTSTRAP -> STABLE																					
	100		3.5	2.0	14.1	4.2	0.0	0.0	0.0	0.0	96.8	98.1	87.2	96.2	0	0	0	98.0	98.4	97.6	7.7	7.4	8.5	log
	50		3.5	2.0	13.6	4.1	0.0	0.0	0.0	0.0	48.6	49.1	44.1	48.2	0	0	0	96.8	97.3	96.1	6.5	6.4	7.1	
26		061006_**SB1.7.4**_a0.02_**b0** cmpgrid-65	NOT STABLE																					
	100		3.1	1.6	13.9	4.4	0.0	0.0	0.0	0.0	97.3	98.5	87.2	96.1	0	0	0	98.6	98.8	98.1	7.6	7.3	8.3	log
	50		2.9	1.6	13.8	4.2	0.0	0.0	0.0	0.0	48.8	49.3	44.1	48.3	0	0	0	97.4	97.8	96.9	6.7	6.5	7.1	
25	50	061006_**SB1.7.4**_a0.02_**b10-5**	NOT STABLE																					log
24	50	061006_**SB1.7.3**_a0.02_**b10-5**	NOT STABLE																					log
23	100	061005_**SB1.7.2**_a0.02_**b10-5** cmpgrid-67	4.3	2.1	16.6	5.5	72.1	83.3	37.7	67.1	42.4	35.0	59.0	49.2	0	164	373	99.1	99.3	98.9	6.0	5.7	6.5	log
	50		3.9	2.1	16.2	5.2	62.1	72.8	30.8	54.3	28.2	25.5	34.7	32.8	0	142	304	98.1	98.5	97.8	5.3	5.2	5.7	

22	100	061005_**SB1.7.2**_a0.02_**b0** cmpgrid-66	4.3	2.1	19.2	5.6	0.0	0.0	0.0	0.0	96.5	98.1	83.7	95.1	0	0	0	99.1	99.3	98.9	6.2	5.9	6.7	log
	50		3.7	2.0	18.0	5.2	0.0	0.0	0.0	0.0	48.5	49.1	43.2	47.8	0	0	0	97.8	98.3	97.5	5.6	5.4	5.9	
21	100	061005_**OLD**_a0.02_b0_synthetic2 cmpgrid-65	3.1	1.4	12.8	4.0	0.0	0.0	0.0	0.0	97.2	98.7	88.3	96.4	0	0	0	97.9	98.4	97.6	8.3	7.8	8.8	
	50		3.1	1.4	12.3	3.9	0.0	0.0	0.0	0.0	48.7	49.4	44.6	48.4	0	0	0	96.5	97.3	96.3	6.9	6.7	7.3	
20	450	061002_**SB1.7.1**_a0.02_**b10-6** cmpgrid-67	3.5	0.8	8.6	2.0	70.3	81.2	34.5	71.0	167.6	140.0	292.2	180.0	0	13	117	99.1	99.4	98.9	12.8	11.6	14.6	
	250		3.5	0.8	8.6	2.0	68.0	77.7	32.0	66.6	112.4	100.2	175.3	121.6	0	13	100	99.0	99.3	98.8	11.0	10.3	12.3	
	150		3.5	0.8	8.4	2.0	63.3	72.3	27.7	61.8	82.2	76.6	114.2	88.2	0	11	85	98.6	98.9	98.3	9.8	9.3	10.8	log
	100		3.5	0.8	8.1	2.0	55.0	63.7	22.5	54.5	63.6	60.9	80.8	68.2	0	7	67	98.1	98.4	97.6	8.9	8.7	9.8	
	50		3.5	0.8	7.7	2.0	40.5	48.0	14.5	37.1	40.0	39.9	44.1	42.8	0	7	36	97.0	97.5	96.4	7.8	7.7	8.4	
19	450	061001_**SB1.7.1**_a0.02_**b0** -cmpgrid-65	3.9	0.5	12.0	2.0	0.0	0.0	0.0	0.0	434.3	447.9	399.1	441.1	0	0	0	99.3	99.4	99.0	13.3	12.3	15.4	
	250		3.7	0.5	11.7	2.0	0.0	0.0	0.0	0.0	242.0	248.9	223.0	245.2	0	0	0	98.9	99.1	98.6	11.7	11.0	13.2	
	150		3.5	0.5	11.6	2.0	0.0	0.0	0.0	0.0	145.6	149.3	134.6	147.2	0	0	0	98.5	98.8	98.2	10.4	10.0	11.5	log
	100		3.5	0.5	11.4	2.0	0.0	0.0	0.0	0.0	97.3	99.6	90.3	98.2	0	0	0	98.2	98.5	97.8	9.6	9.3	10.5	
	50		2.4	0.4	10.9	2.0	0.0	0.0	0.0	0.0	48.9	49.8	45.9	49.2	0	0	0	96.6	97.1	95.9	8.4	8.3	9.0	
18	450	060930_**OLD**_a0.02_b0_synthetic -cmpgrid-65	2.6	0.6	9.2	1.9	0.0	0.0	0.0	0.0	438.3	447.5	410.2	441.7	0	0	0	98.5	98.7	98.0	17.1	16.2	20.3	
	250		2.6	0.6	8.9	1.9	0.0	0.0	0.0	0.0	243.6	248.6	228.3	245.4	0	0	0	98.2	98.4	97.6	13.7	13.2	15.8	
	150		2.6	0.6	8.9	1.9	0.0	0.0	0.0	0.0	146.3	149.2	137.2	147.3	0	0	0	98.0	98.1	97.2	11.9	11.5	13.3	log
	100		2.6	0.6	8.9	1.9	0.0	0.0	0.0	0.0	97.6	99.5	91.7	98.3	0	0	0	96.7	97.1	95.9	10.6	10.4	11.7	
	50		2.6	0.6	8.9	1.8	0.0	0.0	0.0	0.0	48.9	49.8	46.2	49.2	0	0	0	94.8	95.3	93.8	8.7	8.6	9.3	
17	450	060930_**SB1.7.1**_a0.02_**b10-5** -cmpgrid-67	2.4	0.9	10.9	2.1	78.2	86.5	40.6	79.4	119.1	95.8	246.7	127.4	0	161	622	99.1	99.3	98.8	14.9	13.2	16.9	
	250		2.4	0.8	10.3	2.0	76.0	83.9	38.6	75.7	78.5	68.9	147.6	87.7	0	150	573	98.7	99.0	98.3	12.8	11.5	14.0	
	150		2.4	0.8	10.3	2.0	74.5	80.9	35.6	72.9	56.3	52.3	94.9	64.2	0	145	532	97.9	98.5	97.6	11.3	10.3	12.1	log
	100		2.4	0.8	10.3	2.0	72.1	77.9	33.6	68.8	44.3	42.5	67.5	50.8	0	133	481	97.4	98.2	97.2	10.2	9.5	10.9	
	50		2.2	0.8	10.0	2.0	62.7	66.6	28.0	56.5	29.6	29.6	38.5	33.7	0	105	321	95.7	96.4	95.0	8.4	8.2	9.0	
16	450	060929_**SB1.6.1**_a0.02_**b10-5_BU5** cmpgrid-66	2.2	0.8	9.2	1.9	74.3	85.0	37.8	77.4	141.7	105.3	269.5	135.8	0	148	492	98.8	99.0	98.4	19.0	16.8	21.5	
	250		2.2	0.7	8.1	1.9	71.7	82.3	36.2	74.5	91.8	74.4	161.1	91.9	0	141	447	98.1	98.6	97.8	16.0	14.6	17.9	
	150		1.6	0.6	7.7	1.8	70.1	80.0	35.2	71.9	64.6	56.3	104.5	67.3	0	136	417	96.7	97.5	96.2	13.6	12.8	15.1	log
	100		1.6	0.6	7.7	1.8	67.4	75.8	29.7	67.8	49.9	45.4	74.5	53.1	0	125	387	96.0	96.9	95.5	11.9	11.4	13.1	
	50		1.6	0.6	6.7	1.7	57.8	62.7	21.4	53.3	32.4	31.3	41.0	35.2	0	92	266	93.6	94.2	92.3	9.4	9.3	10.2	
15	50	060929_**SB1.7.1**_a0.02_**b10-5**	2.9	0.4	12.8	2.0	61.5	65.5	23.0	53.6	29.4	30.8	38.5	34.8	0	81	298	95.0	95.8	94.4	9.5	9.5	10.4	log
14	50	060928_**SB1.6.1**_a0.02_**b10-5_BU5**	2.0	0.8	10.2	1.9	59.1	62.5	20.0	53.0	31.0	30.3	40.3	35.4	0	100	248	93.0	94.4	92.5	9.5	9.3	10.3	log
13b	50	060929_**SB1.6.1**_a0.02_**b0_BU1**	2.2	0.6	8.1	1.9	0.0	0.0	0.0	0.0	49.0	49.8	46.4	49.2	0	0	0	94.5	95.4	93.7	8.8	8.6	9.4	log
13a	50	060928_**SB1.6.1**_a0.02_**b0_BU1**	2.0	0.6	11.1	2.1	0.0	0.0	0.0	0.0	49.1	49.8	45.2	49.1	0	0	0	94.8	95.6	94.1	8.7	8.6	9.5	log
12b	50	060929_**SB1.6.1**_a0.02_**b0_BU5**	1.8	0.5	7.3	1.7	0.0	0.0	0.0	0.0	49.2	49.8	46.7	49.3	0	0	0	93.6	94.4	92.6	9.4	9.3	10.2	log
12a	50	060928_**SB1.6.1**_a0.02_**b0_BU5**	1.8	0.7	8.6	1.9	0.0	0.0	0.0	0.0	49.2	49.7	46.3	49.2	0	0	0	93.2	94.2	92.3	9.4	9.3	10.3	log
11	50	060926_**SB1.6**_a0.02_**b10-5_BU5**	1.6	0.4	5.8	1.3	56.2	64.5	24.7	52.7	31.6	30.0	40.7	34.5	0	88	230	94.1	94.9	93.3	9.2	9.2	10.0	log
10	50	060927_**SB1.6**_a0.02_b0_**BU5** 5556	1.6	0.6	10.2	2.0	0.0	0.0	0.0	0.0	49.3	49.8	45.6	49.1	0	0	0	92.9	93.9	91.9	9.5	9.4	10.4	log

9	50	060926_**SB1.6**_a0.02_b0_**BU5** 5000	2.2	0.4	5.9	1.3	0.0	0.0	0.0	0.0	49.1	49.8	47.3	49.4	0	0	0	92.1	92.7	90.8	9.9	10.0	11.0	log
8	50	060926_**SB1.6**_a0.02_b0_**BU1** 5556	2.0	0.8	13.4	2.1	0.0	0.0	0.0	0.0	49.2	49.6	45.0	49.1	0	0	0	94.6	95.4	93.7	8.9	8.8	9.8	log
7	50	060926_**SB1.6**_a0.02_b0_**BU1** 5000	2.4	0.4	8.3	1.3	0.0	0.0	0.0	0.0	49.1	49.8	47.0	49.4	0	0	0	94.2	94.5	93.0	9.3	9.5	10.3	log
6	50	060920_**SB1.7**_a0.02_**b10-5**_synthetic_**bayes**	1.6	0.3	8.1	1.4	56.8	63.7	19.8	53.4	32.1	31.0	40.6	34.5	0	119	299	87.3	87.6	85.9	10.5	10.7	11.5	
	100		2.4	0.4	9.5	1.4	57.8	66.5	20.9	56.3	52.0	47.7	75.7	55.7	0	123	318	97.3	97.8	96.8	13.0	12.9	14.4	
5	50	060920_**SB1.7**_a0.02_b0_synthetic_**bayes**	2.2	0.2	8.4	1.5	0.0	0.0	0.0	0.0	49.1	49.9	46.7	49.3	0	0	0	94.2	94.7	93.1	9.3	9.5	10.3	log
4	50	060920_**SB1.7**_a0.02_b0_synthetic ***(not stable)***	2.0	0.3	8.9	1.4	0	0	0	0	49.2	49.9	46.5	49.4	0	0	0	95.2	95.6	94.4	9.0	9.2	10.0	log
3b	50	060928_**OLD**_a0.02_b0_synthetic 5556	2.2	0.5	6.2	1.7	0.0	0.0	0.0	0.0	49.0	49.8	47.4	49.2	0	0	0	93.6	94.6	93.1	9.2	8.9	9.7	log
3a	50	060927_**OLD**_a0.02_b0_synthetic 5556	1.6	0.7	10.8	2.1	0.0	0.0	0.0	0.0	49.3	49.7	45.6	49.1	0	0	0	94.9	95.4	93.7	9.2	9.2	10.3	log
2	50	060918_**OLD**_a0.02_b0_synthetic 5000	2.2	0.5	8.1	1.5	0	0	0	0	49.1	49.8	46.5	49.3	0	0	0	93.1	93.4	91.7	9.6	9.8	10.7	log
1	50	060919_**OLD**_a0.02_b0_oritinal	3.3	1.8	17.8	6.2	0	0	0	0	48.5	49.2	41.9	47.4	0	0	0	97.3	97.7	96.6	5.4	5.5	6.1	log

Conclusions

1. The **randomness of the experiments is high** (about 1% in TN in 50th iteration). This complicates the decisions significantly, because sometimes it is not possible to distinguish between randomness and effect of some change.
2. When beta > 0, new faces are added to the TS. **The distribution new examples doesn't follow the distribution of the initial TS**. The task becomes more difficult and the classifier gets worse. This is caused by the fact that TS and VS are very different.
3. The **ratio has to be estimated on all data available**, otherwise isn't the algorithms stable.

ID	Dataset	# faces	# non-faces	Comment
A	MIT+CMU	491	0	Originally 507 faces but some partially out of image
B	MatejSmid 1229	1129	0	Faces collected by Matej Smid
C	Hitachi 640	640	0	
D	Synthetic_A	5000	0	Synthetic faces with the same distribution as synthetic training/validation data sets
E	Non-skin 10,000	0	10000	
F	500 non face images	0	20,177,169	All windows in 500 non face images (i.e. window scanning with parameters: min_win_size: 24x24, row/col_shift_factor: 2/24, scale_factor: 1.25, no FineSeekOut, no NMS, no SVM)
G	130 degraded CMU images	0	21,263,707	All windows from modified dataset A (each face is upside down). Window scanning with parameters: min_win_size: 24x24, row/col_shift_factor: 2/24, scale_factor: 1.25, no FineSeekOut, no NMS, no SVM

Learning without VS, BlowUP effect

The main objective of the following experiments is to explore the possibilities of enlarging the training set by synthesis and design a algorithm which doesn't use the VS.

#	wc	parameters	FN (%)				TP (%)				Speed P E(#wc/win)				FP (-)			TN (%)			Speed N E(#wc/win)		
			A	B	C	D	A	B	C	D	A	B	C	D	E	F	G	E	F	G	E	F	G
23		061029_SB2.2_a0.02_**b0_D6k_BU1_BBnos** cmpgrid-66	HYP: ref.18, compare BBoci vs. BBnos																				
	500		4.3	2.4	22.3	1.8	0.0	0.0	0.0	0.0	480.6	489.4	396.0	491.3	0	0	0	99.86000	99.91292	99.82314	7.3	7.0	8.5
	50		3.3	2.0	19.7	1.7	0.0	0.0	0.0	0.0	48.7	49.1	41.3	49.2	0	0	0	98.37000	98.59097	97.88444	5.3	5.3	5.8
22		061029_SB2.2_a0.02_**b0_D6k_BU1_BBoci** cmpgrid-65	HYP: ref.19, compare BBoci vs. BBnos RES: BBoci and BBnos are very similar in TN point of view. BBoci allows to enlarge the DB easily from BetaFace.																				
	500		?	2.1	?	?	0.0	0.0	0.0	0.0	341.4	491.2	313.7	301.9	0	0	0	99.92000	99.91025	99.82607	6.5	6.5	7.6
	50		?	1.4	?	?	0.0	0.0	0.0	0.0	39.2	49.4	39.0	34.0	0	0	0	98.29000	98.30840	97.76505	5.2	5.3	5.6
21		061027_SB2.2_a0.02_**b0_D9332_betaFace** cmpgrid-64	HYP: Initial experiment to test the difficulty of TS and BB_eyes definition. RES: BetaFace has probably richer distribution than Schneiderman, seen from \TN EXP: BetaFace doesn't have ROP that's why /FN. **SYNTHETIZE ROP IN BETAFACE AND JOIN WITH SCHNEIDERMAN > 15k PTS SIZE*** BU																				
	500		?	4.9	?	?	0.0	0.0	0.0	0.0	244.9	476.3	252.6	178.9	0	0	0	99.83000	99.89510	99.79236	5.3	5.0	6.0
	50		?	4.2	?	?	0.0	0.0	0.0	0.0	27.5	48.1	30.0	20.4	0	0	0	98.20000	98.36814	97.88394	3.9	4.0	4.2
20		061024_SB2.2_a0.02_b10-6_BU10_D10k cmpgrid-67	HYP: ref.19																				
	500		3.7	2.7	5.9	2.0	75.2	87.1	47.3	87.6	147.5	105.0	279.3	101.8	0	25	96	99.65000	99.76865	99.57815	9.2	8.2	10.2
	50		3.3	2.6	5.0	1.9	56.0	59.0	23.8	61.9	31.9	31.5	41.8	31.1	0	15	41	97.3	97.8	96.9	6.1	6.0	6.6
19		061025_SB2.2_a0.02_b10-6_BU10_D10k_**resNTS** cmpgrid-66	HYP: ref.20 RES: resNTS makes the task more difficult, this can be seen from TN and SpeedN. **DO WE NEED TO EXPLORE THE ENTIRE N-DISTRIBUTION? WHY NOT SHRINK THE NTS > SAMPLE FROM PTS.**																				
	500		3.5	2.1	7.2	1.9	66.2	73.1	27.0	72.6	173.1	147.1	343.0	150.3	0	27	107	98.31000	98.69142	97.94674	14.2	12.4	16.5
	50		3.3	2.6	5.0	1.9	56.0	59.0	23.8	61.9	31.9	31.5	41.8	31.1	0	15	41	97.3	97.8	96.9	6.1	6.0	6.6
18		061024_SB2.2_a0.02_**b10-6_BU6x1.1_D10k** cmpgrid-65	HYP: \FN, \TN																				
	50		3.3	3.1	16.1	2.1	54.4	58.4	22.7	62.6	30.8	31.3	36.8	30.2	0	28	54	97.3	97.9	96.9	6.0	5.9	6.5
17		061021_SB2.2_a0.02_**b10-6_BU6_D10k** cmpgrid-64	HYP: stabilization of exp. 15 RES: EXP: around 300 it, only 50 P in ratio																				
	500		6.5	2.8	26.7	2.6	76.6	87.9	39.4	86.2	130.1	95.0	229.8	101.1	0	20	98	99.98000	99.98737	99.97077	7.1	6.6	7.7
	50		3.5	2.2	18.8	1.9	58.0	63.6	22.2	64.7	30.2	30.7	35.1	30.4	0	14	46	98.0	98.3	97.6	5.5	5.4	5.8

16		061021_SB2.2_a0.02_**b10-6_BU4_D10k** cmpgrid-65	HYP: (ref.14) RES: NOT STABLE																					
	100		3.5	1.8	18.0	2.0	64.2	75.3	28.1	73.9	49.5	44.9	64.7	45.4	0	18	75	98.9	99.1	98.6	6.6	6.4	7.1
	50		3.3	1.8	18.0	2.0	55.4	61.0	19.8	61.2	31.2	30.7	35.9	30.9	0	14	60	97.8	98.2	97.5	5.7	5.7	6.1
15		061021_SB2.2_a0.02_**b510-6_BU5_D9k** cmpgrid-64	HYP: RES: NOT STABLE iter. 110 EXP: not enough BU																					
	100		3.7	2.8	11.7	2.1	73.5	81.0	41.1	82.1	38.8	34.7	62.7	35.7	0	119	215	99.3	99.4	99.1	6.5	6.3	7.1
	50		3.3	2.7	11.7	2.0	65.4	70.6	29.1	70.7	25.9	24.7	36.9	25.8	0	110	185	97.7	98.2	97.3	5.8	5.8	6.3
13		061020_SB2.2_a0.02_**b510-6_BU3_D6k_R-3** cmpgrid-67, **91 removed**	HYP: /TN RES: NOT STABLE EXP: not enough P examples in RS around 90 iteration (~100)																					
	100		15.5	9.4	50.2	8.4	72.7	80.4	36.4	81.4	38.8	36.9	50.6	35.9	0	110	276	100.0	100.0	100.0	5.5	5.5	6.0
	50		4.3	2.6	22.7	2.7	64.4	67.1	29.1	69.5	26.6	26.2	30.0	25.9	0	93	215	98.6	98.7	98.2	5.2	5.2	5.5
12		061020_SB2.2_a0.02_**b510-6_BU3_D6k_R-1** cmpgrid-66, **190 removed**	HYP: /TN RES: NOT STABLE EXP: not enough P examples in RS around 120 iteration (~1)																					
	100		5.7	3.3	34.7	3.3	72.3	81.6	33.1	80.4	41.6	36.2	45.1	36.8	0	77	210	99.8	99.9	99.8	4.6	4.5	5.0
	50		4.5	3.1	33.3	3.0	62.9	71.4	26.6	70.8	28.6	26.6	27.3	26.9	0	65	177	98.8	99.0	98.5	4.3	4.3	4.6
11		061020_SB2.2_a0.02_**b510-6_BU3_D6k** cmpgrid-64	HYP: /TN RES: NOT STABLE EXP: not enough P examples around 110 iteration (~100)																					
	100		4.1	2.4	16.7	2.1	69.7	80.9	34.4	80.3	44.2	37.9	62.1	37.5	0	93	228	99.6	99.7	99.4	6.2	5.9	6.6
	50		3.7	2.4	15.6	2.0	59.7	68.3	25.5	70.3	29.4	27.2	35.8	26.8	0	73	195	98.0	98.5	97.9	5.7	5.5	6.0
10		061019_SB2.2_a0.02_**b0**_BU10_resNTS cmpgrid-65, **9000 TS**, 2x10x5556 RS	HYP: \TN (ref. 9) RES: \TN EXP: symmetric bootstrap boost the performance in TN 1% in 100 iteration																					
	100		3.5	2.4	18.1	2.1	0.0	0.0	0.0	0.0	96.9	97.7	82.9	98.0	0	0	0	97.9	98.1	97.2	7.1	7.0	7.9
	50		3.5	2.4	17.8	2.1	0.0	0.0	0.0	0.0	48.6	48.9	41.8	49.1	0	0	0	97.3	97.5	96.7	6.0	5.9	6.5
9		061019_SB2.2_a0.02_**b10-5**_BU10_resNTS cmpgrid-66, **9000 TS**, 2x10x5556 RS	HYP: /TN (ref. 8) RES: /TN, /FN EXP: better decision bigger over fitting on not seen data due to larger TS, FN still in limits																					
	100		3.1	2.6	19.4	2.2	72.9	80.9	33.6	80.4	39.0	32.9	57.9	34.4	0	243	421	98.8	98.9	98.4	6.5	6.4	7.1
	50		2.9	2.6	18.9	2.1	64.8	74.2	27.7	73.9	26.0	23.7	33.6	24.7	0	223	365	97.8	98.2	97.4	5.7	5.7	6.1
8		061019_**SB2.2**_a0.02_b10-5_BU10_resNTS cmpgrid-67, **7000 TS**, 2x10x5556 RS	HYP: /TN (ref. 7) RES: /TN, /FN EXP: better decision bigger over fitting on not seen data due to larger TS																					
	100		3.1	2.4	16.9	2.0	71.3	79.0	36.7	78.8	39.7	34.3	58.5	35.4	1	200	382	98.2	98.6	97.9	6.6	6.3	7.0
	50		3.1	2.4	16.9	2.0	65.8	72.2	30.2	72.4	25.7	23.7	33.8	24.6	1	182	341	97.3	97.8	97.0	5.5	5.4	5.8

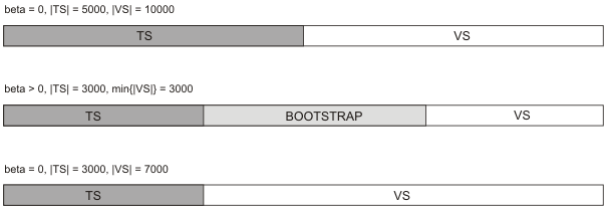
7		061019_**SB2.2**_a0.02_b10-5_BU10_**resNTS** cmpgrid-66, 5000 TS, 2x10x5556 RS	HYP: /TN RES: =TN EXP: no change at all																					
	50		3.5	2.1	11.3	2.1	65.4	69.7	30.3	69.3	25.7	25.4	35.7	25.9	0	166	329	96.7	97.2	96.1	5.8	5.8	6.4
6		061019_**SB2.2**_a0.02_b10-5_BU10_**resTS** cmpgrid-67, 5000 TS, 2x10x5556 RS	HYP: /TN, ?? (ref. 5) RES: \TN EXP: malo pozitivnich, potrebuju prozkoumat log co se tam stalo																					
	50		3.5	2.2	6.7		45.4	49.5	15.9	27.8	31.2	30.3	41.0	37.6	1	119	214	91.8	92.5	91.1	7.1	7.1	7.8
5		061017_**SB2.2**_a0.02_b10-5_BU10 cmpgrid-67, 5000 TS, 2x10x5556 RS	HYP: \FN (ref.3) RES: \FN EXP: TS and RS are more independent																					
	50		3.7	2.0	8.9		64.8	69.7	36.6	49.6	25.6	25.3	35.6	32.6	0	207	308	97.0	97.4	96.4	6.0	6.0	6.6
SB2.2 - **no skew** in synthetic images, TS sub randomly selected and **moved** from **x% RS**																								
4		061015_**SB2.1**_a0.02_b10-5_BU10 cmpgrid-66, 1000 TS, 2x10x5556 RS	HYP: \FN, \speedN, \TN (ref. 3) RES: \FN, \speedN, \TN, \TP EXP: only 1000 examples in TS used, that is not enough for good WC to be found																					
	500		3.5	1.7	14.7		64.8	75.3	23.0	53.9	192.6	159.9	340.5	249.6	0	121	338	97.5	98.1	97.3	23.2	20.0	25.0
	50		3.5	1.5	14.1		48.1	52.6	10.9	32.5	33.1	33.0	41.1	38.3	0	80	203	94.3	95.1	94.0	7.8	7.9	8.5
3		061014_**SB2.1**_a0.02_b10-5_BU10 cmpgrid-66, 5000 TS, 2x10x5556 RS	HYP: \ FN (ref. 1, 2) RES: \ speed, \ FN, \ TN EXP: Only 1/2 TS used compared to exp. 32 and hence not that good WC found -> \TN, \speed																					
	500		3.5	2.6	18.9		80.7	89.1	42.7	75.8	114.2	74.0	229.2	136.3	0	164	471	99.5	99.7	99.5	9.9	8.6	10.7
	50		2.9	2.6	18.0		66.4	73.9	23.3	52.1	26.4	25.3	34.7	33.0	0	125	298	97.3	97.7	96.9	5.8	5.7	6.3
SB2.1 - **no skew** in synthetic images, TS sub randomly selected and **copied** from **x% RS**																								
2		061014_**SB2.0**_a0.02_b10-5_BU10_**all** cmpgrid-65, **5556x2** TS, 2x10x5556 RS	HYP: \FN (ref. 31) RES: =FN, /TP EXP: over fitting almost the same as 1, increasing the number of TS doesn't have sense																					
	100		4.1	2.8	20.9		75.4	85.3	36.6	66.7	37.6	33.3	56.3	48.6	1	144	358	99.2	99.3	98.9	5.9	5.8	6.3
	50		3.9	2.8	20.6		66.8	74.2	28.6	53.6	25.4	24.6	33.1	32.1	1	133	293	98.3	98.5	97.9	5.3	5.3	5.6
1		061014_**SB2.0**_a0.02_b10-5_BU10 cmpgrid-66, **2500x2** TS, 10x2500 ratio	HYP: /FN, \TN, /speed (ref. 29) RES: /FN, \TN, /speed, over fitting EXP: no skew in TS, TS nor rich enough to cover entire CMU testing set																					
	100		4.1	3.2	12.5		72.1	81.1	34.1	63.3	40.7	34.3	65.9	49.6	0	132	292	99.0	99.2	98.7	6.0	5.9	6.7
	50		3.9	3.0	11.9		62.5	71.8	26.6	52.5	27.2	25.0	37.4	32.2	0	120	232	97.6	98.0	97.3	5.3	5.3	5.8
SB2.0 - **no skew** in synthetic images, TS sub randomly selected and **copied** from **100% RS**																								
0	50	OLD	3.3	1.8	17.8	2.5	0.0	0.0	0.0	0.0	48.5	49.2	41.9	48.9	0	0	0	97.3	97.7	96.6	5.4	5.5	6.1

Conclusions

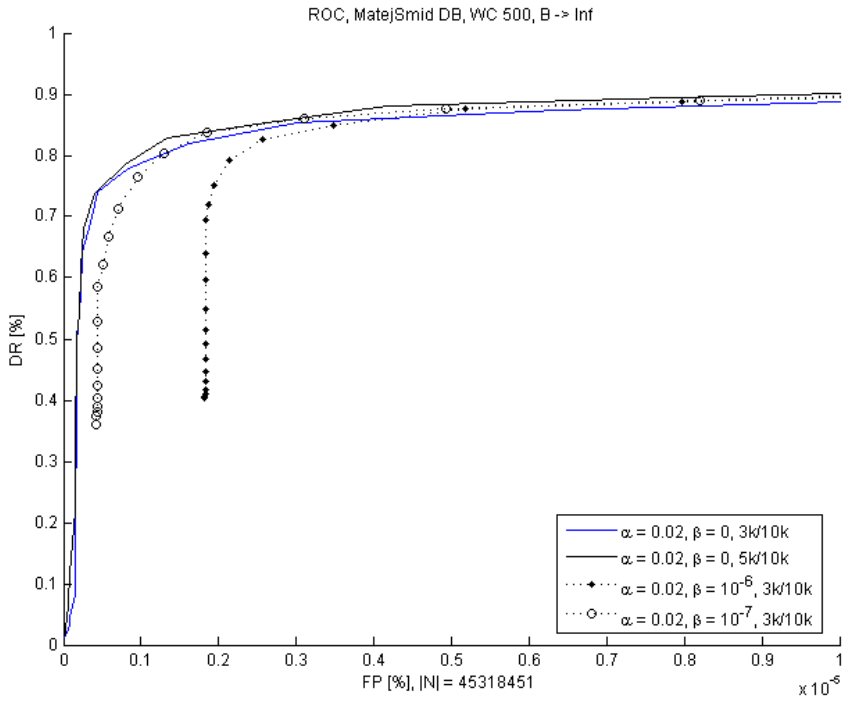
1. **beta = 10-6,** because of a huge disproportion between a priory probabilities, the allowed FP error (controlled by beta parameter) should be very small
2. **Resampling of negative training set doesn't have influence on the classifier performance.** In every step, negative examples from TS were replaces from VS.

Symmetric vs. nonsymmetrical WaldBoost on Schneiderman (Tot 10k)

The scheme of the experiment is illustrated in the following figure:



#	wc	params	FN (%)		TP (%)		Speed P E(#wc/win)		FP (-)		TN (%)		Speed N E(#wc/win)	
			CMU	MS	CMU	MS	CMU	MS	F	G	F	G	F	G
1	500	061108_SB2.3_a0.02_b0_TS3k-Tot10k cmpgrid-64, 43h	4.7	1.9	0.0	0.0	478.7	491.2	0	0	99.55145	99.27186	9.4	11.6
2	500	061108_SB2.3_a0.02_b0_TS5k-Tot10k cmpgrid-66, 43h	5.4	2.3	0.0	0.0	475.3	488.9	0	0	99.79792	99.65366	6.9	8.1
3	500	061110_SB2.3_a0.02_b10-6_3k10k cmpgrid-67, 42h	5.4	1.9	21.8	36.9	371.2	312.4	15	36	99.79411	99.63834	7.0	8.3
4	500	061110_SB2.3_a0.02_b10-7_3k10k cmpgrid-65, 35h	6.7	2.3	18.3	34.5	383.7	325.5	2	15	99.72603	99.52870	7.7	9.3

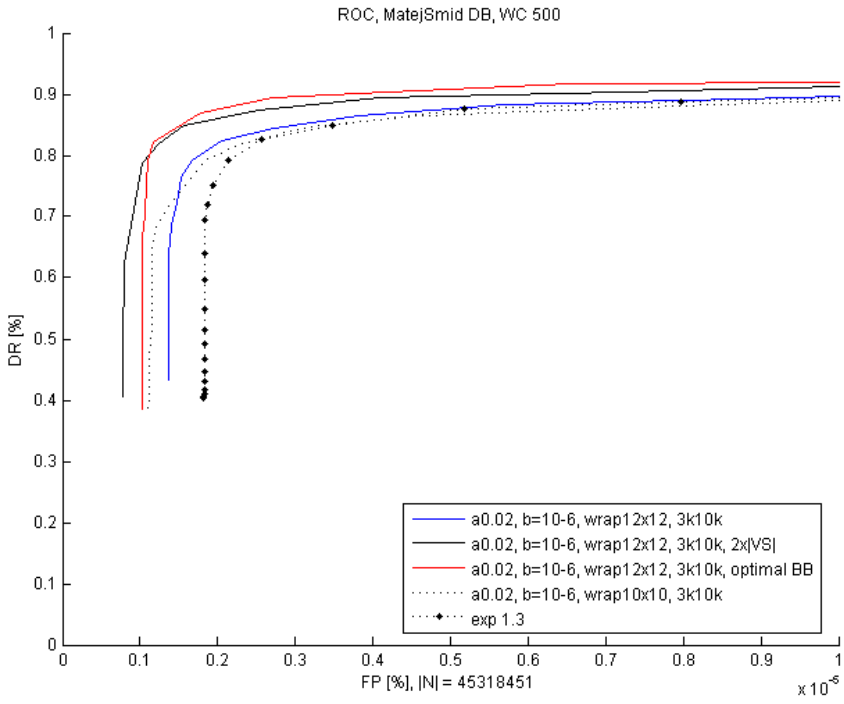


Conclusions

1. According to the ROC curves, there is much bigger FP error then allowed for symmetric bootstrap (beta>0). There are two possible improvements that are considered. **Contextual modeling of p(x|-1)**
   Increase the size of RS. Currently there are 10M negative examples which probably doesn't suffice for the threshold estimation.
2. ROC curves of the symmetric bootstrap are worse than for the non-symmetric. It seems like the slope is higher for smaller FP rates. Solving the problem with correct estimation of the threshold is promising.

Contextual background, maximal confidence bbox, better estimation of Theta_B

#	wc	parameters	FN (%)		TP (%)		Speed P E(#wc/win)		FP (-)		TN (%)	
			CMU	MS	CMU	MS	CMU	MS	F	G	F	G
1		061119_SB3.0_a0.02_b10-6_wrap12x12_3k10k	What is the influence of contextual background compared to exp. 1.3									
	500	cmpgrid-64, 29h	8.9	2.1	25.2	39.5	340.4	300.1	12	38	99.78279	99.67163
2		061119_SB3.0_a0.02_b10-6_wrap12x12_3k10k_ratioon210+7	How much influence the size of RS the errors FP in SB									
	500	cmpgrid-65, 53h	6.9	1.8	26.0	44.3	343.3	278.5	5	20	99.58486	99.43614
3		061119_SB3.0_a0.02_b10-6_wrap12x12_3k10k_optBB1px	Maximal confidence bounding box, it's influence to the dificulty of the ta									
	500	cmpgrid-66	6.9	1.2	27.0	38.1	336.9	310.1	20	42	99.59847	99.46088
4		061119_SB3.0_a0.02_b10-6_wrap10x10_3k10k	Parts of faces in background									
	500	cmpgrid-67	7.7	1.4	22.6	38.1	354.7	309.3	16	70	99.65794	99.53747
5		061120_SB3.0_a0.02_b10-6_wrap12x12_3k10k_160M	Decrease of the size of the NTS to 160M as before for comparison									
	500	cmpgrid-64	6.5	1.8	27.2	40.0	338.0	298.1	11	28	99.47790	99.28049



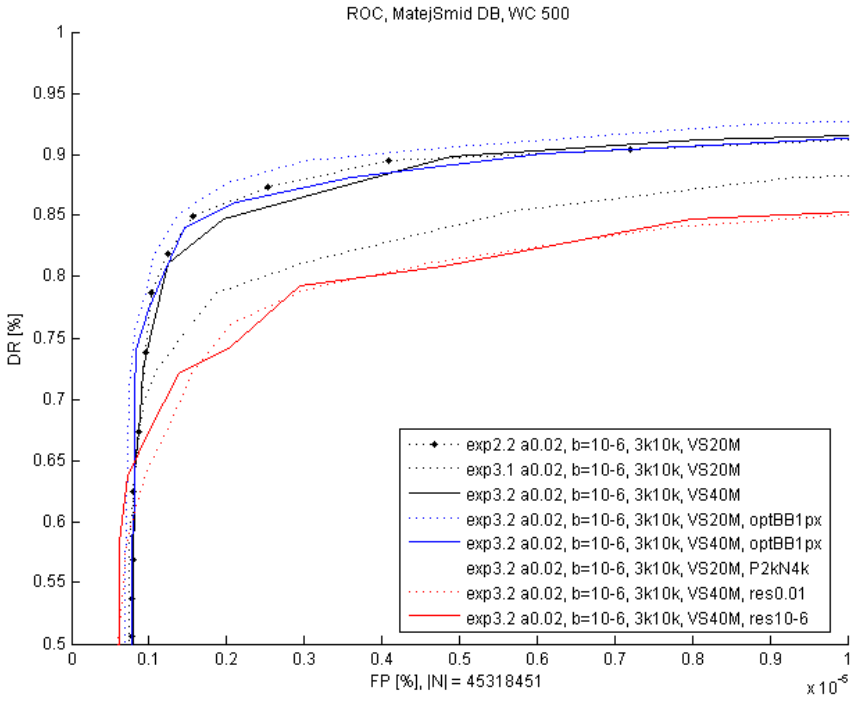
Conclusions

1. Bigger VS improves the performance and eliminates the FP obtained by wrong estimation of the Theta_B threshold.
2. Optimal bbox simplifies the task and doesn't worsen the FN ratio!
3. Contextual background eliminates some portion of FP even with small VS size.
4. All above mentioned improvements should be considered in the future training.

Size of VS, resampling of TS

How does the size of VS affect the quality of classifier? Does it make sense to resample TS?

#	wc	parameters	FN (%)		TP (%)		Speed P E(#wc/win)		FP (-)		TN (%)		Speed N E(#wc/win)	
			CMU	MS	CMU	MS	CMU	MS	F	G	F	G	F	G
1		061122_SB3.1_a0.02_b10-6 3k10k_VS20M_RTS0, cmpgrid-64	Randomnes of the experiment, compared with exp 2.2											
	500		6.2	2.3	27.0	37.7	342.3	308.3	14	38	99.60670	99.45460	9.3	10.6
2		061122_SB3.1_a0.02_b10-6 3k10k_VS40M_RTS0, cmpgrid-65	Bigger RS											
	500		6.3	1.8	22.4	40.4	361.5	297.0	12	47	99.53890	99.34807	8.6	9.9
3		061123_SB3.1_a0.02_b10-6 3k10k_VS20M_RTS0_optBB1px, cmpgrid-66	Maximal confidence BB											
	500		8.3	2.0	21.6	40.8	357.1	296.6	10	40	99.59367	99.46345	7.6	8.4
4	500	061123_SB3.1_a0.02_b10-6 3k10k_VS40M_RTS0_optBB1px, cmpgrid-71	6.0	2.0	23.0	40.6	363.5	297.2	14	21	99.66359	99.55673	8.5	9.3
5		061123_SB3.2_a0.02_b10-6 3k10k_VS20M_RTS0_P2kN4k, cmpgrid-68	Different ratio between p(1)/p(-1)											
	500		6.3	2.0	41.5	59.2	283.4	221.5	20	42	99.50120	99.31203	10.0	11.4
6		061124_SB3.3_a0.02_b10-6 3k10k_VS40M_res0.01, cmpgrid-67	Resampling											
	500		12.5	3.3	22.8	43.0	342.9	281.6	14	26	99.89093	99.84400	7.4	8.3
7	500	061124_SB3.3_a0.02_b10-6 3k10k_VS40M_res10-6, cmpgrid-70	10.5	2.0	22.8	43.0	314.2	257.1	14	26	99.82003	99.75382	7.7	8.6



Conclusions

1. **Resampling of TS doesn't make sense**
2. The randomness of the results is big (3.1 vs. 3.2)
3. Further enlargement of VS doesn't have sense.

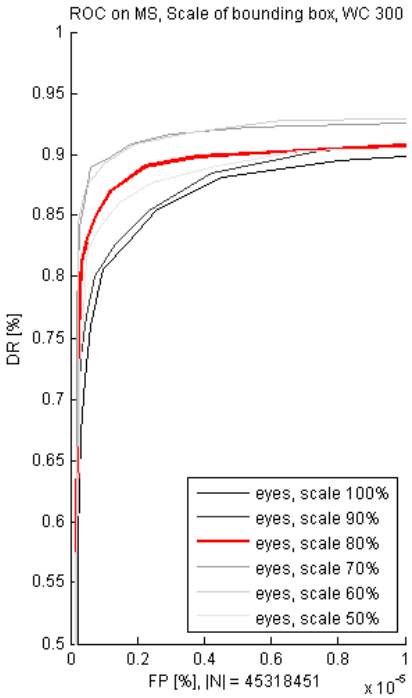
Bounding box alignment

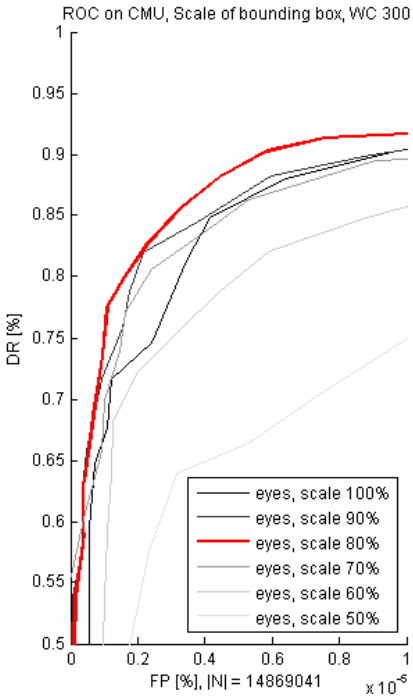
The bbox alignment significantly influences the face detector's performance. Several possible definitions differing in the bbox size and alignment are tested in the following experiment:

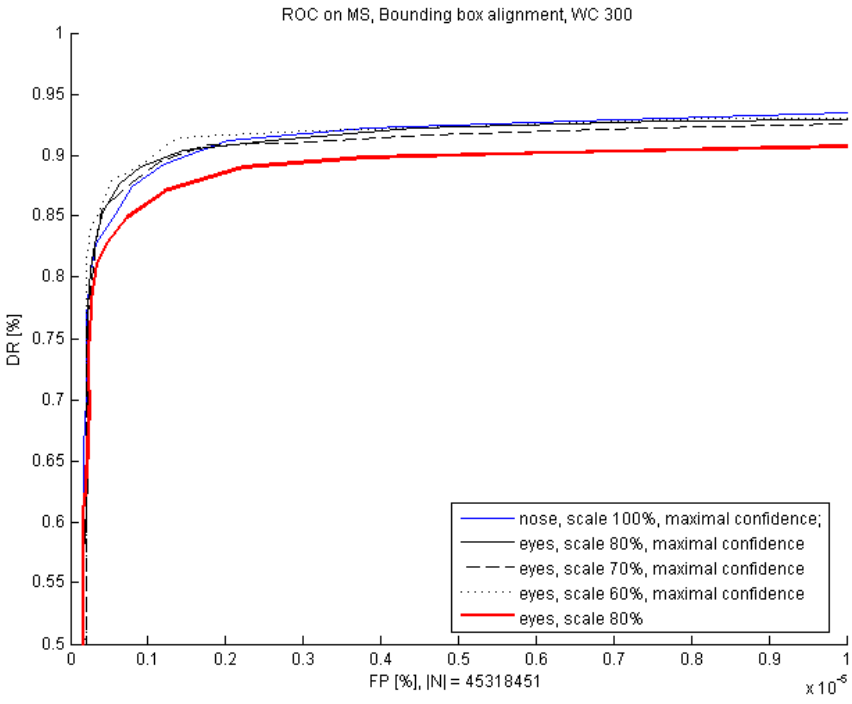
- **nose_2.7**, original bbox proposed by JS, alignment to nose, 2.7 is internal constant determining the size of the bbox
- **eyes_x.x**, new definition originated from BetaFace detections, alignment to eyes, 1.0 denotes the original size
- **maxConf**, new concept of the bbox definition where the size and shift of bbox is chosen according to maximal confidence

#	wc	parameters	FN (%)		TP (%)		Speed P E(#wc/win)		FP (-)		TN (%)		Speed N E(#wc/win)	
			CMU	MS	CMU	MS	CMU	MS	F	G	F	G	F	G
1	300	061129_SB3.4_a0.02_b0_5k10k_nose_2.7	4.8	4.1	0.0	0.0	286.8	287.9	0	0	99.55529	99.20154	7.7	9.5
2	300	061129_SB3.4_a0.02_b0_5k10k_eyes_1	2.7	1.6	0.0	0.0	293.1	295.4	0	0	99.19443	98.71031	9.0	11.1
3	300	061129_SB3.4_a0.02_b0_5k10k_eyes_0.9	4.2	2.3	0.0	0.0	288.7	293.8	0	0	99.74371	99.54833	7.3	8.6
4	300	061129_SB3.4_a0.02_b0_5k10k_eyes_0.8	4.0	4.3	0.0	0.0	288.9	288.3	0	0	99.81806	99.66027	6.2	7.2
5	300	061214_SB4.0_a0.02_b0_5k10k_eyes_0.7	4.4	4.1	0.0	0.0	287.7	288.3	0	0	99.61222	99.38571	6.4	7.3
6	300	061214_SB4.0_a0.02_b0_5k10k_eyes_0.6	5.9	2.0	0.0	0.0	284.3	294.2	0	0	99.56738	99.32527	7.3	8.3
7	300	061214_SB4.0_a0.02_b0_5k10k_eyes_0.5	8.0	1.6	0.0	0.0	278.4	295.5	0	0	99.29518	99.10632	7.5	8.2
8	300	061203_SB3.4_a0.02_b0_5k10k_eyes_0.8 maxConf	5.0	3.1	0.0	0.0	287.7	291.0	0	0	99.85646	99.79236	4.1	4.3
9	300	061217_SB4.0_a0.02_b0_5k10k_eyes_0.7 maxConf	6.9	4.9	0.0	0.0	280.1	285.8	0	0	99.88134	99.80176	4.1	4.5
10	300	061217_SB4.0_a0.02_b0_5k10k_eyes_0.6 maxConf	15.5	3.3	0.0	0.0	258.2	290.7	0	0	99.90989	99.85345	3.9	4.1
11	300	061129_SB3.4_a0.02_b0_5k10k_nose_2.7 maxConf	6.7	2.3	0.0	0.0	282.9	293.2	0	0	99.86213	99.80082	5.5	5.8

Experiment details:
alpha = 0.02, **beta** = 0, **Faces**: Schneiderman DB, |TS| = 5500, |VS| = 5500, **Background**: non-contextual, 160M examples
SB3.4 - possible to run experiments with alignment to eyes/nose, contextual/non-contextual background
SB4.0 - stage WaldBoost







Conclusions

- **maxConf** boosts the performance of both alignments, increases FN on CMU slightly, increases the speed
- the smaller bbox aligned to eyes the better performance on MS, the worse performance on CMU
- **bbox aligned to eyes 80% performs the best all together (very similar to 70%)**

Symmetric|non-symmetric bootstrap

Compare the symmetric and non-symmetric bootstrap. Four classifiers are trained on identical positive training set with more than 67.000 examples, 10.000 faces can be used in training, the rest of faces is used for ratio estimation.
[1] non-symmetric, 10.000 positive training examples.
[2] symmetric, 10.000 + bootstrap, beta = 10-7
[3] symmetric, 10.000 + bootstrap, beta = 10-6
[4] non-symmetric with removal, 10.000 - bootstrap, beta = 10-7

Experiment setting:
version: SB3.6.2, alpha: 0.0200;
training set: num_data_P: 10000; num_data_N: 10000;
ratio estimation: neg_ratio_num: 20000000
positive: pos_blow_up: 2; bb_shift: 0.7500; bb_RIP_laplace: 4.8000; bb_scale: 0.0625; bb_eyes: 1; Total = 67192
negative: context_dsc: '../context_dsc/'; bb_wrap_out: 0.6250; bb_wrap_in: 0.5000; Total = 260.000.000

#	wc	parameters	FN (%)		TP (%)		Speed P E(#wc/win)		FP (-)		TN (%)		Speed N E(#wc/win)	
			CMU	MS	CMU	MS	CMU	MS	F	G	F	G	F	G
1	300	070104_SB3.6.2_b0 **cmpgrid-65**, non-symmetric	4.2	1.8	0.0	0.0	288.8	294.9	0	0	98.98037	98.69044	9.7	10.6
2	300	070104_SB3.6.2_b10-7 cmpgrid-66, symmetric	4.6	1.8	39.8	60.2	193.2	154.3	0	7	99.11527	98.85964	8.9	9.7
3	300	070104_SB3.6.2_b10-6 cmpgrid-68, symmetric	4.8	2.0	45.3	61.9	169.4	136.8	16	58	99.27760	99.07241	9.1	9.9
4	300	070104_SB3.6.2_b10-7_NOPBS cmpgrid-70, symmetric, no positive bootstrap	4.4	1.8	41.6	62.7	193.7	154.8	3	10	98.89977	98.59309	10.1	11.0

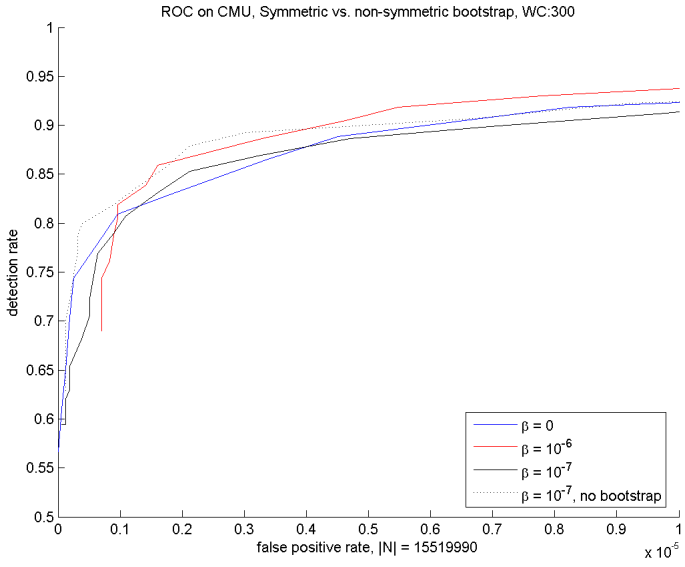


Figure 1: ROC curve tested on MIT-CMU dataset. The ROC curve of the non-symmetric Waldboost is marked by blue color (exp. [1]). Removing the easy faces from the training (dotted black line, exp. [4]) the ROC curve was shifted upward. By adding more difficult faces to training, i.e. symmetric bootstrap (black line, exp. [2]) the ROC curve drops !!!. The best ROC curve corresponds to the symmetric bootstrap with higher allowed TP rate (beta = 10-6, red line, exp. [3])

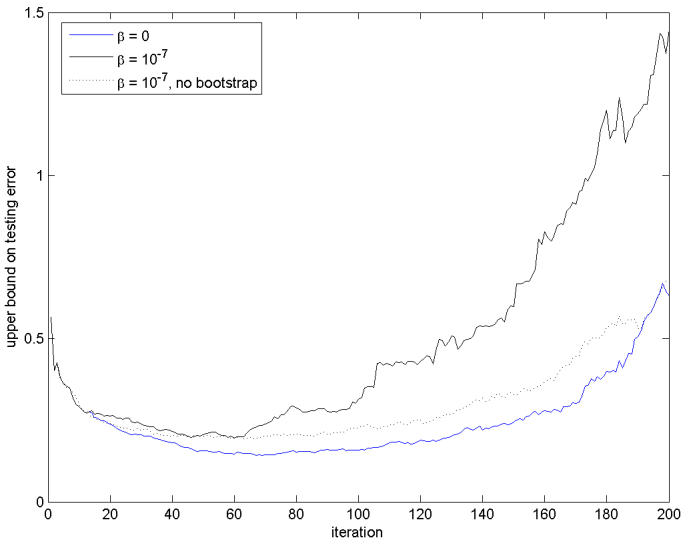


Figure 2: The upper bound on testing error during training. The upper bound seems to grow the fastest for symmetric bootstrap (exp. [2], black line). In the case, when the easy examples are removed from training but no new are bootstrapped (dotted black line, exp. [4]), the upper bound is similar to the non-symmetric version (blue line, exp. [1]).

Conclusions

According to the figure 2, it seems that the symmetric bootstrap reaches faster the possibilities of the weak learner and hence the upper bound grows earlier compared to the non-symmetric version. This suggest that the weak classifiers are not generalizing after some time of learning and hence concentrating on difficult examples doesn't bring much benefit for this particular case. Better results should be expected when stronger, more generalizing, features are used.

ID	Dataset	# faces	# non-faces	Comment
CMU	MIT+CMU	503	0	Originally 507 faces but some partially out of image or not labeled
MS	MatejSmid 1229	1129	0	Faces collected by Matej Smid
F	500 non face images	0	20,177,169	All windows in 500 non face images (i.e. window scanning with parameters: min_win_size: 24x24, row/col_shift_factor: 2/24, scale_factor: 1.25, no FineSeekOut, no NMS, no SVM)
G	130 degraded CMU images	0	21,263,707	All windows from modified dataset CMU (each face is upside down). Window scanning with parameters: min_win_size: 24x24, row/col_shift_factor: 2/24, scale_factor: 1.25, no FineSeekOut, no NMS, no SVM