AdaBoost for Fast Face Detection
PhD Thesis Proposal

Jan Šochman
sochmj1@cmp.felk.cvut.cz

CTU–CMP–2004–04
March 1, 2004

Available at

Supervisor: Dr. Jiří Matas

The author was supported by The Grant Agency of the Czech Republic under project GACR 102/02/1539 and by The Grant Agency of the Czech Technical University under project CTU 0307313.

Research Reports of CMP, Czech Technical University in Prague, No. 4, 2004
Published by
Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
Contents

1 Introduction 1

2 AdaBoost 5
   2.1 The algorithm 6

3 Face detection overview 7

4 Adopted face detection approach 10

5 Data sets 12
   5.1 Training data set 12
   5.2 Test data set 13

6 AdaBoost with totally corrective updates 13
   6.1 Totally corrective step 14
   6.2 Experiments 17
   6.3 Conclusions 20

7 Using previous stage knowledge 20
   7.1 Proposed method 20
   7.2 Experiments 23
   7.3 Conclusions 25

8 Summary and thesis proposal 26
Abstract

The main objective of our research is to study the two class example-based learning problems with different measurement costs for the classes, where time of the classifier evaluation is limited or where the evaluation time is the optimisation parameter. The face detection problem is used as an example of such problem, nevertheless the results should be applicable to the other similar problems. For the classifier training the AdaBoost algorithm is used.

Two improvements of the state-of-the-art algorithms are proposed in this thesis proposal. The totally corrective algorithm with coefficient updates improves the evaluation time of a single classifier trained by the AdaBoost algorithm by minimising the training error upper bound more aggressively and producing shorter classifiers. The second proposed enhancement, using the previous-stage knowledge during the training of a cascaded classifier shorten the classifier average evaluation time by a partial sequentialising of the cascade building training and speeds up the training itself.

Since many problems in computer vision are of the studied form a proper formulation of these problems and development of the corresponding theory will lead to better understanding of these problems and to more efficient solutions.

1 Introduction

In the standard statistical formulation of the example-based learning a finite training set $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is given, where $x_i \in X$ is a training sample from the feature space $X$ and $y_i \in Y$ is a respective class label. The goal of the learning is to find a classification function $d^*(x) : X \rightarrow Y$ (a classifier) with minimal classification error on the whole feature space $X$

$$d^*(x) = \arg\min_d \int_X L(d(x), c(x)) \, dx,$$  

(1)

where $c(x)$ represents the correct label for $x$ and $L(\cdot)$ is a class label distance (e.g. 0/1 inequality indicator). However, having the finite training set $T$ only the empirical estimate can be made

$$d^n(x) = \arg\min_d \sum_{i=1}^n L(d(x_i), y_i).$$

(2)

A typical problem in this formulation is that the size of the training set $T$ is insufficient to fully represent the classified data. Hence, the found classification function $d(x)$ should generalise well to the unobserved data given only limited data sample.
To cover the variability of the data it is desirable to work with the training set as large as possible. However, the training set size is limited by the real-word limitations. Therefore, the training set in the example-based training is usually given ahead of time, limited and it is assumed that sampling of the feature space $X$ is sufficient.

In computer vision many example-based learning problems does not fit into this statistical model. A classical example of such problems is the object detection in an image, e.g. face detection, licence plate detection or text detection. In the following the face detection problem is used as a representative of this type of problems and all devised algorithms are directly applied to this problem. In the face detection the example-based learning is used very often. Nevertheless, several important differences from the above model can be found.

Face detection is a two class problem. Let us denote the face class $\mathcal{F}$, the non-face class $\mathcal{F}$ and $Y = \{c_{\mathcal{F}}, c_{\mathcal{F}}\}$ where $c_{\mathcal{F}}$ is a label for the class $\mathcal{F}$ and $c_{\mathcal{F}}$ for the class $\mathcal{F}$. In the ideal case, the classifier is able to assign a correct label to every sample $x \in X$. In practice, two different types of error can occur. The first type of error occurs when $x \in \mathcal{F}$ is classified as a non-face. This error is called missed detection (or false negative) and its probability

$$P_{MD}(d) = \int_{\mathcal{F}} I[d(x) = c_{\mathcal{F}}] p(x|\mathcal{F}) \, dx$$  \hspace{1cm} (3)$$

where $I[p]$ returns 1 when the predicate $p$ is true and 0 otherwise, should be kept under some very low threshold since such classification excludes a real face from any further processing. Second type of error occurs when $x \in \mathcal{F}$ is classified as a face. This error is called false alarm (or false positive) and its probability

$$P_{FA}(d) = \int_{\mathcal{F}} I[d(x) = c_{\mathcal{F}}] p(x|\mathcal{F}) \, dx$$ \hspace{1cm} (4)$$

we want to minimise. Having in mind this asymmetry of the classification task the example-based learning can be reformulated according to Neyman and Pearson (see for example [23]) to

$$d^* = \arg \min_{d \in D_\alpha} P_{FA}(d)$$ \hspace{1cm} (5)$$

where $D_\alpha = \{d : P_{MD}(d) < \alpha\}$ and $\alpha$ is some low threshold determining the missed detection rate.

Another asymmetry can be seen in the measurement cost of the samples from the classes $\mathcal{F}$ and $\mathcal{F}$. In the above example-based learning formulations large enough training set was given and the measurement cost was neglected. However, it is more difficult to collect the face images than the non-face ones since any
sub-window of an image containing no faces can be used as a non-face sample. Almost arbitrary large training set can be easily constructed using these non-face samples. The problem here is not a small number of training samples but too have to be selected to keep the learning tractable. Two techniques selecting the relevant samples have been proposed.

In the bootstrapping [26] a classifier is trained in rounds. Let $T_1 = T_1^r \cup T_1^f$ be a training set containing all face samples $T_1^r$ and moderate amount of randomly selected non-face samples $T_1^f$. In the round $t = 1, 2, ...$ a classifier $d_t : X \rightarrow Y$ is trained using the training set $T_t$. The trained classifier divides $F$ into two distinct parts $X_t = \{x : x \in F \land d_t(x) = c_F\}$ and $\bar{X}_t = \{x : x \in F \land d_t(x) = c_{\bar{F}}\}$. The training set for the round $t + 1$ is constructed as $T_{t+1}^r = T_t^r$ and $T_{t+1}^f = T_t^f \cup \mathcal{U}_t$ where $\mathcal{U}_t$ is a set uniformly sampled from $\bar{X}_t$. The loop is terminated at a round $T$ when sufficiently accurate classifier is found or when adding new samples to the training set does not improve the classifier performance significantly. The output of the learning is then the classifier $D = d_T$. In this approach the training set grows slowly and hence the learning time increases in each round as well as the complexity of the classifier when not restricted.

Using the notation from the previous paragraph the second technique, the cascade building [27] can be described as follows. Again, the classifier is trained in rounds. The training set $T_1$ is the same as in the bootstrapping. In the round $t$ a classifier is trained according to (5) with an additional stopping criterion $P_{FA} < \beta$. Hence, $d_t$ is not an optimal classifier with minimal $P_{FA}$. However, it has the missed detection rate lower than $\alpha$ and the false alarm rate lower than $\beta$. The non-face class $\bar{F}$ is divided into two distinct parts $X_t = \{x : x \in \bar{F} \land d_t(x) = c_{\bar{F}}\}$ and $\bar{X}_t = \{x : x \in \bar{F} \land (\exists q \leq t : d_q(x) = c_F\}$. The training set for the round $t + 1$ is constructed as $T_{t+1} = (T_t \setminus \mathcal{V}_t) \cup \mathcal{U}_t$ where $\mathcal{V}_t = \{x \in T_t : d(x) = c_F\}$ and $\mathcal{U}_t$ is the same as in the bootstrapping. Note that the non-face class division depends on all already found classifiers and that only few faces are wrongly classified and the rest remains in the training set. Moreover, the size of the training set can be kept constant by letting $|\mathcal{U}_t| = |\mathcal{V}_t|$. The loop termination criterion is again the same as in the bootstrapping. The output of the learning is the sequence of classifiers $D = (d_1, ..., d_T)$. During the classification of an unobserved data sample $x$ the classifiers are evaluated sequentially. The sample $x$ is classified as a non-face at time $t$ if $d_t(x) = c_{\bar{F}}$ otherwise the decision is deferred and and the sample is passed to the classifier $d_{t+1}$. If the sample has passed to the classifier $d_T$ the output $D(x) = d_T(x)$.

Both the bootstrapping and the cascade building construct a training set by adding relevant samples only. Two main differences between the approaches are (1) the size of the training set used for training in each round (is growing in the bootstrapping and stays constant in the cascade building) and (2) the form of the
Viola and Jones showed [27] that one monolithic classifier trained on the same data as the cascaded classifier performs better. However, its training is more computationally expensive using only the bootstrapping and due to the sequential evaluation the cascaded classifier is much faster in average.

Time of the classifier evaluation is another important criterion in designing a classifier for the face detection task which is not covered by the standard statistical formulation. A classifier $D(x)$ used for the face detection is ordinarily evaluated many times a second. Typically an image is scanned and the classifier is evaluated at every position in the image and at several scales. For the face detection in video sequences or when searching huge data sets this process is repeated many times a second. The formulation of the example-based learning have to be modified to work with the classifier evaluation time.

Denoting $T(D)$ the average time of evaluation of the classifier $D$, the two different formulations of the example-based learning with the time parameter are of particular interest. First one is reformulation of (5) for training a classifier with limited time of evaluation $T(D) < T_{max}$ and can be applied for example to the training of the final classifier:

$$D^* = \arg \min_{D \in D_{a,T_{max}}} P_{FA}$$

where $D_{a,T_{max}}$ is a set of classifiers faster than $T_{max}$ in average and with $P_{MD} < \alpha$. The second formulation is related for instance to the problem of training a single classifier in one round of the cascade building:

$$D^* = \arg \min_{D \in D_{\alpha,\beta}} T(D)$$

where $D_{\alpha,\beta}$ is a set of classifiers with $P_{MD} < \alpha$ and $P_{FA} < \beta$.

The main objective of our research is to study the two class example-based learning problems of the form (6) and (7) with different measurement costs for the classes. The face detection problem is used as an example of such problem, nevertheless the results should be applicable to the other similar problems.

The training set size reduction due to the cascade building is not negligible but the size needed for training a reasonable classifier is still too big. Not all learning algorithms can cope with such large training sets. For instance the space complexity of SVM is $O(n^2)$ where $n$ is the training set size which becomes prohibitive when $n > 10,000$ [8]. One of the algorithms suited for training from large training set is the AdaBoost algorithm used by Viola and Jones [27]. The AdaBoost is linear in $n$ both in time and space. Moreover, the classifier complexity can be controlled easily during the training. In the following we therefore restrict ourself to use of the AdaBoost algorithm for the classifier training.

In the following section the AdaBoost algorithm is explained in necessary detail. In Section 3 an overview of the face detection state-of-the-art is given. The
adopted face detection approach is described more closely in the Section 4 and
the data sets used for training and testing discussed in Section 5. Two proposed
methods, the totally corrective algorithm with coefficients updates and using the
previous-stage knowledge are described in subsequent Section 6 and 7. The thesis
proposal is summarised and the future directions outlined in Section 8.

2 AdaBoost

The AdaBoost algorithm is relatively new algorithm proposed by Freund and
Schapire [3]. It is a descendant of the weighted majority algorithm by Little-
stone and Warmuth [14] and the boost-by-majority algorithm by Freund [4]. All
three algorithms construct an ensemble of classifiers (predictors) and use a voting
mechanism for the classification. However, in the boost-by-majority algorithm
a parameter determining the upper bound on the used classifiers error has to be
known ahead of time. Moreover, the accuracy of the final classifier depends on
the error of the worst combined classifier and the algorithm cannot take advantage
of the classifiers with error smaller than the worst one. In AdaBoost, the accuracy
depends on the error of all classifiers in the ensemble which is often lower than the
worst case. The AdaBoost algorithm differs from both its precursors by the way
it combines the classifiers into an ensemble. It uses the weighted voting mecha-
nism and a weight assigned to a classifier depends on its error on the training set
– lower the error, higher the weight. This adaptive approach gave the name to the
algorithm. The algorithm itself has many interesting properties discussed below.

Although the AdaBoost is a quite new algorithm it has become very popular
and many variants of the original algorithm has been proposed. The original al-
gorithm [3] works with classifiers predicting labels of the classes only. Extension
to the real confidence values of the class belonging was proposed by Schapire and
Singer [22]. Schapire and Singer extended the original algorithm to the multi-
class and multi-label cases as well. Connections to many other theoretical results
have been made, like to the linear regression by Friedman et al. [5], to the margin
theory by Schapire et al. [21], to the minimisation of the Bregman divergence by
Collins et al. [1] and Kivinen and Warmuth [9]. A lot of work has been done on
the theoretical bounds of the AdaBoost error [21, 10]. The following explanation
focus mainly on the discrete version of the AdaBoost using a newer and simpler
notation of Schapire and Singer [22] and utilising the most recent theoretical re-
results.

In boosting, classifiers combined into the ensemble are often called weak clas-
sifiers referring to the fact that the performance of these classifiers is boosted to a
stronger classifier. They are therefore in some sense weaker than the final classi-
fier. The final classifier is then usually referred to as a strong classifier although
Algorithm 1 The AdaBoost algorithm.

Given: \((x_1, y_1), \ldots, (x_m, y_m)\); \(x_i \in \mathcal{X}, y_i \in \{-1, 1\}\)
Initialise weights \(D_1(i) = 1/m\)

For \(t = 1, \ldots, T\):

1. Find \(h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j; \quad \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]\)

2. If \(\epsilon_t \geq 1/2\) then stop

3. Set \(\alpha_t = \frac{1}{2} \log \left( \frac{1+\epsilon_t}{1-\epsilon_t} \right)\)

4. Update

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

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]

in some degenerated cases it does not need to be stronger than the used weak ones. Especially in some statistically related texts instead of using the weak and strong classifier terminology both classifiers are called hypothesis and the weak classifiers are often called basis hypothesis.

2.1 The algorithm

The basic scheme of the AdaBoost algorithm is depicted in Algorithm 1. The goal of the AdaBoost is to train a classifier using a set of examples. First, a weight \(D_1(i)\) is assigned to each training example. Learning then proceeds in a simple loop. At time \(t\), the algorithm selects a weak classifier \(h_t\) minimising a weighted error on the training set (Step 1). The loop is terminated if this error exceeds \(1/2\) (Step 2). The value of \(\alpha_t\) is computed next (Step 3) and the weights are updated according to the exponential rule (Step 4). In Step 4, \(Z_t\) is a normalisation factor which assures \(D_{t+1}\) remains a distribution. The final decision rule is a linear combination of the selected weak classifiers weighted by their coefficients. The classifier decision is given by the sign of the linear combination.

There are two properties of AdaBoost exploited in the following sections. First, as has been shown in [22] the algorithm minimises an upper bound on the
classification error $\varepsilon_{tr}(H)$ on the training set

$$
\varepsilon_{tr}(H) \leq \prod_{t=1}^{T} Z_t = \frac{1}{2^T} \prod_{t=1}^{T} \sqrt{\varepsilon_t (1 - \varepsilon_t)} \tag{8}
$$

This upper bound is minimised by selecting a weak classifier with the smallest weighted error on the training set as done in Step 1 and by setting its coefficient as done in Step 3.

Second, the re-weighting scheme assures that the updated distribution satisfies

$$
\sum_{i=1}^{m} D_{t+1}(i) u_{t,i} = 0 \tag{9}
$$

where $u_{t,i} = h_t(x_i) y_i$. After rewriting a weighted error in Step 1 of the AdaBoost algorithm to the equivalent form

$$
h_{t+1} = \arg \max_{h_q \in \mathcal{H}} \sum_{i=1}^{m} D_{t+1}(i) u_{q,i} \tag{10}
$$

it is evident that the selected $h_{t+1}$ is “maximally independent” of the mistakes made by $h_t$ enveloped in $u_{t,i}$ [22].

Note also that equation (9) has implications on the weighted error $\varepsilon_{t+1}^{t+1}$ of $h_t$ where the upper index indicates that the error is measured on weights for the step $t + 1$. Since the error can be expresses as

$$
\varepsilon_{t+1}^{t+1} = \frac{1}{2} (1 - \sum_{i=1}^{m} D_{t+1}(i) u_{t,i}), \tag{11}
$$

from equation (9) the error $\varepsilon_{t+1}^{t+1} = 1/2$. The weak classifier $h_t$ is therefore equivalent to the random guess at time $t + 1$.

Summarising equations (8) - (11), the AdaBoost algorithm minimises the upper bound on the classification error, selects weak classifiers with the smallest weighted error, and the weak classifier selected at time $t$ is maximally independent of the weak classifier selected at time $t - 1$. These properties are used later in Section 6.

### 3 Face detection overview

The face detection problem is pretty old and many algorithms have been proposed. Therefore, only few latest approaches often used for comparison will be
mentioned here. For a detailed survey of the older methods see e.g. [32]. Applications of the face detection are numerous. The range covers applications for the security systems, novel human-computer interfaces, compression of the video signal for video-conferencing, successful face detection is a first step to the automatic face recognition, etc..

The main difficulty in the face detection is the high variability of the face appearance in the images. The image of a face depends on the person, his or her facial expression, head position relative to the camera, lightning conditions and many other aspects. Moreover, a face detector has to be able to recognise that an image is not a face. However, the class of the non-face images is very complex and hard to handle. Mainly due to the high variability of the face class and complexity of the non-face class, the face detection is a hard problem which is not sufficiently solved yet.

Rowley et al. [18, 20, 19] built a neural network based face detector. A neural network is trained to classify a $20 \times 20$ image. For detection of faces of different scales and at different positions the image is repetitively subsampled and scanned throughout every location in every such image. Multiple responses of the detector are then merged. For detection of rotated faces, Rowley proposed a two stage algorithm. First, a neural network determines the rotation of the face (for non-face images, arbitrary rotation is returned), the image is derotated and in the second stage frontal face detector is run. Similar approach was used for head pose, however better results were obtained by dividing the task into several subproblems for different poses. The Rowley’s detector is quite accurate but is very slow. However, the two stage approach is an improvement over the previous approaches scanning the image at every position, scale and rotation.

Schneiderman et al. [25, 24] adopted a fully Bayesian approach. The final decision rule is a simple likelihood ratio test. Both class conditional density functions are modelled as a product of a big number of likelihoods of a single visual attribute where the attributes are assumed independent. The likelihoods are modelled as histograms. The visual attributes used are based on the quantised wavelet coefficients to allow localisation of the attributes in space, scale and orientation. To collect a representative set of the visual attributes the AdaBoost algorithm is used. The detector is able to detect either frontal faces or profiles but is again relatively slow.

Yang et al. [31] use the SNoW architecture to build a classifier. The SNoW (Sparse Network of Winnows) architecture is similar to the perceptron but with very high number of inputs (possibly infinite). Few of them are ”active” and the rest ”inactive”. The inputs correspond to the features in the example images. Measured values in an image determine which input features becomes active. Weights of connections between active inputs and output are summed and thresholded. If a prediction mistake is made the weights are increased or decreased, depending on
the type of mistake (missed detection or false alarm). The weighted sum is similar to the AdaBoost algorithm, yet the feature set reduction is not so immense. The speed of the evaluation is higher but still does not allow a real-time performance.

Another very simple and almost directly Bayesian approach to the frontal face detection was proposed by Liu [15]. The face class is modeled as a multivariate normal distribution. The same model is applied to the non-face class, however only for the non-face samples close enough to the face class. Since the class conditional probabilities are known after training, the Bayes decision rule is applied. For the classification of the rest of the non-face samples a simple threshold on the distance to the face class is used. The important property of the approach is the fast non-face samples classification however the detection is rather slow. Nevertheless, the reported results are very good.

The first really real-time face detector was proposed by Viola and Jones [27]. The Viola and Jones frontal face detector consists of several classifiers trained by the AdaBoost algorithm that are organised into a decision cascade. Each cascade stage classifier is set to reach a very high detection rate and an “acceptably” low false positive rate. Since it is trained on the data classified as a face by the previous stages, the final false positive rate is very low and the final detection rate remains high. Since this algorithm was adopted as a basis for our detector it is described in more detail in Section 4. Besides similar detection rates as the previous approaches, the main advantage of Viola and Jones algorithm is the real-time detection.

Later, Viola and Jones extended their work also to multi-view face detection [7]. The used approach is similar to the work of Rowley et al.. A decision tree is trained to find a head pose and when the pose is known a face detector corresponding to this pose is evaluated. The paper demonstrates that the Viola and Jones approach can be extended to the multi-view face detection without any substantial speed reduction.

The Viola and Jones algorithm was also extended to the multi-view face detection by Li et al. [11]. Instead of using discrete AdaBoost, the real version was used and the algorithm was modified to exclude some of already found weak classifiers to overcome non-monotonicity problem of the greedy selection method. To detect a face independent of the head pose, a pyramid of coarse to fine detectors is trained. The algorithm runs in real-time.

Only for the works of Viola and Jones, and Li et al. a real-time performance is reported. All other approaches are often more accurate but with a large penalty in speed. The main attribute contributing to the real-time performance of the methods is the sequential classifier evaluation. Even Rowley et al. used a simple version of sequential decision making in their two stage architecture and in the approach by Liu the non-face samples are classified as soon as possible. Nevertheless, even in the approaches performing in real-time the lack of a deeper theory working
Figure 1: Examples of the Viola and Jones features. The squares represent a face frame. The value of the filter based on a given feature is computed as the sum over filled rectangle(s) minus the sum over empty rectangle(s). Using parity and threshold on this value a class is predicted.

properly with the time parameter is evident.

4 Adopted face detection approach

Recently, Viola and Jones [27] introduced an impressive face detection system capable of detecting faces in real-time with both high detection rate and very low false positive rates. The desirable properties are attributed especially to the efficiently computable features used, the AdaBoost learning algorithm, and a cascade technique adopted for decision making. This approach was adopted as a basis for our face detector.

Features

The features used by Viola and Jones are reminiscent of Haar basis functions. They operate on the gray level images and their decision depends on the parity of a thresholded difference of sums computed over rectangular regions (see Figure 1). Viola and Jones use three kinds of features. They differ by their division into a two, three or four rectangular areas. A value of each feature is computed as a difference of the sum over the white and black regions depicted in Figure 1. Every feature is characterised by its position in the face frame, prespecified size and type. Other two types of features can be easily generated by rotating the first two types by 90°.

An extended set of features was proposed by Lienhart and Maydt [13] for frontal face detection, and by Viola and Jones [7] and Li et al. [11] for multi-view face detection. This type of features can be easily evaluated due to a structure
called integral image [27]. Only four array lookups are needed to find a sum over a rectangular region in the image. To evaluate the original three types of features maximally 9 lookups have to be made.

Each weak classifier then relies on a single feature (given by position in the face frame, its size and type). The parity and threshold are parameters of each weak classifier and they must be recomputed at each cycle of the AdaBoost to minimise the number of misclassified samples weighted by the current distribution.

**Cascade building**

Instead of training a single classifier, a cascade of classifiers is built. An image window (region) is passed to the first classifier. It is either classified as a non-face or a decision is deferred and the image is passed to the second, etc. classifier. The goal of each classifier is to prune the training set for the next stage classifier of the cascade. Since easily recognisable non-face images are classified in the early stages, classifiers of the later stages of the cascade can be trained rapidly only on the harder, but smaller, part of the non-face training set.

The cascade building is described in Algorithm 2. Inputs to the algorithm are: the desired false positive rate $f$, the detection rate $d$ of the cascade stages, and the final false positive rate of the cascade. Each stage is trained until $f$ and $d$ is reached. Since AdaBoost is neither designed to reach low false positive rates nor high detection rates, a threshold is adjusted ex post.

In the cascade classifier, the overall false positive and detection rates are a product of the rates of individual stages. The pruning process is asymmetric and concentrates on the non-face images. The stage false positive $f$ is usually set to higher values. The multiplication in Step 2 guarantees an exponential reduction of the overall false positive rate. The detection rate must be set close to one to ensure that final $D$ is high.

The cascade evaluation is equivalent to a sequential classification using a degenerated decision tree. When the current stage classifier labels a region in an image as a non-face, the decision process is terminated. Otherwise, the next stage classifier is run. A region is declared a face if it is accepted by all classifiers in the cascade.

Face detection is done by moving the cascade detector across the image at multiple scales and locations. A typical image contains only a small number of face regions compared to the number of regions scanned. Due to early termination of the decision process in non-face regions, only few stages of the cascade are evaluated on average [27, 12]. Hence, the speed of evaluation depends heavily on the computational complexity and rejection rates of the first few stages. The enhanced AdaBoost learning algorithms proposed in Section 6 and 7 produces a
Algorithm 2 Building the cascade.

Input: Allowed false positive rate \( f \), and detection rate \( d \); final false positive rate \( f_{\text{final}} \)

\[
F_0 = 1, \quad D_0 = 1
\]

Do until \( F_i > f_{\text{final}} \)

1. Train a classifier until \( f_{\text{reached}} < f \) and \( d_{\text{reached}} > d \) on the validation set
2. \( F_{i+1} = F_i \times f_{\text{reached}} \)
3. \( D_{i+1} = D_i \times d_{\text{reached}} \)
4. Throw away misclassified faces and generate new non-face data from non-face images

classifier that, for a given detection and false positive rates, is more likely to make a decision early in the evaluation of the cascade.

5 Data sets

The datasets used for training and test purposes in the experiments in the subsequent sections are described here.

5.1 Training data set

The data for training were collected from various sources. Face images are taken from the MPEG7 face dataset [6]. The dataset contains face images of variable quality, different facial expressions and taken under wide range of lightning conditions, with uniform or complex background. The pose of the heads is generally frontal with slight rotation in all directions. Eyes and the nose tip are aligned in all images. The dataset contains 3176 images, one image was removed due to severe distortion.

Pose variability was added synthetically to the data. The images were randomly rotated by up to \( 5^\circ \), shifted up to one pixel and the bounding box was scaled by a factor of \( 1 \pm 0.05 \). Two datasets, training and validation, of the same size as the original dataset were created by the perturbations.

Non-face images were collected from the web. Images of diverse scenes were included. The dataset contains images of animals, plants, countryside, man-made objects, etc.. More than 3000 images were collected and random sub-windows
used as non-face examples.

5.2 Test data set

The classifiers are tested on the MIT+CMU data set [19]. This data set is widely used for the face detectors comparison [19, 25, 27]. The facial features position ground truth information is provided together with the data set. However, the data set contains several disputable faces. Thus the results reported in the papers differ not only by the detection rates but by a subset used for the algorithm evaluation. Moreover, no standard evaluation procedure is given and consequently the results are influenced by the method of successful detection recognition.

These problems are partially overcome by the fact that the results reported in this thesis proposal focus mainly on the speed of the evaluation and not the detection rates. Besides, the direct comparison with the Viola and Jones approach is provided.

The data set consists of the parts A, B, C and the rotated faces images part. The parts A, B and C contain mainly frontal faces and therefore are used in our tests. The used parts consist of 130 images containing 511 faces in total.

6 AdaBoost with totally corrective updates

The totally corrective algorithm with coefficient updates (TCAcu) [28] is based on the AdaBoost algorithm described in Section 2 and its structure is depicted in Algorithm 3. Schapire and Singer’s [22] notation is used and the algorithm differs from Algorithm 1 only by an additional Step 5.

AdaBoost constructs the classifier as a linear combination of weak classifiers chosen from a given, finite or infinite, set. Its goal is to choose a small number of weak classifiers and assign them proper coefficients. The linear combination can be seen as a decision hyper-plane in the weak classifier space. Hence, AdaBoost can be viewed as an optimisation procedure, that operates in the space of weak classifier coefficients, starting with a zero vector and ending with a vector with only small number of non-zero elements.

The standard (discrete) AdaBoost is a greedy algorithm, that in each step sets one zero-valued coefficient to a non-zero value. Because of its greedy character, neither the found weak classifiers nor their coefficients are optimal.

A totally corrective algorithm with coefficients updates (TCAcu) firstly proposed in [28] and described in this section differs from the standard AdaBoost in two main aspects. Firstly, in the standard AdaBoost, a newly added weak classifier can be shown to be “independent” in a precisely defined way of the previously added weak classifier. The TCAcu algorithm finds a new weak classifier that is
independent of all weak classifiers selected so far. Secondly, the coefficients of already found weak classifiers are updated repetitively during the learning process. It is shown that these modifications minimise the classification error upper bound more aggressively and that shorter classifiers are found.

The term “totally corrective algorithm” was introduced by Kivinen and Warmuth [9]. However, the Kivinen and Warmuth algorithm did not update the coefficients of weak classifiers appropriately. The algorithm thus lost the important property of minimisation of the upper bound on the training error. Kivinen and Warmuth made no empirical evaluation of the algorithm. The algorithm was experimentally tested by Oza on several standard problems with poor results [16].

Another attempt to shorten the final classifier and find more compact set of weak classifiers was proposed by Li et al. [11] and was motivated by the feature selection view of AdaBoost. In case, the weak classifiers correspond directly to the features as in the Viola and Jones face detection framework, changing one coefficient to a non-zero value effectively selects this feature [27]. Li et al. proposed FloatBoost, a modification of AdaBoost where some of already non-zero coefficients are set back to zero when it leads to a lower upper bound on the classification error. Instead of the greedy feature selection, the sequential floating forward selection (SFFS) technique [17] is used. Li et al. show that this modification leads to shorter classifiers.

The main contribution of this section is (1) a modification of AdaBoost algorithm which leads to shorter classifiers and a speedup of classification, (2) the introduction of totally corrective algorithm to face detection training. It is shown that resulting classifier performance is comparable to the standard AdaBoost and the resulting classifier runs faster.

### 6.1 Totally corrective step

The independence property discussed in Section 2.1 is very attractive from the feature selection point of view. A question arises whether a new weak classifier, maximally independent of all already selected ones, can be found. In such case, the distribution $D_{t+1}$ must satisfy

$$\sum_{i=1}^{m} D_{t+1}(i) u_{q,i} = 0 \quad \text{for } q = 1, \ldots, t$$

(12)

where $u_{q,i} = h_q(x_i) y_i$.

There is no closed-form solution to the system of equations (12) and sometimes an exact solution does not even exist [9]. This is a consequence of the non-negativity constraint on $D_{t+1}$, which is a distribution. Therefore, TCS is designed as an iterative optimisation algorithm.
**Algorithm 3** TCAcu: Totally Corrective Algorithm with coefficient updates.

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, 1\}\)

Initialise weights \(D_1(i) = 1/m\)

For \(t = 1, \ldots, T:\)

1. Find \(h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j; \quad \epsilon_j = \sum_{i=1}^m D_t(i) I[y_i \neq h_j(x_i)]\)

2. If \(\epsilon_t \geq 1/2\) then stop

3. Set \(\alpha_t = \frac{1}{2} \log\left(\frac{1+\epsilon_t}{1-\epsilon_t}\right)\)

4. Update

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

5. *Totally corrective step* (see Algorithm 4)

Output the final classifier:

\[
H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)
\]

---

**Algorithm 4** The Totally Corrective Step

Initialise \(\tilde{D}_0 = D_t\)

For \(j = 1, 2, \ldots, J_{\text{max}}\)

1. \(q_j = \arg \max_{q=1, \ldots, \epsilon_q} |\epsilon_q - 1/2|\).

2. If \(|\epsilon_{q_j} - 1/2| < \Delta_{\text{min}}\) exit the loop.

3. Let \(\hat{\alpha}_j = 1/2 \ln((1 - \epsilon_{q_j})/\epsilon_{q_j})\).

4. Re-weight

\[
\tilde{D}_{j+1}(i) = \frac{1}{Z_j} \tilde{D}_j(i) \exp(-\hat{\alpha}_j u_{q_j, i})
\]

5. \(\alpha_{q_j} = \alpha_{q_j} + \hat{\alpha}_j\)

Assign \(D_{t+1} = \tilde{D}_j\)
Figure 2: Weighted errors $\epsilon_t$ (eq. (11)) of weak classifiers $h_t$ after ten iterations ($T=10$) for AdaBoost (circles) and TCAcu (crosses). In AdaBoost, $\epsilon_t$ for all but the last weak classifier are arbitrary. In TCAcu, all errors satisfy $|\epsilon - 0.5| < \Delta_{\text{min}}$. Note that the selected weak classifiers may be different for AdaBoost and TCAcu.

In AdaBoost, equation (12) holds at time $t$ (after re-weighting, Step 4) only for $q = t$. A typical situation is depicted in Figure 2. Weak classifier errors are shown after step $t = 10$ of AdaBoost. The weak classifier errors differs from 0.5 except for the last (10th) weak classifier.

Another observation can be made about the change of the upper bound. From equation (8), we see that the upper bound is reduced if the error of a newly added weak classifier differs from 0.5. The bigger the difference, the bigger the reduction of the upper bound. It follows that the upper bound can be further reduced by formally adding an already used weak classifier, if its error differs form 0.5. This addition has two important consequences.

First, because of the linear combination form of the final classifier, addition of an already used weak classifier $h_r$, $r < t$, requires only a change of $\alpha_r$ coefficient, not a change of the final classifier size. A new coefficient is computed as $\alpha_r = \alpha_r^r + \alpha_r^{t+1}$, where the upper indexes express the cycle in which the coefficient was computed.

Second, a new distribution obtained by this addition satisfies equation (12) for $q = r$, but not for any other $q$. If equation (12) is approximately satisfied for all $q$ the goal is reached. If not, another $q$ is selected and $h_q$ ”virtually added”. Each such addition will lower the upper bound.

TCS is formally summarised in Algorithm 4. At time $t$, a distribution $D_t$ is used to initialise the algorithm. In each iteration a weak classifier is selected
from the already used ones so that the absolute difference of its error and 0.5 is maximised. The standard scheme is used to find $\alpha_j$ and the new distribution $D_{j+1}$. The value $\alpha_j$ is added to the corresponding coefficient and the loop is repeated.

Since an exact solution may not exist, the computation is terminated if a close enough solution is found or if the maximum allowed number of iterations is reached. The final distribution is then used in cycle $t + 1$ of AdaBoost learning. A typical result of the algorithm is depicted in Figure 2.

Convergence properties of the TCS step and standard AdaBoost are the same. The only difference is in the set of weak classifiers which in TCS is limited to the already selected ones in the main AdaBoost loop.

A similar algorithm was proposed by Kivinen and Warmuth [9]. TCACu differs from Kivinen and Warmuth algorithm in two important aspects: (1) the coefficients of weak classifiers are updated appropriately, (2) the property of minimisation of the upper bound is kept. The Kivinen and Warmuth algorithm was experimentally tested by Oza on several standard problems [16] with poor results.

### 6.2 Experiments

The performance of TCACu and AdaBoost was compared on the face detection problem described in Section 5.2 and trained on the data described in Section 5.1. The training process and obtained results are discussed next. The performance evaluation concentrates on the speed and complexity of the learned cascaded classifiers.

#### Training process

During the training process, the training and validation dataset are updated for each stage (cf. Algorithm 2). The non-face part of the training and validation datasets consist of 5000 randomly selected regions from the non-face images. Only regions that were not rejected by previous stages of the cascade are included. The face set remains almost the same over the whole training. The faces rejected by some of the stage classifiers are removed, but the cascade is build to ensure that these false rejects are just a small fraction of the face data.

The process is driven by the stage false positive, detection and final false positive rates. In the reported experiments, the values were set to 0.4 stage false positive rate, 0.999 detection rate and 0.0001 the final false positive rate. The final false positive rate was reached in stage eight in TCACu and in stage ten in AdaBoost.
Figure 3: Selectivity comparison. Horizontal axis: the complexity of the cascaded classifier expressed by the number of weak classifiers used. Vertical axis: number of weak classifier evaluations on the MIT+CMU dataset.

<table>
<thead>
<tr>
<th>Number of stages</th>
<th>Stage classifier length</th>
<th>Number of evaluations</th>
<th>False negatives</th>
<th>False positives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AB</td>
<td>TCAcu</td>
<td>AB</td>
<td>TCAcu</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12431151</td>
<td>12431151</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
<td>4009205</td>
<td>3757632</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>9</td>
<td>1643072</td>
<td>1083123</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>11</td>
<td>823246</td>
<td>512004</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>10</td>
<td>435483</td>
<td>183962</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>22</td>
<td>201982</td>
<td>49814</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>33</td>
<td>100887</td>
<td>9052</td>
</tr>
<tr>
<td>7</td>
<td>23</td>
<td>36</td>
<td>52867</td>
<td>3173</td>
</tr>
<tr>
<td>8</td>
<td>25</td>
<td>55</td>
<td>27504</td>
<td>1149</td>
</tr>
<tr>
<td>9</td>
<td>31</td>
<td></td>
<td>14818</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>40</td>
<td></td>
<td>7568</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>47</td>
<td></td>
<td>4194</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>55</td>
<td></td>
<td>2602</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>49</td>
<td></td>
<td>1828</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Comparison of AdaBoost (AB) and TCAcu performance.
Results

The classifiers were tested on the MIT+CMU dataset [19] described in Section 5.2. The main objective of the experiments is to demonstrate the detection speedup in comparison with the classical Viola-Jones approach, rather than improvement of the detection rate per se. This means that we did not try to find e.g. the optimal sets of weak classifiers since this is not important for a fair comparison of AdaBoost and TCACu.

The results for the cascades trained by the two variants of AdaBoost are summarised in Table 1. For each number of stages in the cascade, the following quantities are recorded (left to right in Table 1): the number of weak classifier forming a stage in the cascade, the total number of evaluations in each stage, and the false negative and false positive rates on the MIT+CMU dataset. The first row, the "0th stage", is added only to show the total number of regions scanned.

It can be observed that for both algorithms the complexity of stages increases gradually except for two small fluctuations. At the beginning the growth of TCACu is slower and it changes after four stages. However, the complexity is not the only important factor determining the speed of face detection. Also the number of regions marked as a potential face in each stage is significant. It can be seen that TCACu discards many more regions in early stages than AdaBoost. This early pruning influences false positive and false negative rates that are shown in the last two columns. These two rates measure the performance of the cascaded classifier. The table shows that both algorithms lead to similar false positive and false negative rates, but TCACu converges much faster.

To compare the speed of the cascades trained by TCACu and AdaBoost, the number of weak classifiers evaluated on MIT+CMU dataset was measured. All regions have to be evaluated by the first stage classifier. The number of evaluations is consequently a product of the number of regions and the length of the first stage classifier. The same holds for the second (and higher) stage classifier, but only regions not rejected by the first (previous) stage(s) are evaluated. Summing the numbers evaluations of the first and the second stage gives the number of evaluations of the two-stage cascade classifier. The result for all lengths of the cascade and for both algorithms is depicted in Figure 3.

Figure 3 demonstrates two important phenomena. First, the complexity of the cascades with comparable false negative and false positive rates is up to four times smaller for the TCACu algorithm (six-stage TCACu vs. eleven-stage AdaBoost). Second, the number of evaluations needed in AdaBoost is higher by 20% than in TCACu.
6.3 Conclusions

A new extension of the AdaBoost algorithm was proposed and compared with the state-of-the-art Viola and Jones face detection algorithm. The proposed TCAcu algorithm finds the final classifier by aggressive minimisation of the upper bound on the training error and produces a significantly shorter classifier. The obtained results are comparable to the Viola and Jones method in terms of detection and false positive rates. The classifier trained by the novel method was about 20% faster and consists of only a quarter of the weak classifiers needed for a classifier trained by standard AdaBoost.

The algorithm can be applied with other weak classifiers suitable for face detection and in conjunction with FloatBoost-like feature selection techniques. The reduction of the number of weak classifiers can be important in areas where the weak classifiers are expensive to compute or to implement, e.g. on smart cards or other special purpose hardware.

7 Using previous stage knowledge

In this section, a novel interpretation of the cascade training is proposed. In the Viola and Jones approach, each stage of the cascade classifier is trained from scratch. In each stage a new classification problem is solved. So far trained cascade classifier is used to generate new training data only and the effort already put in finding the classification boundary is abandoned. Unlike in the Viola and Jones approach, in the proposed perspective, a new stage training is seen as finding a more precise approximation of the decision boundary. This modified view can be naturally integrated into the AdaBoost learning of each cascade stage by using previous-stage knowledge.

The face detection problem is used to experimentally verify a hypothesis that using the previous-stage knowledge leads to shorter stage classifiers and consequently to faster face detector.

7.1 Proposed method

In the cascade building algorithm proposed by Viola and Jones [27], more and more specialised classifiers are trained in each stage. Every classifier is trained on data on which a decision has not been reached by the previous stage classifiers. The AdaBoost decision rule used in the Viola and Jones cascade building algorithm is of the form

\[ H_s(x) = \text{sign}(f_s(x) - \gamma_s) \]  

(13)
Figure 4: Histogram of values of $f_s(x)$ in a model stage $s$. A threshold $\gamma_s$ determines the false positive and false negative rate of the stage (see equation (13)). Peaks are caused by the discrete version of AdaBoost used.

$$f_s(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

(14)

where $s$ is the stage number, $T$ is the length of the classifier, $h_t$ is combined weak classifier and $\alpha_t$ its coefficient. The $\gamma_s$ parameter is a threshold adjusted to reach the specified detection rate and false positive rate. It is used to remove substantial part of the non-face examples while keeping majority of face examples.

In the Viola and Jones approach, after a stage classifier is trained, new training set is generated and the AdaBoost algorithm is run on this new problem to generate a classifier. AdaBoost generates a classifier as if no previous training has been done.

**Using previous-stage knowledge**

The parameter $\gamma_s$ in each stage is set to keep the majority of the face examples while rejecting a substantial part of the non-face ones. Nevertheless, a part of the non-face examples is kept for training of the next stage. The current stage classifier can still be very good on these non-face examples. However the bias towards correct face example classification leaves this information unused.

Figure 4 displays a typical situation during the cascade building, at stage $s$. 

21
Face and non-face examples are already sufficiently separated, so $\gamma_s$ can be found such that only small fraction of the face examples is misclassified and, concurrently, large part of the non-face examples is correctly classified. The non-face examples on the left of the $\gamma_s$ are replaced by newly sampled examples from the right side of the $\gamma_s$ in the next stage training. It can be seen that the stage $s$ classifier is still reasonably good classifier on the new training set if a new threshold is found. The previous stage classifier can be therefore further used as a very good starting point for the next stage training.

The AdaBoost algorithm runs in cycles. In each cycle, a new weak classifier with the smallest weighted error on the training set is added to the sum in equation (14). The proposed algorithm uses the previous-stage knowledge by constructing an additional weak classifier. Since the previous-stage knowledge is fully encompassed in the $f_s$ function, this function is used with different threshold to originate the weak classifier. The previous-stage weak classifier is inserted into the classifier at the beginning of the AdaBoost learning with coefficient $\alpha_0$ found by the AdaBoost algorithm. The AdaBoost decision rule then becomes

$$H_s(x) = \text{sign}(f_s(x) - \gamma_s)$$ (15)

$$f_s(x) = \alpha_0 H'_{s-1}(x) + \sum_{t=1}^{T} \alpha_t h_t(x)$$ (16)

where

$$\alpha_0 = \frac{1}{2} \ln \left( \frac{1 - \epsilon_0}{\epsilon_0} \right)$$

and $H'_{s-1}$ is the previous-stage weak classifier with a threshold $\tau_s$

$$H'_{s-1}(x) = \text{sign}(f_{s-1}(x) - \tau_s).$$

The weighted error $\epsilon_0$ is computed as for an ordinary weak classifier as

$$\epsilon_0 = \sum_{i=1}^{m} D_0(i) [\text{sign}(f_{s-1}(x_i) - \tau_s) \neq y_i]$$

where $D_0(i)$ is a weight of the $i$th example $(x_i, y_i)$ and $m$ is the size of the current training set. The threshold $\tau_s$ is set to minimise $\epsilon_0$.

Decision to insert the previous-stage weak classifier to the stage classifier as first has several reasons. Since the weights of the training examples are initialised to the uniform distribution ($D_0(i) = 1/m$), they do not influence the performance of this weak classifier. Therefore, the weak classifier is very strong compared to the other simple weak classifiers (it is already boosted). It also helps to focus AdaBoost learning to parts of the problem not learned by the previous stages.
An important property of such weak classifier is its zero evaluation cost. In both training and detection phase, a stage has to be evaluated to find out whether the next stage should be trained (in training) or used (in detection) on a given example. On the example $x$, $H_s(x)$ has to be evaluated and ergo the value of $f_s(x)$ is known. Comparing this value with different threshold $\tau_s$ is very cheap operation compared to the evaluation of any other weak classifier and can be regarded as a zero-cost.

### 7.2 Experiments

The performance of the cascade building with previous-stage knowledge and the cascade building of Viola and Jones was compared on the face detection problem described in Section 5.2 and trained on the data described in Section 5.1. The training process and obtained results are discussed next. The performance evaluation concentrates on the speed and complexity of the learned cascaded classifiers.

#### Training process

During the training process, the training and validation dataset are updated for each stage (cf. Algorithm 2). The non-face part of the training and validation
datasets consist of 5000 randomly selected regions from the non-face images. Only regions that were not rejected by previous stages of the cascade are included. The face set remains almost the same over the whole training. The faces rejected by some of the stage classifiers are removed, but the cascade is build to ensure that these false rejects are just a small fraction of the face data.

The process is driven by the stage false positive, detection and final false positive rates. In the reported experiments, the values were set to 0.4 stage false positive rate, 0.999 detection rate and 0.0001 the final false positive rate. The final false positive rate was reached in the stage ten in both algorithms.

**Results**

The classifiers were tested on the MIT+CMU dataset [19] described in Section 5.2. The main objective of the experiments is to demonstrate the detection speedup in comparison with the classical Viola and Jones approach, rather than improvement of the detection rate per se. This means that we did not try to find e.g. the optimal sets of weak classifiers since this is not important for a fair comparison of the methods.

The results for the cascades trained by the Viola and Jones approach and by the proposed approach using the previous-stage knowledge are summarised in Table 2. For each number of stages in the cascade, the following quantities are recorded (left to right in Table 2): the number of weak classifier forming a stage in the

<table>
<thead>
<tr>
<th>Number of stages</th>
<th>Stage classif. length</th>
<th>Number of evaluations</th>
<th>False negatives</th>
<th>False positives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V&amp;J</td>
<td>PSK</td>
<td>V&amp;J</td>
<td>PSK</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
<td>12431151</td>
<td>12431151</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>5</td>
<td>4009205</td>
<td>4009205</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>11</td>
<td>1643072</td>
<td>1609004</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>7</td>
<td>823246</td>
<td>806185</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>13</td>
<td>435483</td>
<td>385159</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>15</td>
<td>201982</td>
<td>179603</td>
</tr>
<tr>
<td>7</td>
<td>23</td>
<td>14</td>
<td>100887</td>
<td>80642</td>
</tr>
<tr>
<td>8</td>
<td>25</td>
<td>19</td>
<td>52867</td>
<td>39324</td>
</tr>
<tr>
<td>9</td>
<td>31</td>
<td>16</td>
<td>27504</td>
<td>19671</td>
</tr>
<tr>
<td>10</td>
<td>40</td>
<td>31</td>
<td>14818</td>
<td>10179</td>
</tr>
</tbody>
</table>

Table 2: Comparison of the cascade building with previous-stage knowledge (PSK) and the original Viola and Jones algorithm (V&J) performance on MIT+CMU dataset.
cascade (the previous-stage weak classifier is not included, since its evaluation costs nothing), the total number of evaluations of each stage, and the false negative and false positive rates on the MIT+CMU dataset.

It can be observed that using previous-stage knowledge leads to the shorter stage classifiers. The saving is 25–50%. Only the first stage remains the same, since there is no previous stage to use. Obvious improvement is to shorten this stage to a minimal length and use its stage-knowledge in the next stage building. Similar approach was taken by Viola and Jones, but without the previous-stage knowledge incorporation. Another observation can be made about the false negative and false positive rates. The false negative rate is almost the same for both methods. However, the false positive rate is slightly better when using previous-stage knowledge. Therefore, using previous-stage knowledge leads to slightly lower classification error.

To compare the speed of the cascades trained using previous-stage knowledge and by the Viola and Jones approach, the number of weak classifiers evaluated on MIT+CMU dataset was measured. All regions have to be evaluated by the first stage classifier. The number of evaluations is consequently a product of the number of regions and the length of the first stage classifier. The same holds for the second (and higher) stage classifier, but only regions not rejected by the first (previous) stage(s) are evaluated. Summing the numbers evaluations of the first and the second stage gives the number of evaluations of the two-stage cascade classifier. The result for all lengths of the cascade and for both algorithms is depicted in Figure 5.

Figure 5 demonstrates two important phenomena. First, the complexity of the cascade classifiers with the same number of stages is about 30% smaller when using previous-stage knowledge. Second, the number of evaluations needed in original cascade building is higher by 25% than when the previous-stage knowledge is used.

7.3 Conclusions

A modification of the cascade building for the AdaBoost algorithm was proposed and compared with the Viola and Jones algorithm. The proposed algorithm is based on a novel view of the cascade building process which is seen as an algorithm for gradually finding more precise decision boundary in each stage. A rule incorporating the previous-stage knowledge into a new stage classifier to better focus the learning on the examples on which a decision has not been reached by the previous stage classifier was proposed. As was shown, using previous-stage knowledge leads to shorter stage classifiers with the same false positive and false negative rate without increase in their computational complexity. The cascade trained by the proposed method was about 30% shorter and 25% faster compared
to the cascade trained by the Viola and Jones algorithm.

Since the proposed algorithm gives almost the same results as the Viola and Jones algorithm and the resultant classifier is faster, it could be used instead of the original one without any disadvantages. The reduction of the number of weak classifiers can be important in areas where the weak classifiers are expensive to compute or to implement, e.g. on smart cards or other special purpose hardware.

8 Summary and thesis proposal

In this thesis proposal methods for fast face detection using the AdaBoost algorithm are explored and two methods improving previous results have been proposed. The totally corrective algorithm with coefficients updates [28] speed up the classification part by minimising the training error upper bound more aggressively. It leads to shorter classifiers and hence to faster classification. The second method, using previous-stage knowledge [29], concentrates on the speed of training. A novel view of the training a cascaded classifier allows not to lost the information learned by one cascade stage but to use it in the consequent training.

A classifier training for the face detection is only one example of whole group of two class example-based learning problems with different measurement costs for the classes, where time of the classifier evaluation is limited or where the evaluation time is the optimisation parameter. In the future research we would like to study these problems even more theoretically. However, the face detection is difficult but important problem and it will remain as our test example. In the given time of the PhD study, we would like to touch the following problems:

- **Fully sequential learning and classification.** Using the previous-stage knowledge is the first step towards a fully sequential learning and classification. The goal is to devise an algorithm for training a single classifier, which can be evaluated sequentially. The results presented in Section 7 suggest a direction of future research. Moreover, Friedman et al. [5] showed that asymptotically the AdaBoost converges to the likelihood ratio used in the sequential analysis studied by Wald [30]. The sequential analysis gives direct recipe for sequential evaluation, which have to be modified for the case of sequential learning of an AdaBoost classifier and for the inaccurate estimation of the likelihood ratio during the learning.

- **Using weak classifiers with different complexity.** So far, all weak classifiers have approximately the same strength and very similar complexity. When different types of the weak classifiers are combined together and the classifier is evaluated sequentially a question arises whether it is better to use one complex weak classifier or several simpler ones. The main criterion
here is the evaluation time and the algorithm have to be adapted to work with this new criterion.

- **Substitution of the tail of the cascade by another classifier.** Experiments in Section 6 and 7 show that only few sub-windows are processed by the later stages of the cascaded classifier. This suggest that using more complex but more powerful classifier on these sub-windows would not affect the evaluation time dramatically while the accuracy can be increased. The “tail” of the cascaded classifier would be replaced for instance by SVM trained on the data passed through several stages of the cascaded classifier.

- **Multi-view face detection.** The classifiers discussed in this thesis proposal were designed to be used for frontal face detection. However, the real applications require very often non-frontal faces to be detected as well. Although these problems are similar and often solved as two instances of the same recognition problem a more appropriate method would be to use the nature of the problem, the a priori knowledge of the rigidity of the head. First similar attempts can be found in [18, 11, 7]. Since the multi-view face detection is harder problem the time limitations are even more tight.

- **Coarse-to-fine face detection.** Similarly to the sequential classification of one sub-window a classifier can be trained to be sequential in the sub-window resolution as well. At a very coarse resolution many sub-windows can safely be classified as a non-faces. For the promising sub-windows the classification continue and even the face position (or even rotation or head pose) can be refined. This approach was used by Fleuret et al. [2] and can lead to another speedup of the classification.

- **Use other versions of AdaBoost.** A discrete version of the AdaBoost have been used in all reported experiments. Nevertheless, Lienhart et al. [12] showed in his study on the classification accuracy of the classifiers trained with different AdaBoost variants that a better choose than discrete AdaBoost is a variant called Gentle AdaBoost [5]. Usage of the Gentle AdaBoost would probably require a changes in the classifier sequential learning.
References


