Learning for Sequential Classification

A Dissertation Presented to the Faculty of the Electrical Engineering of the Czech Technical University in Prague in Partial Fulfillment of the Requirements for the Ph.D. Degree in Study Programme No. P2612 - Electrotechnics and Informatics, branch No. 3902V035 - Artificial Intelligence and Biocybernetics, by

Jan Šochman

February 25, 2009

Thesis Advisor

Doc. Dr. Ing. Jiří Matas

Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo náměstí 13, 121 35 Prague 2, Czech Republic
fax: +420 224 357 385, phone: +420 224 357 465
http://cmp.felk.cvut.cz

Abstract

In many computer vision classification problems, both the error rates and the evaluation time characterises the quality of a decision. Yet most of the learning algorithms do not optimise the evaluation time explicitly. We show that such learning problems can be formalised in the framework of sequential decision-making.

The proposed formalisation has been already studied (without learning) by Wald in the 1940's, who proved that the optimal sequential strategy in terms of the shortest average time to decision (number of measurements used) is the sequential probability ratio test (SPRT). We built on the SPRT test and enlarge its capabilities to problems with dependent measurements. We show, how the limitations of SPRT to *a priori* ordered measurements and known joint probability density functions can be overcome. We propose a learning algorithm, called WaldBoost, which handles the evaluation time vs. error rate trade-off during the learning, which integrates the AdaBoost learning algorithm for measurement selection and ordering and the joint probability density estimation, with the optimal SPRT decision strategy.

The WaldBoost algorithm is tested on the face detection problem. The reached detection results are superior to the state of the art methods in the average evaluation time and comparable in the detection rates.

Next, we show how existing (slow) binary decision algorithms can be approximated by a (fast) trained WaldBoost classifier. The WaldBoost learning is used to minimise the decision time of the emulated algorithm while guaranteeing predefined approximation precision. Moreover, we show that the WaldBoost algorithm together with bootstrapping is able to efficiently handle an effectively unlimited number of training examples provided by the implementation of the approximated algorithm.

Two interest point detectors, the Hessian-Laplace and the Kadir-Brady saliency detectors, are emulated to demonstrate the approach. Experiments show that while the repeatability and matching scores are similar for the original and emulated algorithms, a 9-fold speed-up for the Hessian-Laplace detector and a 142-fold speed-up for the Kadir-Brady detector is achieved. For the Hessian-Laplace detector, the achieved speed is similar to SURF, a popular and very fast handcrafted modification of Hessian-Laplace; the WaldBoost emulator approximates the output of the Hessian-Laplace detector more precisely.

Finally, an on-line learning version of the WaldBoost algorithm is proposed. On-line boosting allows to adapt a trained classifier to changing environmental conditions or to use sequentially available training data. Yet, two important problems in the on-line boosting training remain unsolved: (i) classifier evaluation speed optimisation and, (ii) automatic classifier complexity estimation. We show how the on-line boosting can be combined with Wald's sequential decision theory to solve both of the problems.

The properties of the proposed on-line WaldBoost algorithm are demonstrated on a visual tracking problem. The complexity of the classifier is changing dynamically depending on the difficulty of the problem. On average, a speedup of a factor of 5-10 is achieved compared to the non-sequential on-line boosting.

Acknowledgments

I would like to express my thanks to my colleagues at Center for Machine Perception who I had the pleasure of working with. My special thanks go to Prof. Václav Hlaváč for maintaining such a high quality workplace with such a pleasant atmosphere. It was pleasure for me to work and study in his group.

I am greatly indebted to my advisor Jiří Matas for his unflagging enthusiasm, enormous patience, for all the theoretical knowledge and critical view on research he shared with me, and for all his support.

I thank to Prof. Mirko Navara, Alexander Shekhovtsov and Ondra Drbohlav for their helpful comments on the thesis manuscript, and to Michal Perďoch, Ondra Chum, Štěpán Obdržálek, and Zdeněk Kálal for their help with my conference and journal papers.

I also would like to thank to may family and to my friends for all their support that made it possible for me to finish this thesis.

For his limitless and unconditioned support of all sentient beings I would like to thank to lama Ole. I wish all good I gained on both professional and personal levels during my PhD studies may help other beings to reach happiness and to avoid suffering.

I gratefully acknowledge the support of European Commission project FP6-IST-027113 eTRIMS, the Grant Agency of the Czech Technical University under project CTU 0307313, and STINT Foundation project Dur IG2003-2 062.

Contents

Keywords: sequential decision making, AdaBoost, sequential probability ratio test, WaldBoost, machine learning, face detection, interest point detection, tracking

Commonly used symbols

Sequential Decision Making

$\mathbf{x} \in \mathcal{X}$	a classified object
$y \in \{+1, -1\}$	object class label
$\Xi = \{\xi_1, \xi_2, \ldots; \xi_i : \mathcal{X} \to \mathcal{X}_i\}$	a set of ordered measurement functions ξ_i
$x_1, x_2, \ldots; x_i \in \mathcal{X}_i$	measurements on the object $\mathbf{x}$, outputs of the measurement functions ξ_i
$S : \mathcal{X} \to \{+1, -1\}$	a sequential decision strategy
$S_t : \mathcal{X}_1 \times \ldots \times \mathcal{X}_t \to \{+1, -1, \sharp\}$	a decision function
$\sharp$	"still undecided" output
$\bar{T}_{S,y}$	average evaluation time of a sequential strategy S for the class y
α_S, β_S	false negative and false positive rate of a strategy S
α, β	user defined upper bounds on the false negative and the false positive rate
S^*	SPRT sequential decision strategy
R_t	likelihood ratio on t measurements
A, B	SPRT decision thresholds

AdaBoost

$\mathcal{T}$	training set
m	size of the training set
$\mathcal{H}$	set/class of weak classifiers
$h^{(t)} : \mathcal{X} \to \mathbb{R}$	selected weak classifier
$f_t(x) : \mathcal{X} \to \mathbb{R}$	strong classifier response
$H_t(x) : \mathcal{X} \to \{+1, -1\}$	strong classifier classification
$w_t(i), \mathbf{w}_t$	weight of i-th sample, weight vector in training round t
T	number of training rounds, length of the AdaBoost classifier
K	number of bins in a weak classifier

WaldBoost

$\mathcal{P}$	sample pool
N	sample pool size
$\theta_A^{(t)}, \theta_B^{(t)}$	decision thresholds for classifier length t
γ	final threshold

1 Introduction

The thesis connects these two lines of research – the machine learning theory driven by Bayesian risk minimisation and the sequential hypothesis testing. The result of this fusion is a *learning* algorithm which produces a *sequential* classifiers. As shown in the thesis, this fusion also leads naturally to a learning algorithm which can process large training sets effectively. The usefulness of the proposed algorithm is demonstrated on several practical problems.

A common formulation of the pattern recognition problem is that of finding a decision strategy which minimises the Bayesian risk. However, to be able to use the Bayesian formulation, the statistical model has to be fully defined, i.e. the necessary probabilities have to exist and be known. Often our knowledge of the problem is not sufficient to directly minimise the Bayesian risk and the necessary model parameters or a discriminative decision strategy need to be learned.

In supervised learning, some parameters of the statistical model are unknown. The learning algorithm is provided with a training set and the task is to find a decision strategy or the unknown model parameters. Ideally, the learning should minimise the risk, however this is impossible. Instead, the learning task often reduces to training error minimisation with some regularisation, which guarantees generalisation properties of the decision strategy.

In many practical problems, the classification error of the learned decision strategy is only one of the characteristics of a learning algorithm. For instance, when the training data are available progressively, i.e. not all at once, the ability of the learning algorithm to update the decision strategy incrementally is necessary [25, 87, 84]. Also, some learning algorithms are more robust in the presence of outliers in training data [22, 20, 60]. In medical applications, the cost of measurements is not negligible, differs among measurements, and has to be considered. A strategy which can use this information will be preferred [14]. When a very large training set is available, the ability to process it effectively with limited memory resources is valuable [77]. Sometimes, the measurement cost is negligible, learning is run only once and thus can take quite a long time, but the evaluation speed of the classifier is critical [81].

In this thesis we focus on the problem of learning fast classifiers, especially in the context of computer vision. The importance of the evaluation speed in computer vision applications has been understood for a long time. Various code optimisations, hardware implementations, or architecture changes have been proposed in the literature to speed up existing algorithms. A more systematic interest has emerged in computer vision literature recently [81, 62, 5, 7, 34, 10], in particular with connection to the fast object detection problem.

The problem of fast sequential decision-making have been studied in statistical hypothesis testing theory. In classical hypothesis testing a decision strategy is based on a fixed number of observations (measurements). The necessary number of observations is determined by the required probability of error of the test. However, the required sample size could be very large

which is impractical in many applications. Imagine for instance a situation where one needs to open half of tins in a freight to verify that the shipment is not spoilt.

In the 1940's, Abraham Wald published his theory of sequential hypothesis testing [83]. In his formulation, the number of observations is not determined in advance of the experiment but depends on the observations made during the test. A sequential test takes the observations one by one and when the required decision precision is attained, the test is terminated. On average, fewer number of observations is needed to attain the required error rates. Wald formulated the sequential hypothesis testing problem and found an optimal solution for it – the sequential probability ratio test (SPRT).

Sequential analysis has been further developed and enriched and is now used as a basic and well known tool in statistics [72]. Nevertheless, its applicability to computer vision problems is not always straightforward due to its underlying assumptions. The SPRT is based on measurements that are drawn independently from a known distribution. In computer vision problems the measurements are usually correlated and the distributions are complex and difficult to estimate or parametrise.

1.1 Motivation and Goals of the Thesis

The thesis was originally motivated by the latest progress in the development of fast algorithms for object detection. Several authors showed that building a real-time detector is possible with partially automatic learning algorithms with only small manual intervention to the training procedure. Here we aim at developing an automatic training procedure which avoids those manual tunings. Especially, we are interested in formalisation of the training problem with time-to-decision vs. precision trade-off and optimal or near-optimal solutions to it.

Another goal of the thesis is to emphasise the importance of the classifier evaluation time as an optimisation parameter for training algorithms. It is clearly observable in many areas of computer vision research that an algorithm is valued more if it is not only precise but if it is fast as well. We would like to stress the lack of theory in this field and encourage others to explore this important problem.

1.2 Contributions of the Thesis

This thesis contributes to state of the art of learning a sequential classifier.

The WaldBoost algorithm. As the main contribution, we show in Chapter 5 how Wald's sequential probability ratio test (SPRT) and the AdaBoost learning algorithm can be combined into a single learning algorithm, called WaldBoost, with several favourable properties. First, the learned classifiers are sequential in evaluation which allows fast classification. Second, the WaldBoost overcomes limitations of SPRT to (i) *a priori* selected and ordered measurements, and (ii) class-conditionally independent measurements or known measurement joint probability density functions. Third, the WaldBoost learning is naturally suitable for very large training sets due to its combination with bootstrapping technique. Finally, even though the learned sequential

strategy is built greedily by the AdaBoost algorithm, and hence is not optimal, the experiments on different problems show WaldBoost's ability to find very effective strategies.

Fast face detection. We apply the WaldBoost algorithm to the face detection problem in Chapter 6. Our solution can be viewed as a principled way to build a close-to-optimal "cascade of classifiers" [81]. The face detection problem is used to demonstrate the properties and useful settings of the WaldBoost learning. The experiments show competitiveness of the WaldBoost learning to the state of the art algorithms in detection rates and superior performance in evaluation speed.

Speeding up binary decision processes by emulation. As another contribution, we show in Chapter 7 how existing (slow) binary decision algorithms can be approximated by a (fast) trained WaldBoost classifier. Two interest point detectors, the Hessian-Laplace and the Kadir-Brady saliency detectors, are emulated to demonstrate the approach. Experiments show that while the repeatability and matching scores are similar for the original and emulated algorithms, a 9-fold speed-up for the Hessian-Laplace detector and a 142-fold speed-up for the Kadir-Brady detector is achieved. For the Hessian-Laplace detector, the achieved speed is similar to SURF, a popular and very fast handcrafted modification of Hessian-Laplace; the WaldBoost emulator approximates the output of the Hessian-Laplace detector more precisely.

On-line learning with WaldBoost. Finally, we propose an extension of the WaldBoost algorithm to the on-line learning. Two important problems unsolved so far in the on-line boosting training are treated by the approach: (i) classifier evaluation speed optimisation and, (ii) automatic classifier complexity estimation. We show how the on-line boosting can be combined with Wald's sequential decision theory to solve both of the problems.

1.3 Structure of the Thesis

In Chapter 2 we formalise the problem of learning with time-to-decision parameter for two-class classification. The state of the art methods related to the problem defined are reviewed in Chapter 3. Then, two main building blocks of the proposed WaldBoost algorithm, the AdaBoost algorithm and the sequential probability ratio test (SPRT) are described in Chapter 4. The main result of the thesis, the WaldBoost algorithm, is presented in Chapter 5. Then, two applications of the WaldBoost algorithm to the face detection in Chapter 6 and to algorithm speedup by emulation in Chapter 7 are described and experimental evaluation is discussed. Finally, an extension of the proposed algorithm to the on-line learning is presented in Chapter 8. The thesis is concluded in Chapter 9.

1.4 Authorship

I hereby certify that the results presented in this thesis were achieved during my own research in cooperation with my thesis advisor Jiří Matas, published in [73, 74, 50, 30, 75] and with Helmut Grabner and Horst Bischof, published in [30].

2 Problem Formulation

In this chapter we formulate the learning task for the two-class sequential classification problem. Unlike standard learning algorithms, we consider classifier evaluation speed as a parameter optimised in learning.

In standard two-class sequential classification problem, an ordered set of measurement functions $\Xi = \{\xi_{(1)}, \xi_{(2)}, \ldots, \xi_{(M)}; \xi_{(i)}: \mathcal{X} \to \mathcal{X}_i\}$ on an object $\mathbf{x} \in \mathcal{X}$ is given. The outputs of the measurement functions ξ_i are measurements x_i, i.e. values of some measurable quantities of the object $\mathbf{x}$. The task is to determine object's unknown class $y \in \{-1, +1\}$ based on measurements x_i.

To classify object $\mathbf{x}$, a sequential decision strategy $S: \mathcal{X} \to \{-1, +1\}$ is adopted. The strategy S is evaluated as a sequence of decision functions $S_t: \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_t \to \{-1, +1, \sharp\}$. Besides deciding to one of the two classes -1 and $+1$, the decision functions may also return the '$\sharp$' sign, which stands for "still undecided". The decision functions S_t are evaluated one by one until a decision is reached. Note that in order to evaluate S_t, only measurement functions $\xi_1, \ldots, \xi_t$ have to be evaluated. This is in contrast to fixed sample size tests where all measurements are taken in advance.

The precision of a sequential decision strategy S is characterised by strategy's false negative rate α_S and false positive rate β_S

$$\alpha_S = P(S = -1 | y = +1) , \tag{2.1}$$

$$\beta_S = P(S = +1 | y = -1) . \tag{2.2}$$

The number of measurements needed by a sequential strategy S to reach a decision is a random variable. We will denote this number by N_S. Depending on the class y, the average evaluation time of the strategy S is then defined as

$$\bar{T}_{S,-1} = E[N_S | -1] \quad \text{and} \quad \bar{T}_{S,+1} = E[N_S | +1]. \tag{2.3}$$

It depends on both the adopted strategy and the object class y, since for each class the strategy may need different number of measurements on average to reach the decision.

Definition 1. *A sequential decision strategy S^* is said to be* evaluation-time-optimal *if for a given distribution $p(x_1, x_2, \ldots, y)$ and all sequential decision strategies S*

$$(\alpha_{S^*} \geq \alpha_S) \wedge (\beta_{S^*} \geq \beta_S) \quad \Rightarrow \quad (\bar{T}_{S^*,-1} \leq \bar{T}_{S,-1}) \wedge (\bar{T}_{S^*,+1} \leq \bar{T}_{S,+1})$$

Wald in his work [83] showed that the sequential probability ratio test (SPRT), described in more detail in Chapter 4, is an evaluation-time-optimal sequential decision strategy, i.e. for

given $\alpha_{S^*}, \beta_{S^*}$ it is the fastest strategy on average. However, the decision functions S_t in SPRT are based on the likelihood ratio

$$R_t = \frac{p(x_1, \ldots, x_t | y = -1)}{p(x_1, \ldots, x_t | y = +1)} \,. \tag{2.4}$$

This is easy to evaluate when the measurements are independent since $p(x_1, \ldots, x_t | y = c) = \prod_{i=1}^{t} p(x_i | y = c)$.

Learning problem formulation. Nevertheless, in many practical problems the measurements are dependent. Also the densities are not known and their estimation quickly becomes intractable even for moderate t. Thus, SPRT becomes difficult to evaluate. To overcome this difficulty we formulate the following *learning* problem, where $\hat{\alpha}_S$ and $\hat{\beta}_S$ are empirical estimates of the false negative and the false positive errors of a strategy S computed on a training set and $\hat{T}_{S,y}$ is an empirical estimate of the average evaluation time.

Two-class Sequential Classification Learning Problem

Given:

- a training set $\mathcal{T} = \{(\mathbf{x}_1, y_i), \ldots, (\mathbf{x}_m, y_m); \mathbf{x}_i \in \mathcal{X}, y_i \in \{-1, +1\}\}$
- an *unordered* set (class) of measurement functions

$$\Xi_u = \{\xi_1, \xi_2, \ldots, \xi_M; \xi_i \colon \mathcal{X} \to \mathcal{X}_i\}$$

- two values α, β such that $0 \leq \alpha, \beta \leq 1$

The objective:

Out of all sequential strategies S trained on $\mathcal{T}$ using *ordered* measurement functions from Ξ_u and fulfilling

$$\hat{\alpha}_S \leq \alpha \,, \tag{2.5}$$

$$\hat{\beta}_S \leq \beta \,, \tag{2.6}$$

find a strategy S^L such that

$$\hat{T}_{S^L,y} \leq \hat{T}_{S,y} \qquad \text{for } y \in \{-1, +1\}. \tag{2.7}$$

The learning objective is thus to find the decision functions S_t^L such, that the learned sequential strategy fulfilling the required conditions (2.5) and (2.6) on the strategy error rates and minimises the evaluation time. Note, that this also incorporates a search for ordering of the set of measurement functions Ξ_u.

It is also necessary to study the generalisation properties of the learned strategy in terms of both the error rates and the evaluation time. The proposed learning algorithm mostly relies on

the generalisation properties of SPRT and the AdaBoost algorithm for evaluation speed and error generalisation respectively. Moreover, the learning is organised so that the training set is split into training and validation parts to avoid biased estimates.

3 State of the Art

In this chapter we review previous work on sequential classification and learning of sequential classifiers. We start by giving an overview of methods for sequential decision making which do not comprise any learning and we use this overview to show various solutions to the problem but also different problem formulations used. Then, the learning methods related to the learning problem formulation introduced in Chapter 2 are described and their relation to the proposed solution is discussed.

3.1 Being Sequential

The history of the formulation of a classification task with time-to-decision vs. precision trade-off dates back to 1940's to Wald's sequential analysis [83]. However, the problem of sequential decision making is inherent in many other old problems like the Twenty questions game [1] or medical diagnoses. People have intuitively understood the advantages of the sequential architecture and have been using such architectures implicitly or explicitly with or without theoretical justification or optimality of their solution. The optimisation problem with time-to-decision vs. precision trade-off can be posed variously, so different formulations of the problem have arisen. We will first describe the sequential methods which do not learn the sequential strategy.

Wald's sequential analysis. Instead of using a fixed set of measurements for finding the optimal decision, Wald formulated a *sequential* classification task. The goal is to optimise the number of measurements taken, given an upper bound on the quality of the attained solution, i.e. to find an evaluation-time-optimal decision strategy.

Wald was motivated by the demands for more efficient sampling inspection procedures during World War II. A typical application in his research was the freight inspection: Given a large freight containing one brand of goods, the test needs to be performed to decide whether the quality of the goods in the freight is above some threshold and so it can be accepted, or whether to reject it. The test has to be designed so that the number of tested items in the freight is minimised and the errors of accepting a third-rate freight and rejecting a good-quality freight are small.

To this end, he devised a theory for sequential hypothesis testing – sequential analysis. In particular, he found an evaluation-time-optimal solution to the sequential two-class classification problem – the sequential probability ratio test (SPRT). The SPRT is used in the thesis and it will be described in more detail in Chapter 4.

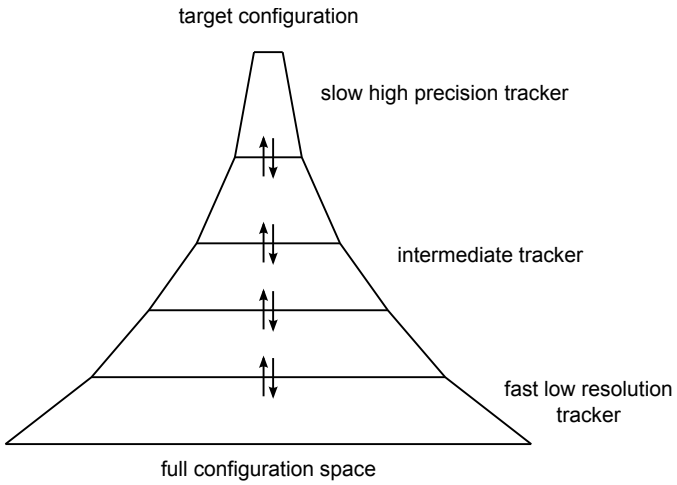


Figure 3.1: Incremental focus of attention (IFA) architecture for tracking [80].

However, SPRT was developed under certain assumptions which do not need to hold generally. The SPRT is based on measurements that are drawn independently from a known distribution with unknown parameter. Although this is often the case in statistical problems, in computer vision applications the measurements are usually correlated and the distributions are complex and difficult to estimate. Due to these preconditions, Wald's results are directly applicable to the computer vision problems rather rarely [49, 58]. Nevertheless, the SPRT has been applied to e.g. classifying examinees in a variable-length computerised classification test (CCT) [79] and is widely used for the quality control. The sequential analysis has been further developed and enriched [72] and is now used as a basic and well known tool in statistics.

Incremental Focus of Attention (IFA) architecture. Increasing processing power had allowed increasingly more tasks to be solved in (or close to) real-time using classical non-sequential methods. However, in tracking the amount of processed data was still unmanageable for both fast and precise localisation. To handle this problem, Toyama [80] proposed an IFA framework (see Figure 3.1), which combines several tracking algorithms in a layered architecture to flexibly trade the precision for robustness and speed. The tracking algorithms are organised into layers by their accuracy and speed – the higher the layer, the more accurate and slower algorithm it contains. During tracking, the layers are applied sequentially starting with the lowest one. Application of each layer incrementally focuses the search space using increasingly more complex algorithms. The IFA architecture leads not only to fast and accurate tracker but was shown to increase the robustness as well. Nevertheless, the framework does not give any advice on how to treat the speed vs. precision trade-off systematically and a particular architecture is highly dependent on constructor's design choices.

Similar architecture has been used by Rowley [65] in his neural network-based rotationally invariant face detector. He trained a two-layered detector with one layer reducing the search space

by determining the in-plane rotation and the other layer containing a frontal face vs. background classifier.

Coarse-to-fine hierarchical visual selection. Geman et al. [5, 16] studied theoretically a related problem of sequential classification and pose estimation. This problem can be split into two: (i) classification to object vs. non-object class, and (ii) estimating the pose of the object.

To solve the classification problem, object class characteristic co-occurrences (arrangements) of features are tested. Increasing the size of the arrangements decreases probability of their appearance for the non-object class. Setting the arrangement tests to have zero false negative rate, the false positive rate decreases with increasing size of the tested arrangement.

Object parameters space is assumed to be hierarchically organised, like in the case of the space of object shifts and rotations. To find the true parameters value, the arrangements are found such that the parameter space is incrementally pruned. The strategy can be represented as a tree with splits leading either to another test on a parameters subspace or directly to the "non-object" decision.

Geman et al. showed empirically that the probability of the non-object class really decreases exponentially. For the parameters estimation part they prove that the best strategy in terms of mean computation is coarse-to-fine, meaning that the strategy should use the most generic tests covering whole space first and proceed in the order of test un-specificity to certain subset of parameters.

One disadvantage of the approach is that all the features have to be evaluated at all positions first, before testing the arrangements. Later, we will see approaches which amortise this computational burden in a better way (e.g. the work of Viola and Jones discussed below). Also the construction of such hierarchical strategies is not easy. In their paper on face detection [16], Fleuret and Geman present a method for constructing such detector, however the approach is rather heuristic, e.g the tests are "forced" to "spread out" over the image, so that they are not all focused on small image area. Also, in their face detection application their false positive rate is higher than that of Rowley [64]. One possible explanation could be that the feature/test selection scheme is too weak.

An application of the hierarchical framework of Geman et al. can be found in [66]. The framework is used to build a hierarchical SVM face detector. Nevertheless, the results are far from real-time even for 384×288 images and the results are reported only for false positive rates which are very high compared to other approaches.

Sequential multi-class classification. In the sequential multi-class classification, the task is to effectively discriminate the right class out of many. In this context, the efficiency of the classification has been studied by Baker and Nayar [2]. Their pattern rejection theory is similar to the Fisher's discriminant analysis (FDA). Given labelled data, projection vectors (called rejectors) are computed to discriminate the classes the best. In contrast to FDA, the projections are designed to allow quick elimination (rejection) of maximum number of classes. The projections are constructed and applied sequentially, i.e. the data projected to one vector are

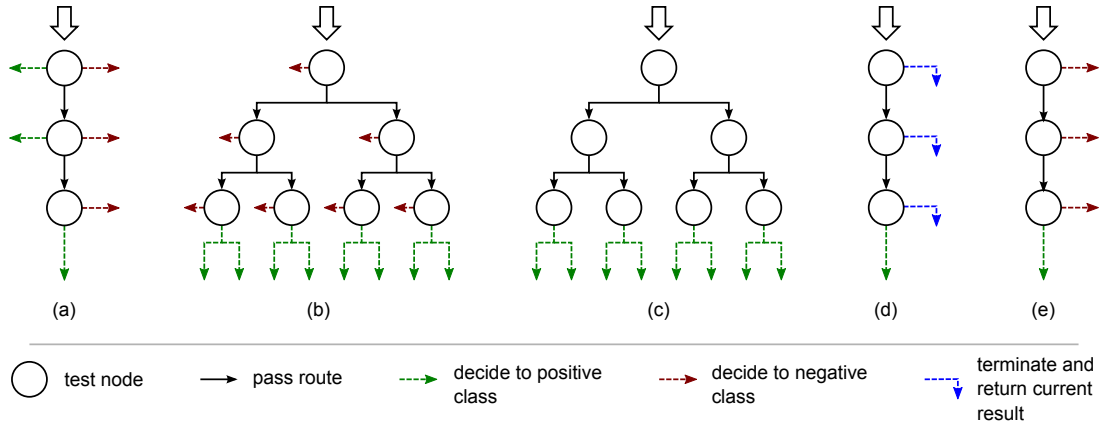


Figure 3.2: Sequential strategies taxonomy. Strategies represented as decision trees: (a) Wald's SPRT, (b) hierarchical visual selection scheme, (c) multi-class rejectors, (d) anytime algorithms, (e) cascaded classifier.

binned and a next-stage rejector is constructed independently for each bin for the classes falling to that bin only.

Anytime algorithms. A different view on sequential classification can be found in so called anytime algorithms [12]. These algorithms are designed to provide *some* solution anytime they are interrupted. Instead of directly minimising the evaluation time, the goal is to provide the most accurate result in a given time. Such algorithms are used in artificial intelligence for agent action planning [6], in planning of medical treatment procedures [14], but a classical example is also the Newton-Raphson iteration applied to finding the square root of a number.

The anytime formulation may seem to be more relevant for the real-time application design (cf. agent planning example). However, there are many problems where a low error decision is needed and the task is to deliver it as fast as possible, while the intermediate solutions are not very useful. For instance in the face detection application in Chapter 6, a solution with high false positive rate would require correspondingly higher effort of subsequent processing (e.g. face recognition or human-computer interaction) so it is not that valued as a slightly slower solution but with only few false positives.

Taxonomy of sequential strategies. A variety of sequential strategies is summarised in Figure 3.2. Each sequential strategy can be represented as a decision tree. Wald's SPRT forms a spruce-like tree with two possible decisions at each node and one "continue" branch leading to another test (Figure 3.2a). In hierarchical visual selection of Geman et al. (Figure 3.2b) the process incrementally refines the estimate of the object pose, while allowing early decisions to the non-object class. The multi-class sequential classification of Baker and Nayar can be represented as a (full or partial) tree with decisions made only at leaf nodes. The test nodes

are constructed to reduce the depth of the tree (Figure 3.2c). Anytime algorithms produce the optimal solution only when all the tests are performed, but they may be interrupted at any tree level (Figure 3.2d). In the next section a simplified version of the spruce-like tree called cascade (Figure 3.2e) is discussed in more detail. In cascaded classifiers the decision to positive class is postponed until the lowest tree level. A simplified IFA architecture (without backward links) can be also represented by this type of decision tree.

The number of examined tree levels is often linked to the computational complexity of the sequential strategy. Often, the cost of the tests in the nodes is not the same. A good strategy is to use cheap and fast tests in the top nodes and increase their power and complexity at lower levels [80, 5, 14, 81]. When all the tests have the same complexity, their are ordered by their discriminability [2, 5, 80].

3.2 Learning to be Sequential

Not surprisingly, the increased interest in learning sequential strategies is correlated with increasing processing power of computers and especially with increased interest in image and video processing. Whereas Wald's motivation to speed up the classical statistical decision methods was due to new demands of World War II, today's development is motivated by real-time performance requirements on applications processing images and large datasets. The classical learning algorithms do not optimise the evaluation time and thus new learning techniques need to be developed. Surprisingly, not many learning methods for sequential classification can be found in literature.

Learnable decision lists. In 1987, Rivest [61] studied learnability of decision lists in the context of Boolean functions. The decision lists are shown to be a generalisation of decision trees and CNF (class of formulae in conjunctive normal form) and DNF (class of formulae in disjunctive normal form). They can be seen as a cascaded classifier (see Figure 3.2e) with early decision either to the positive or to the negative class in each layer. Rivest showed that the decision lists are polynomially learnable, i.e. they can be learned in polynomial time with respect to the training set size and the size of the elementary test set. However, Rivest did not study the efficiency of evaluation of the learned lists.

Cascaded face detection. Probably the most influential work on training sequential classifiers is the cascaded face detector by Viola and Jones [81]. They used the brute force search for face detection illustrated in Figure 6.1 and still were able to achieve real-time performance (i.e. millions of classifications per second). The speed of their classifier can be attributed to (i) sequential design of the cascaded classifier, (ii) efficiently evaluated Haar-like features (measurements), (iii) the measurement selection and combination into stage classifiers by the AdaBoost algorithm, and (iv) the bootstrapping technique used to explore large training set.

A cascade, as defined in [81], is an ordered set of classifiers (stages) of increasing complexity, each rejecting a fraction of negative samples while retaining majority of the positive ones (see Figure 3.3). An image sub-window is passed to the first stage classifier and if it is not rejected it

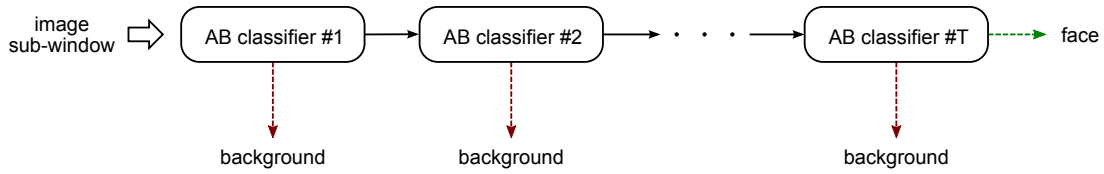


Figure 3.3: The cascaded classifier of Viola and Jones [81] for face detection. The layers (stages) contain increasingly more complex AdaBoost classifiers.

is passed to the second stage, third stage, etc. It is classified as a face if it passes without rejection all the stages. The cascaded structure allows fast decisions by quick rejection of dominating but simple background examples (only stages of low complexity are applied) and by concentrating the computational power to the more difficult but rare ones.

A modification of the bootstrapping technique [77] is used to collect training samples for the stage classifier learning. To train the t-th stage classifier, only samples which have passed through the previous stages $1, \ldots, t-1$ are used. It allows to train on large training sets by repeated pruning and re-sampling of the sample set.

The stage classifiers are trained using the AdaBoost algorithm [18]. The early stages use very short classifiers, the later increasingly more complex ones.

The idea of the classification cascade is not new. Some references have been already cited in the previous section. As pointed out by Schneiderman [70], the principle has been used already in the 1970's for automatic target recognition [4]. Also the approach of Rivest [61] can be interpreted as a cascade learning approach. However, the work of Viola and Jones in a sense culminated the effort by combining the long-studied cascaded approach with an automatic training algorithm and demonstrated its strength on a non-trivial face detection task.

Variants of cascaded classifier. The cascade learning algorithm as proposed by Viola and Jones left some space for improvement. In fact, although they proposed an automatic training procedure for cascade building, the cascade structure of their best reported detector was partially manually tuned. Most of the papers building on the Viola and Jones approach focus on improving the sequential design of the classifier and its training, yet there have been many other papers improving other aspects of the detector (e.g. [45, 24, 56, 23]).

One often mentioned inefficiency of the original cascaded structure is that it starts to train each stage from scratch without taking into account the output of the previous stages. It can be overcome by so called boosting chain architecture – by incrementally training a "monolithic" classifier with bootstrapping, with several early decision thresholds at the positions of original stage splits [86, 85, 32, 7, 10]. This architecture change have been shown to decrease the evaluation time substantially, since much shorter stage was required to improve the output of the previous ones.

Grossmann [32], in order to optimise further the cascade training parameters, proposed to learn a monolithic classifier first and to find the decisions thresholds ex post, creating a boost-

ing chain-type classifier. He did not use the bootstrapping to explore the large training set and the training algorithm requires the a priori probability of the positive class to be known – however, it is not even constant in many applications. His approach is not compared properly with other methods and he shows only small improvement in the evaluation speed on the expense of decreased detection accuracy. Yet, the evaluation speed is considered to be an optimisation parameter of the thresholds search algorithm.

Luo [47] expects the cascade stages to be trained and proposes an algorithm for optimising the stage rejection thresholds. Nevertheless, the evaluation speed is not a part of the optimisation and only the false positive and false negative rates are optimised.

A more complex approach to the cascade optimisation is taken in the Soft Cascade algorithm [7]. First, a relevant set of weak classifiers (see Chapter 4 for detailed explanation of the AdaBoost algorithm) is found by greedily adding weak classifiers as in AdaBoost, incrementally extending the training set by the most difficult samples (bootstrapping) and potentially removing old weak classifiers if it results in a drop in the classification error. The training algorithm then optimises almost all boosting chain parameters. Given the required detection rate, the false positive rate and the time-to-decision are minimised by re-ordering the weak classifiers and finding the rejection thresholds. The algorithm produces state-of-the-art results in the error rates, however the evaluation speed achieved is rather low compared to other methods. It is not clear whether it is caused by more difficult training data or by the algorithm itself.

Another approach to cascade optimisation was proposed by Brubaker et al. [10, 9, 76]. In the original Viola and Jones cascade training algorithm the expected stage error rates and the number of stages are fixed in advance such that the desired overall error rates are reached. Brubaker et al. propose a cascade indifference curve framework for better planning of the stage goals during the training. However, their ROC results are not good enough and even the speedup reached is rather low. So they propose to use recycling (boosting chain) and retracing (thresholds are added even into the stage ensembles) and CART-based weak classifiers. Combined together, a state of the art performance is reached in both the error rates and the evaluation speed.

Learning a sequential support vector machine (SVM) classifier. The SVM learning is known to produce a classifier with good generalisation properties. However, it is slow to evaluate since the number of support vectors needed for a precise decision boundary representation can be very high and each support vector is a multi-dimensional vector for which a dot product needs to be computed with the tested image.

The number of support vectors can be reduced using the Reduced Set Vector Machine (RVM) algorithm [71]. Moreover, RVM provides a hierarchy of classifiers of increasing complexity. Both properties have been used to train a sequential face detector in [62]. They used a favourable property of the RVM hierarchical classifier that the first Reduced Set Vector is the most discriminative one, the second discriminates the samples misclassified by the first one, etc. The RVM classifier was sequentialised by estimating rejection thresholds for all classifier lengths. The approach has been further accelerated by using the integral image representation in [59].

The sequential RVM approach uses conceptually different view of sequential classifier learning from that of Viola and Jones. Instead of combining weak but fast classifiers to increase their

accuracy, an optimal classifier is found by SVM, approximated by RVM and sequentialised to make it fast. The training is faster, globally optimal and fully automatic.

It is difficult to compare these approaches, however the reported speeds and accuracy results speak for the Viola and Jones approach. What is probably an important factor slowing down the RVM classifier is the complexity of the elementary tests – one dot product in a high dimensional space must be computed. Even if this computation is approximated using the integral image representation, the tests are still much more complex than those of Viola and Jones.

In principle, the sequential SVM should reach better classification rates. However, the detection rates in [62, 59] are reported only for very high false positive rates (0.001% per sub-window) which is more than what is required in a real application.

Literature review for Other sequential task formulations. We are interested mainly in binary classification problems without hierarchical parameter space and the evaluation is not terminated suddenly from outside. There has been an intensive research on learning sequential hierarchical classifiers for multi-pose face detection [43, 36, 85, 35]. A good state of the art overview can be found in [34]. An overview on learning anytime algorithms can be found in [14]. SVM based approach for visual selection was proposed in [40].

3.3 Relation of the Thesis to the State of the Art

In Chapter 5 we propose a novel learning algorithm called WaldBoost for solving the two-class sequential classification learning problem defined in Chapter 2. In contrast to RVM, the WaldBoost algorithm retains the greedy learning of the Viola and Jones cascaded classifier learning. It uses the results of Wald and builds a boosting chain-type classifier. Compared to the Viola and Jones approach and its variants, the WaldBoost algorithm can be seen as a theoretically justified cascaded classifier. The evaluation time is an optimisation parameter of the learning process. In contrast to learning algorithms of Brubaker [10] and Bourdev [7] we do not post-optimise the trained classifier and still show competitive results.

4 Preliminaries

The proposed WaldBoost algorithm which will be described in detail in Chapter 5 builds on properties of two algorithms. It uses the AdaBoost algorithm to select and order the measurements and Wald's sequential probability ratio test (SPRT) to find the decision thresholds. In this chapter, these two algorithms are explained in detail – Section 4.1 gives an overview of the AdaBoost algorithm and Wald's theoretical results are described in Section 4.2.

4.1 AdaBoost

The name "AdaBoost" comes from "adaptive boosting". Boosting techniques allow to combine so called "weak classifiers" into a "strong classifier" and thus *boost* their performance. One of the most popular boosting techniques, the AdaBoost algorithm, was proposed by Freund and Schapire [21]. Its predecessors [67, 19, 8] needed an upper bound on the weak classifiers errors over training set weightings to be known to be able to combine them into a strong classifier. The AdaBoost avoids this weakness (the upper bound is often not know in practice) by *adapting* to actual errors of weak classifiers measured on the training set.

AdaBoost selects and combines weak classifiers[1] $h^{(t)}: \mathcal{X} \to \mathbb{R}$ into a single ensemble by combining their responses into a sum

$$f_T(\mathbf{x}) = \sum_{t=1}^{T} h^{(t)}(\mathbf{x}) \ . \tag{4.1}$$

For a binary-classification problem the final class is given by

$$H_T(\mathbf{x}) = \mathrm{sign}(f_T(\mathbf{x})) \ . \tag{4.2}$$

The weak classifiers could be very simple decision functions (e.g. if an email contains the word "Viagra" classify it as a spam) or could themselves be already quite strong classifiers (e.g. a SVM classifier). The only requirement for AdaBoost to converge to a reasonable classifier (see the explanation below) is that the selected weak classifiers are better than a chance on a weighted training set, i.e. they perform better than random guessing.

There are many variants of AdaBoost for various weak classifier domains – discrete, real-valued, multi-class, ranking scores, ... In the thesis the real-valued version is used [68]. Thus the strong classifier response function f_T is a sum of real-valued responses of weak classifiers $h^{(t)}$. For other variants see e.g. [68, 22].

[1] Versions of AdaBoost for other weak classifiers do exist. Here we limit the explanation to the real-valued ones.

Algorithm 1 Real AdaBoost Learning

Given: $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}; \mathbf{x}_i \in \mathcal{X}, y_i \in \{-1, +1\}$,
set/class $\mathcal{H} = \{h_j; h_j : \mathcal{X} \to \mathbb{R}\}$ of weak classifiers

Initialise weights $w_1(i) = 1/m$ for $i = 1, \ldots, m$
For $t = 1, ..., T$:

1. Choose a hypothesis $h^{(t)} \in \mathcal{H}$ and its parameters minimising training error upper bound
2. If the hypothesis error $\epsilon_t = \sum_{i=1}^{m} w_t(i) [\![y_i \neq \text{sign}(h^{(t)}(\mathbf{x}_i))]\!] \geq 1/2$ then stop
3. Update weights
$$w_{t+1}(i) = \frac{w_t(i) \exp(-y_i h^{(t)}(\mathbf{x}_i))}{Z_t}$$
where Z_t is a normalisation factor chosen so that w_{t+1} sums to 1 over $i = 1, \ldots, m$

Output the final classifier:
$$f_T(\mathbf{x}) = \sum_{t=1}^{T} h^{(t)}(\mathbf{x})$$
where the class membership is given by $H_t = \text{sign}(f_T)$.

4.1.1 AdaBoost Learning

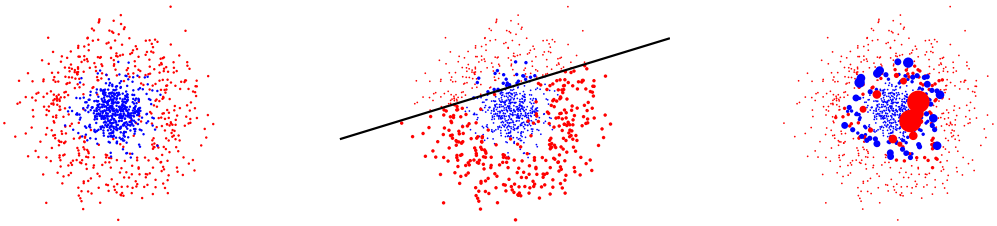
where $I[p]$ returns 1 when the predicate p is true and 0 otherwise

AdaBoost is a greedy learning algorithm [68, 18]. Its structure is shown in Algorithm 1. An input to the algorithm is a labelled training set $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ with samples belonging to either positive, $y_i = +1$, or negative class, $y_i = -1$. Another input is a set (or class) $\mathcal{H}$ of classifiers from which AdaBoost selects the weak classifiers.

The algorithm maintains a vector of training sample weights $\mathbf{w}_t$. The weights encode difficulty of the samples for the greedily constructed strong classifier. At the beginning, all the samples are considered to be equally difficult, so the weights are initialised by a uniform distribution.

The AdaBoost learning runs in a loop. In each iteration, one weak classifier is added to the final ensemble and the weights are updated. In the weak classifier error computation, $[\![p]\!]$ returns 1 when the predicate p is true and 0 otherwise. The learning is stopped when the error of the best weak classifier from the set $\mathcal{H}$ is equal or worse than random guessing. In this case, the upper bound on the training error of the final classifier could not be improved anymore. Another common stopping criterion is the number of weak classifiers combined, T, or the cross-validated performance on a validation set.

The output of the learning loop is a classifier response function f_T which linearly combines the selected weak classifiers $h^{(t)}$ and which minimises the training error on the training set $\mathcal{T}$.



$t = 0$ $t = 1$ $t = 39$

Figure 4.1: Training sample weights evolution. The weights are initialised uniformly over the training set (blue - positive, red - negative class). In the first step a weak classifier (linear perceptron) is found and the weights of incorrectly (correctly) classified samples are increased (decreased). After 39 weak classifiers the weights of the most difficult samples are the highest.

For binary-classification problems the class membership is given by the $\mathrm{sign}(\cdot)$ function on the strong classifier response function f_T.

Next, the re-weighting scheme, the weak classifier selection and properties of the algorithm are discussed in detail.

4.1.2 Re-weighting

The AdaBoost re-weighting scheme adapts the weights depending on the classification output of the selected weak classifier. For each training sample $\mathbf{x}_i$ its old weight $w_t(i)$ is multiplied by an exponential factor to get an updated weight $w_{t+1}(i)$. Note that

$$\exp(-y_i h^{(t)}(\mathbf{x}_i)) \begin{cases} < 1, & y_i = \mathrm{sign}(h^{(t)}(\mathbf{x}_i)) \\ > 1, & y_i \neq \mathrm{sign}(h^{(t)}(\mathbf{x}_i)) \end{cases} \tag{4.3}$$

Thus the weights of correctly classified samples are decreased while the weights of incorrectly classified ones are increased. This way, the learning focuses on more difficult samples. An example of this effect is shown in Figure 4.1 where a simple two-class problem is treated by combining outputs of linear perceptrons (class $\mathcal{H}$).

Another effect of the re-weighting scheme is a "classification independence" of consecutively selected weak classifiers (see Figure 4.2). For a weak classifier $h^{(t)}$ selected in t-th step, the re-weighting ensures [68] that

$$\epsilon_t^{t+1} = \sum_{i=1}^{m} w_{t+1}(i) [\![y_i \neq \mathrm{sign}(h^{(t)}(\mathbf{x}_i))]\!] = \frac{1}{2} , \tag{4.4}$$

where ϵ_t^{t+1} is a weighted error of $h^{(t)}$ measured on the weights $\mathbf{w}_{t+1}$.

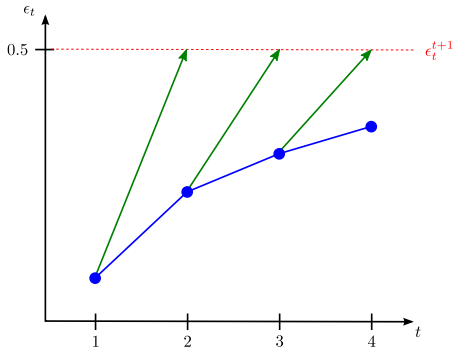


Figure 4.2: Classification independence of consecutively selected weak classifiers. Blue dots show the weighted errors of selected weak classifiers in step t and the green arrows indicate the error of the same weak classifier on the updated weights $\mathbf{w}_{t+1}$.

The weak classifier $h^{(t)}$ is therefore equivalent to a random guess on the weights $\mathbf{w}_{t+1}$ and so $h^{(t+1)}$ has to classify correctly different training samples in order to be selected [68]. This property is useful when thinking of AdaBoost as a feature or measurement selector as we will do in Chapter 5.

4.1.3 Training Error Upper Bound

To find a weak classifier in each step and to find its parameters, the AdaBoost uses the following theorem.

Theorem 1 (Schapire and Singer [68])**.** *Assuming the re-weighting scheme from Algorithm 1, the following bound holds on the training error of* H_T

$$\frac{1}{m}\left|\{i : H_T(\mathbf{x}_i) \neq y_i\}\right| \leq \prod_{t=1}^{T} Z_t \tag{4.5}$$

Thus instead of minimising the training error directly, the upper bound is minimised greedily. In each step t, the weak classifier and its parameters are chosen such that

$$Z_t = \sum_{i=1}^{m} w_t(i) \exp(-y_i h^{(t)}(\mathbf{x}_i)) \tag{4.6}$$

is minimised. Since $Z_t < 1$ when $\epsilon_t < 0.5$, the upper bound is decreased in each step.

4.1.4 Domain-Partitioning Weak Classifiers

The weak classifiers class $\mathcal{H}$ used throughout the thesis return their confidence based on a partitioning of the feature response domain [68]. Each weak classifier $h(\mathbf{x}) \in \mathcal{H}$ is linked to one feature $q(\mathbf{x})\colon \mathcal{X} \to \mathbb{R}$ and operates on its response. Each weak classifier partitions the feature

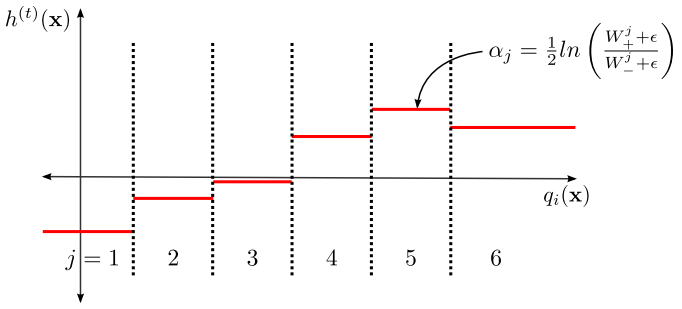


Figure 4.3: The domain-partitioning weak classifier. The response of feature $q(\mathbf{x})$ on object $\mathbf{x}$ is partitioned into bins $j = 1, \ldots, K$. The leftmost and the rightmost bins cover the respective half-spaces. In each bin j, the response of the weak classifier $h(\mathbf{x})$ is computed from the sum of positive (W_+^j) and negative (W_-^j) weights of training samples falling into the bin. To avoid numerical problems, a smoothing factor ϵ is used [68].

response domain into disjoint blocks $X_1, \ldots, X_K$ covering the whole domain, and outputs one real number for each block. So, it returns one of K numbers α_j depending on which block a tested sample falls in. Examples of such weak classifiers are decision trees whose leaves define the partitioning or simple interval-based partitioning classifiers (see Figure 4.3) used in the thesis where the partitioning corresponds to uniform-width interval bins.

The values of α_j are found by minimising Z_t in equation (4.6). Let us define

$$W_b^j = \sum_{i:\mathbf{x}_i \in X_j \wedge y_i = b} w_t(i) \qquad j = 1, \ldots, K; \quad b \in \{+1, -1\} \,, \tag{4.7}$$

a sum of the weights of samples from class b falling into j-th bin X_j. Then equation (4.6) can be rewritten to

$$Z_t = \sum_j \sum_{i:\mathbf{x}_i \in X_j} w_t(i) e^{-y_i \alpha_j} \tag{4.8}$$

$$= \sum_j \left(W_+^j e^{-\alpha_j} + W_-^j e^{\alpha_j} \right) \tag{4.9}$$

and using simple calculus this is minimised when

$$\alpha_j = \frac{1}{2} \ln \left(\frac{W_+^j}{W_-^j} \right) \,. \tag{4.10}$$

The stronger the response the more different are the sums W_+^j and W_-^j, and its sign corresponds to the class with dominating weights in the bin. When plugged back to equation (4.9) we get

$$Z_t = 2 \sum_j \sqrt{W_+^j W_-^j} \,. \tag{4.11}$$

Clearly this is smaller the more the multiplicands differ and is minimised when the sum of the weights of "misclassified" class in each bin is minimal. Thus, the criterion used by the AdaBoost algorithm – minimisation of ϵ_t – minimises also the training error upper bound.

In practice, very small or even zero W_+^j and W_-^j may lead to numerical instability. To avoid these numerical problems, Schapire and Singer [68] proposed to use a smoothing coefficient ϵ in bin's response computation

$$\alpha_j = \frac{1}{2} \ln \left(\frac{W_+^j + \epsilon}{W_-^j + \epsilon} \right) . \tag{4.12}$$

They show that it only slightly weakens the upper bound when $\epsilon \ll 1/(2K)$ and they recommend to set $\epsilon = 1/m$. In the following, we are using this smoothed version of domain-partitioning weak classifiers.

4.1.5 Training Convergence

An important property for the WaldBoost algorithm proposed in Chapter 5 is the asymptotic convergence of the AdaBoost algorithm. As shown in [22], choosing a weak classifier minimising the upper bound (4.5) in each cycle of the AdaBoost learning converges asymptotically to

$$\lim_{T \to \infty} f_T(\mathbf{x}) = \frac{1}{2} \log \frac{P(y = +1 | \mathbf{x})}{P(y = -1 | \mathbf{x})} . \tag{4.13}$$

Note that the strong classifier's response is asymptotically proportional to the likelihood ratio. We will use this convergence property later when the AdaBoost is combined with the SPRT in the WaldBoost algorithm in Chapter 5.

4.2 Sequential Analysis

The sequential decision-making theory was developed by Wald [83] as a statistical tool for sequential hypothesis testing. Motivated by the quality control problem where one does not want to check all items in a freight to say if it fulfils a required quality criteria but wants to decide as soon as possible, he formulated a two-class sequential classification task. He proved that evaluation-time-optimal (see Definition 2) solution to this classification problem is the *sequential probability ratio test*.

4.2.1 Sequential Probability Ratio Test (SPRT)

Let $\mathbf{x} \in \mathcal{X}$ be an object characterised by its class (hidden state) $y \in \{-1, +1\}$. The class is not observable and has to be determined based on successive measurements $x_1, x_2, \ldots; x_i \in \mathcal{X}$. Let the joint conditional density $p(x_1, \ldots, x_t | y = c)$ of the sequence of measurements $x_1, ..., x_t$ be known for $c \in \{-1, +1\}$ and for all t.

SPRT is a sequential strategy S^* (see Chapter 2 for the definition), which is defined as:

$$S_t^* = \begin{cases} +1, & R_t \leq B \\ -1, & R_t \geq A \\ \sharp, & B < R_t < A \end{cases} \tag{4.14}$$

where R_t is the likelihood ratio

$$R_t = \frac{p(x_1, ..., x_t | y = -1)}{p(x_1, ..., x_t | y = +1)} \ . \tag{4.15}$$

The constants A and B are set according to the required error of the first kind α and error of the second kind β. Optimal A and B are difficult to compute in practice, but tight bounds are easily derived.

Theorem 2 (Wald)**.** *The value of A is upper bounded by $(1-\beta)/\alpha$ and the value of B is lower bounded by $\beta/(1-\alpha)$.*

Proof. For each sequence of measurements $(x_1, \ldots, x_t)$, for which SPRT returns the class -1 we get from (4.14) and (4.15)

$$p(x_1, \ldots, x_t | y = -1) \geq A \cdot p(x_1, \ldots, x_t | y = +1) \ . \tag{4.16}$$

Since this holds for all sequences of measurements classified to class -1 ($S^* = -1$), summing over these sequences

$$P\{S^* = -1 | y = -1\} \geq A \cdot P\{S^* = -1 | y = +1\} \ . \tag{4.17}$$

The term on the left side is the probability of correct classification of an object from the class -1 and is therefore $1 - \beta$. The term on the right side is the probability of incorrect classification of an object from the class $+1$, and is equal to α. After this substitution and rearranging, we get the upper bound on A. Repeating this derivation with samples classified by SPRT to $+1$, the lower bound on B is derived. □

In practical applications, Wald suggests to set the thresholds A and B to their upper and lower bound respectively

$$A' = \frac{1-\beta}{\alpha}, \qquad B' = \frac{\beta}{1-\alpha} \ . \tag{4.18}$$

The effect of this approximation on the test error rates was summarised by Wald in the following theorem.

Theorem 3 (Wald)**.** *When A' and B' defined in (4.18) are used instead of the optimal A and B, the real error probabilities of the test change to α' and β' for which*

$$\alpha' + \beta' \leq \alpha + \beta \ . \tag{4.19}$$

Proof. From Theorem 2 it follows that

$$\frac{\alpha'}{1-\beta'} \leq \frac{1}{A'} = \frac{\alpha}{1-\beta} \ , \qquad \text{and} \tag{4.20}$$

$$\frac{\beta'}{1-\alpha'} \leq \frac{1}{B'} = \frac{\beta}{1-\alpha} \ . \tag{4.21}$$

Multiplying the first inequality by $(1-\beta')(1-\beta)$ and the second by $(1-\alpha')(1-\alpha)$ and summing both inequalities, the result follows. □

This result shows that at most one of the probabilities α and β can be increased and the other has to be decreased by the approximation.

Theorem 4 (Wald)**.** *SPRT (with optimal A and B) is evaluation-time-optimal sequential strategy.*

Proof. The proof is complex. We refer the interested reader to [83]. □

Wald analysed SPRT behaviour when the upper bound A' and B' is used instead of the optimal A and B. He showed that the effect on the speed of evaluation is negligible.

5 WaldBoost

In this chapter we describe the main result of the thesis – the WaldBoost algorithm. It is a learning algorithm for addressing the two-class sequential classification learning problem formulated in Chapter 2. In contrast to the SPRT proposed by Wald, the WaldBoost sequential strategy is not based on randomly selected i.i.d. measurements but the measurements are selected during the learning phase together with the decision thresholds. This scenario reflects the fact that in many computer vision problems the measurements taken cannot be assumed i.i.d. and thus the densities required by the SPRT are difficult (or even impossible) to compute. The WaldBoost overcomes this limitation of the SPRT by combining it with the AdaBoost algorithm.

5.1 Difficulties with SPRT

Wald in his work [83] considers mainly classical statistical hypothesis testing problems where the measurements are i.i.d., i.e. sampled randomly from some distribution. In this case it is easy to compute the likelihood ratio R_t in equation (4.15) since the required densities can be computed incrementally from one-dimensional probability density functions as

$$p(x_1, \ldots, x_t | y = c) = \prod_{q=1}^{t} p(x_q | y = c) \ . \tag{5.1}$$

Wald only mentions the situation of dependent measurement by noting that the theory is still valid only the densities have to be known.

In computer vision problems, the measurements are often dependent (e.g. pixel intensities in the image). By taking more and more dependent measurements a direct multi-dimensional density estimation becomes quickly intractable. In the following we show how to estimate the likelihood ratio R_t in this situation.

In the non i.i.d. case also the ordering of the measurements matters. This is again a property not studied originally by Wald. Thus, to compute the likelihood ratio R_t one has to select and order the measurements and to estimate the class probability densities.

5.2 SPRT for non i.i.d. Samples

As noted, the SPRT can still be used for dependent measurements if the likelihood ratio R_t from equation (4.15) can be estimated. To this end, we propose to use the AdaBoost algorithm for measurement selection and ordering and for the conditional density estimation. Next, we

propose an approximation for the likelihood ratio estimation for statistically dependent measurements. The proposed algorithm, WaldBoost, which combines SPRT and AdaBoost is described in Section 5.3. Some other related topics and implementation details are discussed in Section 5.4.

5.2.1 Likelihood Ratio Estimation with AdaBoost

The likelihood ratio (4.15) computed on the outputs of weak classifiers found by AdaBoost has the form

$$R_t(\mathbf{x}) = \frac{p(h^{(1)}(\mathbf{x}), ..., h^{(t)}(\mathbf{x}) \,|\, y = -1)}{p(h^{(1)}(\mathbf{x}), ..., h^{(t)}(\mathbf{x}) \,|\, y = +1)} \,, \tag{5.2}$$

where $h^{(i)}(\mathbf{x})$ was substituted for x_i. Nevertheless, the outputs of the weak classifiers cannot be treated as statistically independent. To avoid the computation of $R_t(\mathbf{x})$ involving a high-dimensional density estimation we propose to approximate the tests (4.14) by similar tests on the value of the strong classifier response $f_t(\mathbf{x})$ which are much simpler to evaluate.

To derive the proposed approximation we use the asymptotic property (4.13) which can be rewritten using the Bayes formula to the form

$$\lim_{t\to\infty} f_t(\mathbf{x}) = -\frac{1}{2}\log R(\mathbf{x}) + \frac{1}{2}\log\frac{P(+1)}{P(-1)} \,. \tag{5.3}$$

Thus, in the asymptotic case the strong classifier response is a monotonic function of the likelihood ratio. Thresholding the likelihood ration $R(\mathbf{x})$ on the value A is therefore equivalent to thresholding the strong classifier response on some value $\theta_A^{(t)}$. Similarly, for the threshold B on $R(\mathbf{x})$ there exists a threshold $\theta_B^{(t)}$ on $f_t(\mathbf{x})$. Knowing the thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$, the sequential decision strategy would become

$$S_t = \begin{cases} +1, & f_t(\mathbf{x}) \geq \theta_B^{(t)} \\ -1, & f_t(\mathbf{x}) \leq \theta_A^{(t)} \\ \sharp, & \theta_A^{(t)} < f_t(\mathbf{x}) < \theta_B^{(t)} \,, \end{cases} \tag{5.4}$$

which is easy to evaluate. Note that the inequalities are inverted since $f_t(\mathbf{x})$ is proportional to $-R_t(\mathbf{x})$.

The thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$ could be computed directly by expressing $R(\mathbf{x})$ from equation (5.3). However, the a priori probabilities would need to be known. Also, the thresholds would only hold for the asymptotic case. Instead we do estimate the thresholds in each training step.

To derive a procedure for finding the decision thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$ similar argumentation to that of proof of Theorem 2 will be used. The full derivation is shown only for $\theta_A^{(t)}$, the derivation for $\theta_B^{(t)}$ is analogous.

Let us denote

$$D_t^- = \{\mathbf{x}\colon R_t(\mathbf{x}) \geq A\} \tag{5.5}$$

Algorithm 2 WaldBoost Classification

Given: $h^{(t)}, \theta_A^{(t)}, \theta_B^{(t)}, \gamma \quad (t = 1, \ldots, T)$
Input: a classified object $\mathbf{x}$.
For $t = 1, \ldots, T$ *(SPRT execution)*
  If $f_t(\mathbf{x}) \geq \theta_B^{(t)}$, classify $\mathbf{x}$ to the class +1 and terminate
  If $f_t(\mathbf{x}) \leq \theta_A^{(t)}$, classify $\mathbf{x}$ to the class −1 and terminate
end
If $f_T(\mathbf{x}) > \gamma$, classify $\mathbf{x}$ as +1. Classify $\mathbf{x}$ as −1 otherwise.

the set of samples decided as negative when t weak classifiers have been measured. Since the condition $R_t(\mathbf{x}) \geq A$ holds for all samples from D_t^-, also the following condition holds

$$\sum_{\mathbf{x} \in D_t^-} p(h^{(1)}, \ldots, h^{(t)} | y = -1) \geq A \sum_{\mathbf{x} \in D_t^-} p(h^{(1)}, \ldots, h^{(t)} | y = +1) \ . \tag{5.6}$$

The term on the left side is the probability of correct classification of an object from the class −1 using t weak classifiers. Let us denote this probability as $1 - \beta_t$. The term on the right side is the probability of incorrect classification of an object from the class +1 using t weak classifiers and will be denoted by α_t.

Using the equivalent definition of the set D_t^- with the strong classifier response $f_t(\mathbf{x})$ instead of the likelihood ratio $R_t(\mathbf{x})$

$$D_t^- = \{\mathbf{x}: f_t(\mathbf{x}) \leq \theta_A^{(t)}\} \tag{5.7}$$

a search for $\theta_A^{(t)}$ can be performed on $f_t(\mathbf{x})$ such that condition (5.6) is fulfilled. This search is much simpler than the search with the definition (5.5) of D_t^-, since it is over a one-dimensional space of the strong classifier response only. For each threshold value both β_t and α_t are estimated and condition (5.6) is tested.

Among the thresholds fulfilling (5.6), the highest value of $\theta_A^{(t)}$ and the lowest value $\theta_B^{(t)}$ are taken to allow maximum number of samples to be rejected and accepted and thus to maximise the decision speed of the classifier.

5.3 WaldBoost

The classification and the training parts of the proposed WaldBoost algorithm are described in this section. Moreover, we show how specific properties of some decision tasks lead to a simplification of the algorithm.

5.3.1 Classification

The WaldBoost classification procedure is summarised in Algorithm 2. The input to the algorithm is a learned set of weak classifiers (measurements for SPRT) and the decision thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$. The classification executes the SPRT test (5.4) via a trained strong classifier f_T with a sequence of thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$. If f_t exceeds the respective threshold, a decision

Algorithm 3 WaldBoost Learning with Bootstrapping

Input:

- sample pool $\mathcal{P} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$; $\mathbf{x}_i \in \mathcal{X}, y_i \in \{-1, 1\}$,
- set of features $\mathcal{F} = \{q_s\}$,
- desired final false negative rate α and false positive rate β,
- the number of iterations T.

Set $A = (1 - \beta)/\alpha$ and $B = \beta/(1 - \alpha)$
Sample randomly the initial training set $\mathcal{T} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$ from the pool $\mathcal{P}$
Initialise weights $w_1(\mathbf{x}_i, y_i) = 1/m$
for $t = 1, ..., T$

1. Find $h^{(t)}$ by AdaBoost using $\mathcal{F}$ and $\mathcal{T}$ and add it to the strong classifier $f_t = f_{t-1} + h^{(t)}$
2. Find decision thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$ for f_t on $\mathcal{P}$ using $\hat{R}_t$ (equation (5.8))
3. Throw away samples $\mathbf{x}$ from $\mathcal{P}$ for which $f_t(\mathbf{x}) \geq \theta_B^{(t)}$ or $f_t(\mathbf{x}) \leq \theta_A^{(t)}$
4. Sample new training set $\mathcal{T}$ from the updated pool $\mathcal{P}$

end

Output: Ordered set of weak classifiers $\{h^{(t)}\}_{t=1}^{T}$ and the decision thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$

is made. Otherwise, the next weak classifier is computed. If a decision is not made within T cycles, the input is classified by thresholding f_T on a value γ specified by the user.

For practical reasons, only limited number of weak classifiers is used, which implies truncation of the sequential test. Wald [83] studies the effect of truncation of the sequential test procedure, however, his derivations hold only for cases where i.i.d. measurements are taken. For that case, he shows how the effect of truncation on the false negative and false positive rates of the test declines with number of measurements taken. In our implementation, the final threshold is left unspecified. It is used to control the false positive and the false negative rate in the application. It is also used in a ROC curve generation in the experiments.

5.3.2 Learning with Bootstrapping

WaldBoost learning is summarised in Algorithm 3. The input of the learning algorithm is a pool of positive and negative samples $\mathcal{P}$, a set of features $\mathcal{F}$ - the building blocks of the classifier, the bounds on the final false negative rate, α, and the false positive rate, β, and the number of training steps T. The output is an ordered set of weak classifiers $h^{(t)}$, $t \in \{1, \ldots, T\}$ (i.e. measurements) and a set of SPRT thresholds $\theta_A^{(t)}$, $\theta_B^{(t)}$ defining the decision function S_t in the two-class sequential classification problem defined in Chapter 2 for all lengths $t = 1, \ldots, T$.

The user-defined upper bounds on error rates determine the two thresholds A and B (equation (4.18)) used by the SPRT-based sequential strategy (equation (4.14)). The upper bounds α and β could be related to the final false positive and detection rates in the Viola-Jones cascade building [81]. Unlike Viola-Jones, no stage false positive and detection rates are required as input, only the final ones are needed.

The algorithm starts by sampling a small random training set $\mathcal{T}$ out of much larger sample pool $\mathcal{P}$, i.e. $m \ll N$. It is used for the weak classifier selection, which is by far the most time consuming operation in the learning process, to keep the speed and memory requirements of the training process acceptable. On the other hand, the training set size m has to be chosen so that it represents the problem at hand sufficiently.

The training then runs in a loop. The training set $\mathcal{T}$ is used for training the best weak classifier (Step 1) using standard AdaBoost approach as described in Chapter 4.1. As in the AdaBoost algorithm, a vector of sample weights $\mathbf{w}_t$ is initialised uniformly and the samples are re-weighted at each iteration.

Then, the SPRT thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$ are found (Step 2), as described in Section 5.2.1. The strong classifier response function f_t is needed for each sample to estimate the likelihood ratio. This response has to be computed anyway later in Step 4 for the whole pool, so it is also used for the threshold estimation without affecting the efficiency of the learning. Using the larger pool robustifies the estimation, since most of the decisions are made at the tails of the distributions where only few samples are available. The thresholds are applied directly to the strong classifier response function f_t (not to the likelihood ratio as in SPRT) and are set to $\pm\infty$ if no appropriate threshold was found in the training iteration.

The training set is updated in steps 3 and 4 using the bootstrapping technique [77]. Based on the extended strong classifier and the found decision thresholds, the sample pool is pruned (Step 3). Only samples not decided yet are kept in the pool. Then, a new training set is sampled from the updated pool (Step 4). Steps 3 and 4 are similar to the bootstrapping in the cascade building procedure [81] where the training set is updated after each stage classifier (consisting of several weak classifiers) is trained.

As the training process prunes repeatedly the sample pool $\mathcal{P}$ using bootstrapping, little care has to be taken when estimating the likelihood ratio for $\theta_A^{(t)}$ and $\theta_B^{(t)}$ estimation. As the pool changes, direct density estimation gives $p(f_t(\mathbf{x}) \,|\, y = C, \rightarrow t)$ instead of desired $p(f_t(\mathbf{x}) \,|\, y = C)$, where $C \in \{-1, +1\}$ and the term "$\rightarrow t$" stands for the condition that the sample has not been decided up to training step t.

Using Bayes formula on $p(f_t(\mathbf{x}) \,|\, y = C, \rightarrow t)$ we get

$$\hat{R}_t(\mathbf{x}) = \frac{p(f_t(\mathbf{x}) \,|\, y = -1, \rightarrow t)p(\rightarrow t \,|\, +1)}{p(f_t(\mathbf{x}) \,|\, y = +1, \rightarrow t)p(\rightarrow t \,|\, -1)} \tag{5.8}$$

which gives the correction factor $\frac{p(\rightarrow t|+1)}{p(\rightarrow t|-1)}$ for the likelihood ratio estimation on a bootstrapped validation set. The rest of the threshold search is the same as described in Section 5.2.1.

5.4 Implementation Details

Now, several implementation issues will be discussed.

5.4.1 Importance Sampling

The bootstrapping (steps 3 and 4 in Algorithm 3) was originally proposed by Sung and Poggio [77] to handle large training datasets during learning. A classifier is first trained on a training set of a manageable size. It is then extended by collecting new samples misclassified by the current classifier, and the classifier is re-trained. Later, Viola and Jones [81] used re-sampling of the training set instead of extending, keeping the training set size the same during the learning.

The sampling strategy was shown to influence significantly both the training process and the final classifier [39]. The authors of [39] propose a new sampling strategy, quasi-random weighted sampling + trimming (QWS+), using the sample weights given by the AdaBoost re-weighting scheme. They show that QWS+ is superior to the commonly used uniform sampling (each sample has the same weight) for training an AdaBoost classifier.

In Section 6.2 the uniform and QWS+ samplings are compared in the WaldBoost framework and it is shown, that QWS+ improves significantly the efficiency of the WaldBoost learning, confirming the findings of [39].

5.4.2 Balancing the positive and negative weights

To avoid numerical problems with too small weights of positive samples we keep constant the ratio of positive and negative weights in the training set. This is equivalent to adding a constant-output weak classifier to the ensemble at each training iteration, so only the strong classifier response is biased by a constant. Since the decision thresholds are found irrespective of the response bias, the performance of the algorithm is not affected.

5.4.3 Non-symmetric Wald Decisions

The SPRT sequential strategy (4.14) and its corresponding WaldBoost strategy (5.4) are symmetric, i.e. early decisions are allowed to both the positive and the negative class. However, for many learning tasks a non-symmetric version of the strategy fits better.

In the face detection task discussed in Chapter 6, the positive class (faces) is difficult to collect. The number of collected positive samples is usually just enough for a representative training set. When being further pruned by early decisions, the learning would start to overfit to few remaining difficult samples. On the other hand, the negative samples are easy to collect and the negative class (background) is very large and complex and difficult to represent by a small training set. To use this complex class for learning efficiently, it is better to processed iteratively by deciding easy samples quickly and concentrating on the difficult ones.

In our interest point detection application of WaldBoost in Chapter 7, an arbitrary number of both positive and negative samples is available for bootstrapping. However, if positive samples were bootstrapped, i.e. early positive classification was allowed in equation (5.4), all early

positive decisions would have a confidence close to $\theta_B^{(t)}$ and precise localisation via the non-maximum suppression algorithm would not be possible.

In both cases, the WaldBoost training task can be specified in the following way. Let the required false positive rate β be set to zero and the required false negative rate α to some small constant. In this setting, equations (4.18) reduce to

$$A = \frac{1-0}{\alpha} = \frac{1}{\alpha}, \qquad B = \frac{0}{1-\alpha} = 0 \tag{5.9}$$

and the SPRT strategy (4.14) becomes

$$S_t^* = \begin{cases} +1, & R_t \leq 0 \\ -1, & R_t \geq 1/\alpha \\ \sharp, & 0 < R_t < 1/\alpha \ . \end{cases} \tag{5.10}$$

Since R_t is always positive, the algorithm will never make an early decision to the positive class. The only allowed decision is to the negative class.

A learning process with $\beta = 0$ will not bootstrap the positive samples in the sample pool but will bootstrap the negative ones. Such initialisation thus leads to the exploration of the negative class using bootstrapping while working with a small and mostly unchanging positive sample pool. The detection rate of the final classifier is assured to be $1 - \alpha$ while the false positive rate is progressively reduced by each training cycle.

The WaldBoost decision strategy (5.4) then becomes

$$S_t = \begin{cases} -1, & f_t(\mathbf{x}) \leq \theta_A^{(t)} \\ \sharp, & \theta_A^{(t)} < f_t(\mathbf{x}) \ . \end{cases} \tag{5.11}$$

When evaluated, the classifier thus makes early decisions only to the negative class. Everything what passes though all the decisions is classified to the positive class or a simple threshold γ can be used on the strong classifier response function as in Algorithm 2. The responses of the classifier are distinguishable for the non-maximum suppression algorithm by their f_T value.

For the non-symmetric sequential strategy (5.11) the average evaluation time under the positive class, $\bar{T}_{S,+1}$, is approximately T since all the weak classifiers need to be evaluated and only false negative decisions reduce the time slightly. The strategy is thus characterised by its error rates, α_S and β_S and the average evaluation time under the negative class, $\bar{T}_{S,-1}$.

6 Fast Face Detection

In this chapter, the WaldBoost learning framework is applied to the face detection problem. It is an example of application where the classification is repeated many times in a limited amount of time. Thus a single classification needs to be fast. The face detection problem is also used to demonstrate different properties of the WaldBoost algorithm and to show how to choose the learning parameters.

The scanning-window approach to the face detection adopted in the following experiments is illustrated in Figure 6.1. A detection window is swept over the image at varying scales and a face vs. background classifier is applied for each window position and scale. The final detections are merged from different scale sweeps. The task of the classifier is to distinguish between a face (positive class) and a general background (negative class) sub-image at current scanning position.

Relevant state of the art methods are discussed and compared to the WaldBoost detector at the end of Section 6.2.

6.1 WaldBoost Applied to Face Detection

First, details of the training setup are described, then the test protocol is defined and, finally, test results presented. In the following experiments the non-symmetric version of WaldBoost is used as discussed in Section 5.4.3, to explore the large and complex negative class and to preserve the hard-collected positive samples.

Sample Pool

The positive sample pool for the experiments was created by downloading 11349 cropped faces from an early version of the Betaface project web page[1]. The set contains images of human faces in approximately frontal view with aligned eye positions and a small surrounding of the head visible. The size of images is 100×100. Some of the images are at the edge of what one would call a frontal face. Our experience from annotating frontal faces in images is that when context is available, people perceive faces as frontal even in the cases where cropped faces alone look non-frontal. Since miss-detection of such faces is often perceived as a failure, all the downloaded faces are kept for training. Generally, the training set is rather difficult and challenging.

To avoid overfitting to the fixed eyes position, the images are further synthetically rotated in the range $(-5°,+5°)$. The face bounding box is slightly reduced by the rotation but still captures

[1] `http://www.betaface.com`

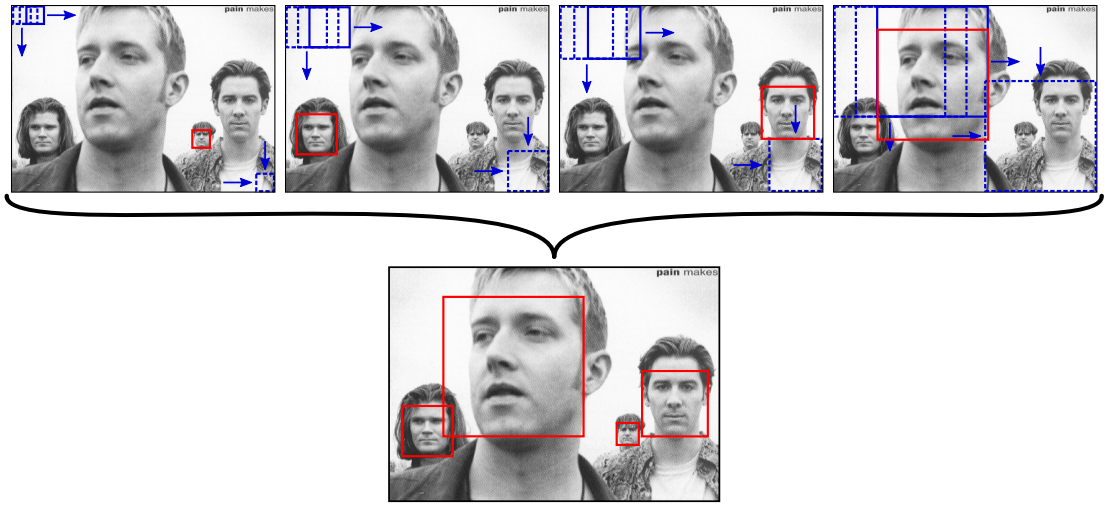


Figure 6.1: The scanning window approach to face detection. A detection window is swept through the image and an object vs. background classifier is evaluated at each position and scale.

enough edges typical for frontal faces (chin, cheeks, fringe and ears). Figure 6.2 shows random examples of synthetically rotated positive samples in the Betaface dataset.

As discussed later in Section 6.1, the evaluation of used Haar-like features is sensitive to rounding errors. Keeping the sizes of the training images all the same, the classifier would overfit to typical rounding errors for the given size. Since the detection process runs over different scales, this overfitting may hurt the performance significantly. Thus, each training image is put into the sample pool randomly scaled down with minimal size 24×24 pixels.

To further increase the robustness and to increase the number of positive samples, horizontally mirrored versions of positive samples are put into the pool.

Negative sample pool was created from more than 3000 images not containing faces from the non-skin database [37]. A manual database inspection was needed since the original dataset is quite imprecise and contains faces (and also skin-colour). From this set of images about 190 millions of sub-windows were collected to the negative sample pool. The negative samples are used in their original sub-window resolution.

The collected sample pool represents the face class under a wide range of acquisition conditions. The positive samples include images with various lighting conditions, facial expression, skin colour, background or age. The negative samples contain images of natural scenes, offices, man-made objects, or animals. The idea is to train a general purpose WaldBoost face detector working almost anywhere (e.g. as a part of a camera software). Obviously, when the application area of the detector is known, the respective positive and negative samples could be collected, making the training task more focused and simpler.

Figure 6.2: Random examples of positive samples from the Betaface dataset used for training the WaldBoost classifiers.

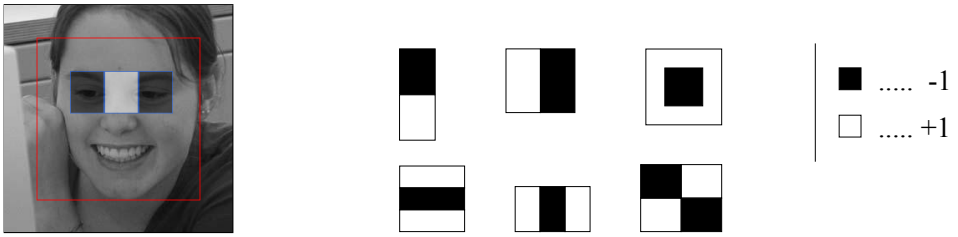


Figure 6.3: Haar-like features. Each feature is parametrised by its type, size and position in the scanning window. To evaluate a Haar-like feature, sums of pixel intensities over the white and black sub-rectangles are computed (see equation (6.1)).

Features and Weak Classifiers

We use the Haar-like feature set proposed by Viola and Jones [81] plus the center-surround feature proposed by Lienhart [45]. The six used types of features are shown in Figure 6.3, right. Each features is parametrised by its position in the scanning window (red rectangle in Figure 6.3, left), its size and type. This parametrisation gives 26,136 features for a 24×24 window.

The feature response is computed as

$$q(\mathbf{x}) = \left(\frac{\sum_{i \in W_q} \mathbf{x}(i)}{|W_q|} - \frac{\sum_{i \in B_q} \mathbf{x}(i)}{|B_q|} \right) / \sigma_w \tag{6.1}$$

where $\mathbf{x}(i)$ is the pixel intensity at pixel i, B_q and W_q are the black and white regions of feature q, and σ_w is the standard deviation of pixel intensities in the scanning window. The sums and σ_w can be computed in a constant time independently of the feature size using the integral image representation [69, 81].

In training, each feature is linked to one domain-partitioning weak classifier (see Figure 4.3). The leftmost and the rightmost bins are found to contain 5 % of the training samples each and the remaining interval is partitioned into 8 equally wide bins.

In spite of the feature response normalisation by their area, the response varies slightly depending on the feature alignment with the pixel grid and the window size. Also σ_w is more noisy for smaller windows. To reduce these effects, only rather small number of weak classifier bins is used and scaled versions of training samples are introduced to the sample pool as discussed earlier.

Other features like LBP [56, 23], histograms of gradients [11] or granular features [34] could be used in the same framework and different types of weak classifiers like decision trees [10] or RBF units [60] could be examined. Although they would influences the performance ([10, 44]), the main focus of the thesis is on the sequential architecture. To make all the experiments comparable and transparent neither we do enhance the feature set nor do we introduce another weak classifiers.

Test Set

The WaldBoost detector was tested on the MIT+CMU dataset [64] consisting of 130 images containing 507 labelled faces. The dataset contains images of varying quality, resolution and scene types. The set is very challenging as also hand-drawn and playing card faces, Start Trek masks, not fully frontal faces and one very schematic drawing on a complex background are present.

To avoid comparison on difficult hand-drawn faces for which the detector was not trained, many authors reported their results on a subset of the MIT+CMU dataset of their choice. This makes direct comparison of the methods reported in literature difficult. Here we use the full set for better comparability with future methods.

Another problem is a possible overfitting to the MIT+CMU dataset. Since all the detectors are trained using different training set, it is not clear how they would generalise to another data. To avoid this problem, we do some cross-validation tests and train a cascaded detector similar to [81].

Test Protocol

The most common measure of a face detector's performance is the receiver operating characteristic (ROC) curve. In ROC the detection rate is plot against the number of false positives. However, there is no agreement in literature how to decide which reported detection is correct and which one to count as a false positive. Most of the papers do not describe the used methodology.

Here, a simple procedure to distinguish true positives from false positives adopted. A detection is considered to be correct if its size is in the range of 50 % – 150 % of the ground truth bounding box and their centres are not further than 30 % of the ground truth bounding box size. When two detections fulfil this criterion, only one is considered to be good, the other is counted as a false positive.

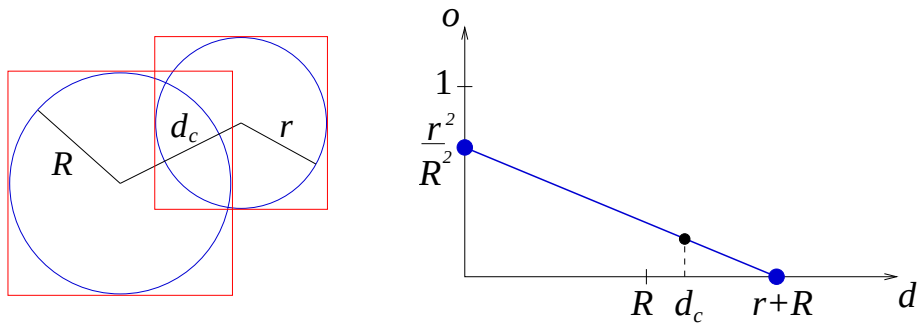


Figure 6.4: The non-maximum suppression algorithm scheme for two detections.

Another detector quality compared in the experiments is the evaluation speed, $\bar{T}_{S,-1}$ (see equation (2.3)). It is computed as an average number of weak classifiers (i.e. features or measurements) evaluated on a given dataset per scanning window position. Note, that due to non-symmetric problem formulation only the average evaluation time for class -1 is used.

For all experiments the WaldBoost detector is applied at minimal scale 24×24 pixels. To detect faces at different scales, the detector is repeatedly scaled up with 1.25 scale factor. The scanning window is shifted by 1/24 of its size in both horizontal and vertical directions.

Non-Maximum Suppression

The non-maximum suppression (NMS) algorithm is an important part of the detection process as it influences the detection results. The classifier needs to be robust to small variations in position and scale and thus it does not produces a single peak around each detection but rather a cluster of detections. NMS merges these multiple detection into one.

There exist a wide range of NMS algorithms (e.g by designed heuristics [64, 82], by Mean-Shift [26] or by avoiding NMS at all [40]), but the most common way is to use some overlap measure on detections, group those with overlap higher than some threshold and keep only the detection with maximal response f_T in each group (or the average). When the overlap measure is computed efficiently and detections are kept in KD-tree structure, this approach is able to handle even high number of detections like in Chapter 7.

A possible overlap measure is the bounding rectangles overlap. However, this measure is not center-symmetric, making certain overlaps more important than the others. Moreover, it becomes complicated when in-plane rotated detections are considered. Computing the overlap of circles inscribed to the bounding boxes does not have these problems. However, it involves goniometric functions which are computationally demanding. To avoid the goniometric functions, we propose a linear approximation to the circles overlap.

The overlap computation is schematically shown in Figure 6.4. Each detection is represented by a circle inscribed to the corresponding scanning window (Figure 6.4, left). For two such circles, let us denote the radius of the smaller circle as r, the radius of the bigger one as R, and the distance of the circle centres as d_c. Exact overlap can be easily computed in two cases. First,

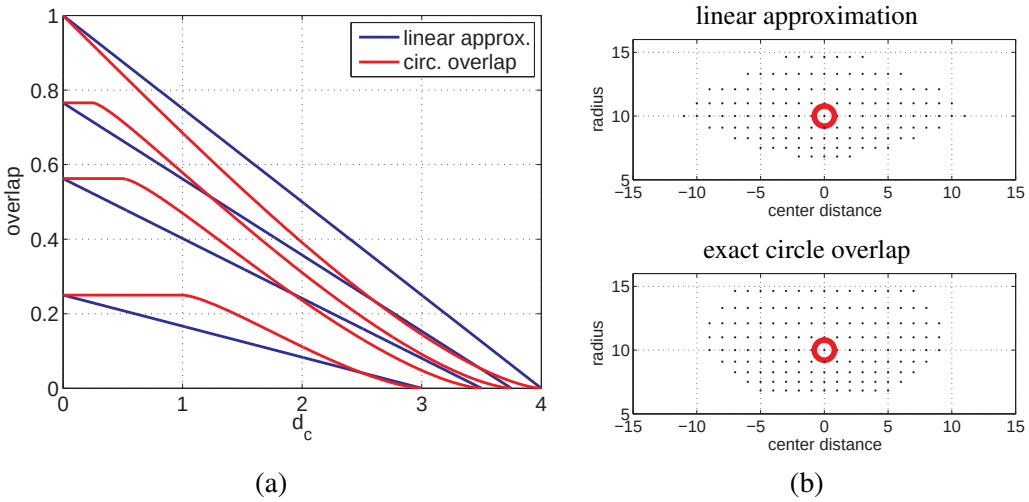


Figure 6.5: The proposed non-maximum suppression overlap measure is compared to the exact circle overlap. (a) Both measures are shown for four circles radius ratios (from bottom 1:2, 3:4, 7:8 and 1:1) as a function of circle centers distance, d_c. (b) Black dots show scanning window neighbouring positions (in radius-distance space) with overlap higher than a fixed threshold (0.4). The red circle marks the central position for which the neighbourhood is computed.

when the circle centres coincide, the overlap is $o = r^2/R^2$. It equals to one for two circles of the same radius and decreases as the radiuses become different. Second, when two circles have just one point in common ($d_c = r + R$), the overlap is zero. These two situations are marked by blue dots in Figure 6.4, right. Linear interpolation (blue solid line in Figure 6.4, right)

$$o = \frac{r^2}{R^2}\left(1 - \frac{d_c}{r+R}\right) \tag{6.2}$$

is used to approximate the overlap between these two states.

The proposed overlap measure approximation is compared to the exact circle overlap in Figure 6.5. The left plot shows the difference of the measures for several circle radii ratios (multiple plots) as a function of the centres distance, d_c. One interesting property of the proposed approximation is that it penalises off-center position of the smaller circle fully contained in the bigger one. The exact circle overlap can not distinguish these situations.

In Figure 6.5b the neighbourhoods for a fixed overlap threshold are depicted for both measures. Due to aforementioned property of the proposed approximation, top and bottom ends of the neighbourhood (large scale difference) are more narrow. Otherwise, both measures create similar size of the neighbourhood.

Training Process Details

The size of the training set $\mathcal{T}$ is set to 10,000 in the experiments. One half of the training set contains positive samples, the other half contains the negative ones. QWS+ and uniform sampling are compared in following experiments.

In the training and evaluation the sub-windows with standard deviation of pixel intensities below five are not considered. Since the standard deviation needs to be computed in advance to evaluate any feature, this test has no slow-down effect on the evaluation. Nevertheless, it avoids the feature response normalisation to emphasise the noise in these uniform intensity patches which would make them sometimes look as a face.

As discussed in Section 5.4.2, the sum of the weights of positive and negative samples is kept in constant ratio 1:1 during the training.

6.2 Experiments

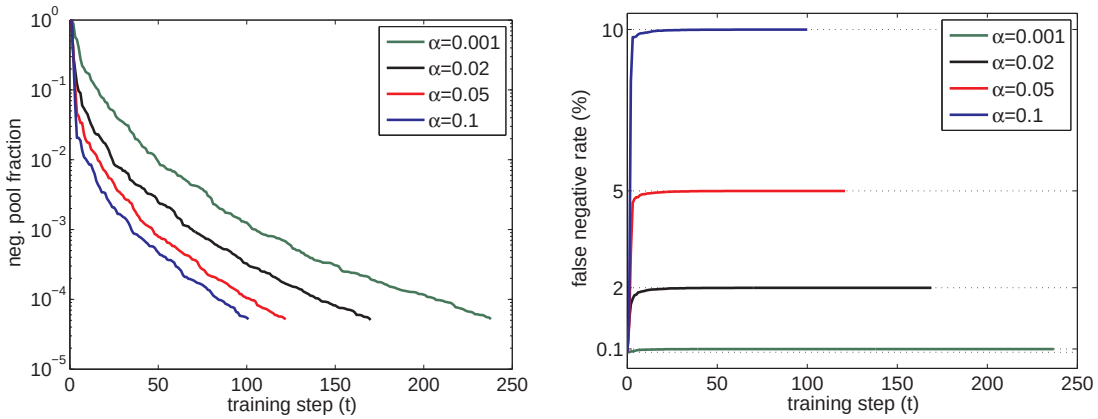
Several experiments were conducted to provide better insight into the WaldBoost learning. Also a comparison to the state of the art algorithms is presented to demonstrate capabilities of the WaldBoost algorithm.

False positive and false negative rates in training. Setting different α parameter for training results in different training process characteristics. As shown in Figure 6.6a, the negative samples are pruned exponentially in the number of training steps. With decreasing α, the number of steps needed for successful classification of the whole negative sample pool increases. Note that when only a small fraction of the negative sample pool is available, the training is switched to build a monolithic classifier (i.e. the decision thresholds are set to $\pm\infty$).

Figure 6.6b shows the evolution of the false negative rate during training. The increase of the false negative rate is also exponential in the number of steps, up to the allowed limit defined by the value of α.

Attained false negative rate. The detection rates attained in the detection experiments do not need to be the same as the rates prescribed to the learning algorithm. There are several factors influencing the results. First, the statistics of the training set and the test set need not be the same. For example, consider a detector trained for human faces which is run on a dataset of hand-drawn faces or on a dataset with a special illumination. This factor usually lowers the detection rates. Second, in the scanning window detection process, the classifier is given multiple chances to detect a face at neighbouring positions due to its robustness to small shift and scale changes. Due to this robustness, the attained detection rate may be higher than $1 - \alpha$. Third, the non-maximum suppression algorithm is designed mainly to reduce the false positive rate by merging multiple detections, however for two close detections or for a very strong false positive vs. weaker true positive, the correct detection may be suppressed.

To avoid these influences of the detection process on the reached rates, the following experiment was conducted. To compare the attained false negative rate with the required false negative



(a) (b)

Figure 6.6: Pruning of the negative sample pool during training.

rate α, we run a 5-fold cross-validation on the positive sample pool. So, 4/5 of the positive samples are used in training and the rest is used for validation. In the validation test the cropped faces are used directly, which avoids the multiple detections problem and the NMS algorithm. The test was performed for $\alpha = 0.1, 0.05, 0.02$ and 0.01. The length of the classifiers was $T = 100$.

The average reached false negative rates and their standard deviation are shown in Table 6.7. While the reached rates are slightly biased but close to expected values for $\alpha = 0.1, 0.05$ and 0.02, the learning clearly reached its limits with $\alpha = 0.001$. What happened is that the number of positive samples in the pool was not sufficient to provide stable statistics in threshold estimation. The statistics are estimated at the tail of the distribution where very few samples fall, resulting in unreliable estimates.

For comparison, the false negative rates computed on the MIT+CMU dataset are shown in the last column of Table 6.7. Note that these numbers are computed during *detection*. The definition of bounding boxes differs slightly in both sets, so computing the false negative rate directly on the cropped faces results in very high false negative rates. The false negative rates are reported for final threshold $\gamma = -\infty$ and correspond to the top-most point on the ROC curve in Figure 6.9 (indicated by a horizontal line). Clearly, the detection process and different test set statistics influence the rates significantly.

QWS+ vs. uniform sampling. In Step 4 of the WaldBoost learning algorithm (Algorithm 3) a new training set $\mathcal{T}$ is sampled from the sample pool $\mathcal{P}$. As discussed in Section 5.4.1, the sampling algorithm influences the training process and the final classifier. Figure 6.8 depicts a comparison of two WaldBoost detectors, one using the uniform sampling and the other using the quasi-random weighted sampling + trimming (QWS+) to sample the training set. The results confirm the conclusions from [39]. The QWS+ sampling selects better samples resulting in stronger weak classifiers which are combined into faster classifier with lower false positive

α	**mean(FN)**	**std(FN)**	**MIT+CMU**
0.1	11.23	0.67	7.9
0.05	6.22	0.49	6.1
0.02	2.99	0.29	5.5
0.001	1.30	0.28	5.1

Figure 6.7: 5-fold cross-validation false negative rate results (in percents) on the Betaface dataset for classifiers of length $T = 100$ and varying α parameter. The last column reports the false negative rates attained on the MIT+CMU dataset (cf. the horizontal lines in Figure 6.9.

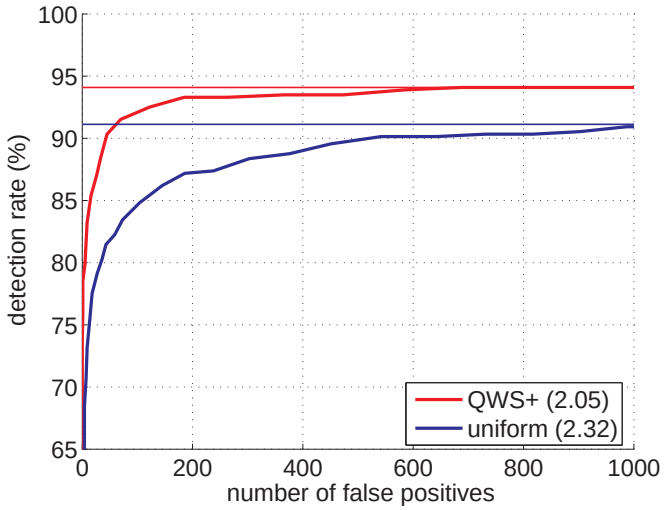


Figure 6.8: Comparison of QWS+ and uniform sampling of the training set $\mathcal{T}$ from the sample pool $\mathcal{P}$. ROC computed on the MIT+CMU dataset for classifiers trained with $\alpha = 0.1$ and $T = 500$. The evaluation speed, $\bar{T}_{S,-1}$, for each method is shown in parentheses. The horizontal lines show the reached detection rate without applying the final threshold $\gamma = -\infty$.

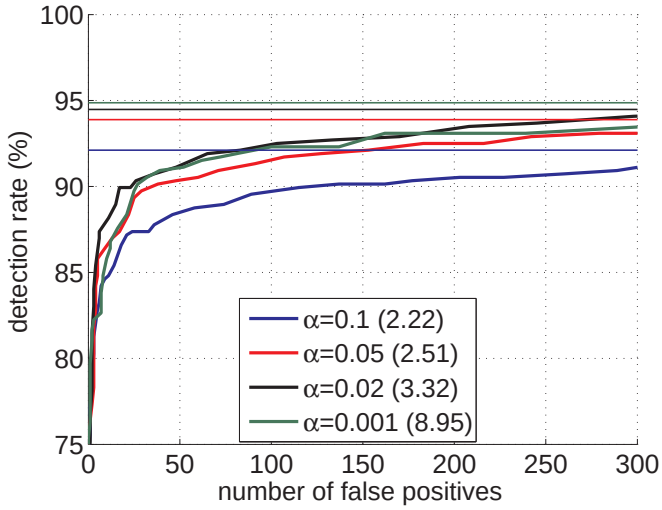


Figure 6.9: ROC on MIT+CMU dataset for WaldBoost detectors trained for four values of the α parameter.

rates. Learning with QWS+ needs only 73 steps to prune the whole negative sample pool while the learning with uniform sampling needs more than 200 steps.

Comparison of the trained classifiers on MIT+CMU. The classifiers trained with different false negative rate requirements are compared on the MIT+CMU dataset in Figure 6.9. With decreasing value of the α parameter the ROC curve improves and the detection speed decreases. At some point, here for $\alpha = 0.001$, no more significant improvement is gained due to limitations of the power of selected features and the training set used, and only the speed is affected. The classifier trained with $\alpha = 0.02$ reaches both, good detection rates and the evaluation speed, and is thus used in the following comparisons.

WaldBoost vs. cascade. The algorithm from [82] was used to train a cascaded classifier on the same training data as the WaldBoost classifier. The first seven stage classifiers are build with the number of weak classifiers fixed (2, 10, 25, 25, 50, 50, 50). Next seven stages are trained by an automatic threshold search with allowed stage false negative rate 0.001 and allowed stage false positive rate 0.5. When the negative sample pool is pruned to a too small set for bootstrapping, the remaining samples are used to train the last monolithic stage. The final cascade consists of 15 stages and 2187 weak classifiers (which is comparable to the WaldBoost classifier containing 2000 weak classifiers).

The number of weak classifiers in the first seven stages is not optimal for our problem. These lengths were manually tuned for a different training set in [82]. Nevertheless, a comparison of the true positive rates of the first stages in our cascade with results in [82] shows that our training

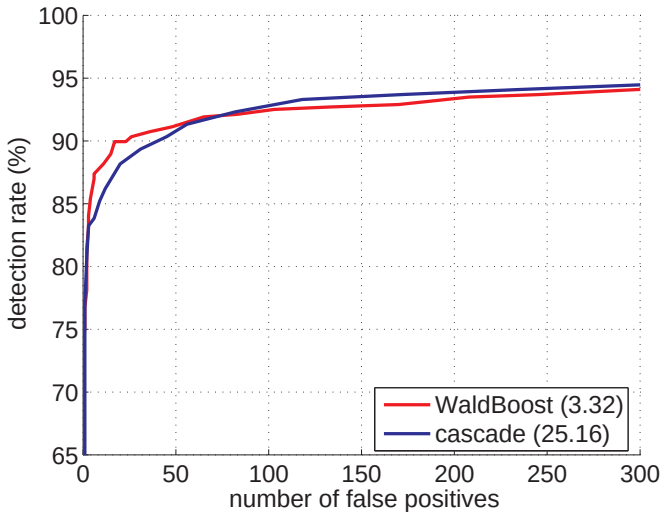


Figure 6.10: WaldBoost and cascaded detector comparison on the MIT+CMU dataset. The WaldBoost classifier with $\alpha = 0.02$ and $T = 2000$ is compared to cascaded classifier trained according to [82] with 15 stages and 2187 weak classifiers. The evaluation speed, $\bar{T}_{S,-1}$, for each method is shown in parentheses.

dataset is more difficult (lower true positive rates are reached). Thus the manual tuning would not probably lead to a faster classifier than $T_S = 8$ reported in [82].

The results in Figure 6.10 show that both methods produce classifiers with similar detection rates, however the WaldBoost classifier is 7.5× faster without any manual tuning.

Comparison to the state of the art. The performance of face detectors is commonly compared on the MIT+CMU dataset using the ROC curve. Since Viola and Jones paper [81], some of the authors report also the average evaluation time, T_S. Table 6.1 summarises available results of the most successful methods from recent years.

Most of the methods extended the original idea of the cascaded classifier by introducing various classifier architecture improvements [42, 85, 47, 10, 7], some introduced stronger features or weak classifiers [85, 70, 10], and some introduced a different organisation of the cascade evaluation implementation [70]. Each detector was trained on different training set of varying difficulty, so a direct and fair comparison is difficult and a commentary is necessary. Here we try to discuss on different aspects of the compared methods and to explain the observed results.

Compared to the original Viola and Jones results [81, 82], the WaldBoost algorithm improves the evaluation speed and detection rates especially for smaller false positive rates (Viola and Jones (2004)a). Although the better detection rates are caused probably by using a more difficult training set and by using strong weak classifiers (cf. the comparison of WaldBoost and cascaded classifier above), still the evaluation speed improvement is significant. The WaldBoost rates are

even slightly better than the weighted classifier (Viola and Jones (2004)b) which evaluates three cascaded classifiers and requires a consensus of two to report a detection.

Li and Zhang's FloatBoost algorithm [42] was aimed at minimising the number of weak classifiers needed in the cascade. It allows to optionally remove already added weak classifiers if it reduces the training error. The parentheses around the evaluation time in the table indicate that the speed was measured in another FloatBoost implementation [86]. The better selection of weak classifiers improved the rates compared to Viola and Jones but decreased the evaluation speed significantly. The WaldBoost detector outperforms the FloatBoost in both detection rates and speed.

Schneiderman [70] also built a cascaded classifier but his first stage classifier used so called "feature-centric" evaluation – feature evaluations are re-used across multiple scanning windows. Besides evaluation speedup, this brings implicitly contextual information at the feature level. He also used different, more powerful features. Due to this improvements his detection rates are significantly better than that of WaldBoost detector[2]. However, Schneiderman reported run time which is even longer than that of Viola and Jones in spite of more powerful computer used. Using the WaldBoost algorithm instead of the cascade training could possibly improve this inefficiency.

In their work [85], Wu et al. proposed a nesting structure classifier (see the boosting chain architecture in Section 3.2) which is half the way between the cascaded classifier and fully sequential WaldBoost classifier. The used Haar-like features and the domain-partitioning weak classifiers implementation are comparable to ours. Although the average evaluation time is not reported, their implementation is able to detect even in-plane and out-of-plane rotated faces in several frames per second. Together with their further work [34] this is probably the best published (multi-view) face detector. The WaldBoost detector has similar detection rates for small false positive rates but is worse for higher false positive rates.

Luo [47] optimised the detection rate and the false positive rate of a cascade trained by Viola and Jones algorithm. His rates are higher than that of Viola and Jones but still below the rates reached by WaldBoost. Luo did not report the average evaluation time. However, since his main objective was the error rates minimisation, the average evaluation time is likely to be higher than that of Viola and Jones.

The SoftCascade algorithm by Bourdev and Brandt [7] optimises all the parameters of the boosting chain-type classifier (i.e. selected weak classifiers, false positive and false negative rates and speed). Nevertheless, both reported detectors achieved much higher average evaluation times than other methods. The decrease in the detection rates for the faster of the detectors indicate that with higher evaluation speed the rates would be much lower. The reached rates are only slightly higher that that of the WaldBoost detector.

The experiments in Brubaker et al. [10] showed that a sequential classifier can greatly improve the average evaluation speed while preserving the detection rates. The first detector in the table uses discrete weak classifiers and by several *automatic* sequentialising steps its speed is reduced to $T_S = 8$ which is comparable to manually tuned Viola and Jones cascade. The detection rates

[2] Images with hand-drawn faces were removed from the test in [70]. The detection rates on the full dataset would be probably slightly lower.

of the Brubaker's detector are slightly better but still lower than the WaldBoost rates. To increase the accuracy, a stronger CART-based weak classifier is introduced. The weak classifier uses two Haar-like features, so it is slower to evaluate. Using this weak classifiers, the rates improved even above the WaldBoost detection rates. Since the weak classifiers are more complex but stronger, the evaluation speed of the final detector is hard to predict.

State of the art comparison summary. Although a common test set used for face detectors comparison is available, it is non-trivial to judge various methods based purely on their ROC performance. In practice, the training set, weak classifiers used, details of the classifier's architecture, but also the implementation skills play some role. Taking as many of these influences into account, we tried to comment on the results reached by different methods. In spite of the lack of a clear face detection benchmark settings, the WaldBoost algorithm was shown to gain competitive detection rates and to be superior in the reached average evaluation time. Examples of detections on the MIT+CMU dataset are shown in Figure 6.11. An on-line accessible demo of the detector is available at `http://cmp.felk.cvut.cz/demos/FaceDetection/`.

6.2.1 Application Speed Optimisation

The speed of the face detection application depends highly on the average evaluation time $\bar{T}_{S,-1}$. However, when comparing run-times of different detectors, the comparison can be easily dominated by the detector's *implementation overhead*. The code optimisation offers plentiful opportunities for speedup. Apart from optimising the data structures, using integers instead of doubles when possible, playing with the compilation flags, ... two speed up techniques used in our application are worth mentioning.

First, the whole classifier evaluation is a sequence of feature evaluations followed by a decision test. Both operations are very simple. Adding few more auxiliary operations for each features evaluation, like a function call, may spoil the running time completely. Thus we automatically convert each detector into a fully linear code without any loops, function calls or unnecessary ifs. This expansion assures that the most demanding operations are the feature evaluation, the weak classifier evaluation and the decision tests. In our case, the expansion revealed that the auxiliary operations in the non-linear implementation needed about the same time as the classifier evaluation itself.

Another factor influencing highly the speed of the application is the number of sub-windows examined. The number of sub-windows depends on the minimal detectable scale and the amount of shift and scale change in the sub-window sweep. For applications like human-computer interface, where a single high-resolution face is expected, increasing the smallest detectable face size increases the speed by a factor of 2–10.

6.3 Summary

In this chapter, the WaldBoost learning framework has been applied to the face detection problem. It has been shown, how the training with different parameters influences the resulting classifier. A comparison to the cascaded classifier demonstrated the WaldBoost learning ability to reduce the average evaluation time while keeping the detection rates high. The WaldBoost

Detector	$\bar{T}_{S,-1}$	False positives															
		1	2	6	9	10	26	31	39	41	46	50	57	65	77	78	95
Viola and Jones (2004)a [82]	8	–	–	–	–	76.1	–	88.4	–	–	–	91.4	–	92.0	–	92.1	92.9
Viola and Jones (2004)b [82]	3*8	–	–	–	–	81.1	–	89.7	–	–	–	92.1	–	93.1	–	93.1	93.2
Li and Zhang (2004) [42]	(18.9)	–	–	–	–	83.6	–	90.2	–	–	–	–	–	–	–	–	–
Schneiderman (2004) [70]	n/a	–	–	89.7	–	–	–	–	–	–	95.7	–	–	–	–	–	–
Wu et al. (2004) [85]	n/a	–	–	–	–	90.1	–	–	–	–	–	–	94.5	–	–	–	–
Luo (2005) [47]	n/a	–	–	86.6	–	87.4	–	90.3	–	–	–	91.1	–	–	–	–	–
Bourdev (2005)a [7]	37	83.6	85.6	90.9	91.9	–	93.5	–	–	94.3	–	–	–	–	–	–	–
Bourdev (2005)b [7]	25	–	–	–	–	–	–	91.7	92.1	–	–	92.7	–	–	92.9	–	–
Brubaker et al. (2008)a [10]	8	–	–	81.7	–	85.8	–	88.8	–	–	90.1	90.1	–	90.3	–	90.5	90.9
Brubaker et al. (2008)b [10]	n/a	–	–	89.1	–	89.5	–	91.3	–	–	91.9	91.9	–	92.1	–	92.1	92.3
WaldBoost	3.32	77.5	81.5	87.4	87.5	88.2	90.3	90.5	90.7	90.7	91.1	91.1	91.3	91.9	92.1	92.1	92.5

Table 6.1: State of the art comparison. The table summarises detection rates of the state of the art methods on the MIT+CMU dataset as a function of the number of false positives. When available, the average evaluation time, $\bar{T}_{S,-1}$, is shown. The WaldBoost classifier was trained for $\alpha = 0.02$ and $T = 2000$.

Figure 6.11: Examples of WaldBoost detections on the MIT+CMU dataset.

detector has been compared to the state of the art methods with competitive results and superior evaluation times.

7 Learning Fast Emulators of Binary Decision Processes

We show, how the WaldBoost algorithm can be used for speeding up existing binary decision processes, such as detectors or two-class classifiers. The WaldBoost algorithm produces a fast and accurate approximation of the original process. The approach is successfully demonstrated on two commonly-used interest point detectors. Nevertheless, the approach is general and is applicable to other areas, e.g. edge detection.

For the problem of efficient approximation, the speed of the trained classifier becomes important – a property directly optimised by very few machine learning methods. The WaldBoost learning algorithm was adopted as it handles the precision-speed trade-off automatically and produces an efficient sequential classifier minimising the decision time while guaranteeing the predefined emulation precision. The user influences the emulation process by defining suitable feature sets from which the emulator is built and by specifying constraints on the classifier's precision.

One uncommon feature of our setting is the training set size. Since our objective is to learn an emulator of an existing binary-decision process, labelled training samples are obtained by running the process on unlabelled data. If unlabelled data are easily accessible, which is common, a training set of arbitrary size can be collected at effectively zero cost. The speeding up problem thus becomes a problem of learning the algorithm's outputs on a very large training set while optimising the classification speed.

We demonstrate the framework by emulating two interest point detectors, Hessian-Laplace [55] and Kadir-Brady saliency detector [38]. The Hessian-Laplace is a state-of-the-art detector of blob-like structures. Moreover, a handcrafted simplified version of Hessian-Laplace called SURF [3], designed for maximum speed, is available for comparison. The Kadir-Brady detector incorporates entropy measure to find salient regions which has been successfully used in several recognition tasks [15, 88].

7.1 Learning Interest Point Detectors — State of the Art

There has been much work on the general interest point detection problem [53]. To our knowledge, learning techniques have been applied only to parameter tuning, not to the whole process of interest point detector design. Lepetit and Fua [41] treated interest points matching as a classification problem, learning the descriptor. Rosten and Drummond [63] used learning techniques to find parameters of a hand-designed tree-based Harris-like corner classifier. Their motivation was to speed-up the detection process, but the approach is limited to the Harris corner detection. Martin et al. [48] learned a classifier for edge detection, but without considering the decision

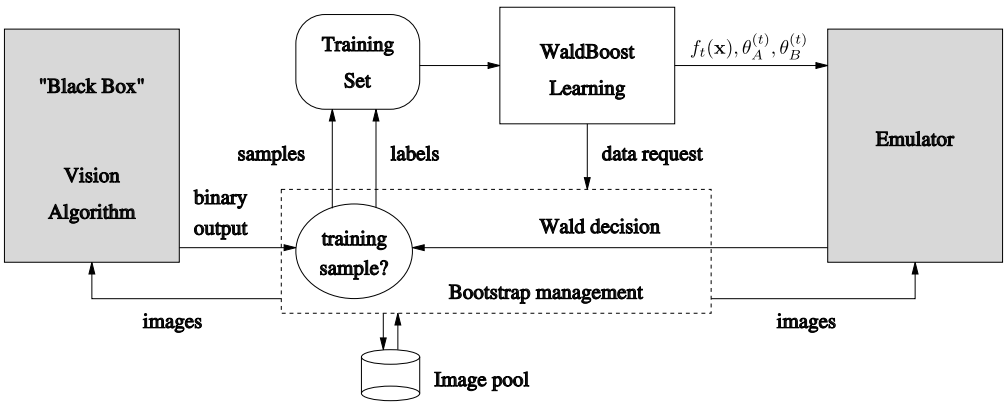


Figure 7.1: The proposed learning scheme.

time and with significant manual tuning. They tested a number of classifier types with the conclusion that a boosted classifier was comparable in performance to other classifiers and was preferable for its low model complexity and low computational cost.

The most closely related approach to our method is that of Dollár et al. [13] who use learning techniques to train an edge detector. The paper shows impressive examples of applications of such detector. Nevertheless, Dollár et al. were primarily concerned with the accuracy of the detector and did not consider speed. There has also been significant interest in speeding up various interest point detectors manually, i.e. without training. Grabner et al. proposed a fast version of the SIFT detector [31] and Bay et al. proposed a fast approximation-based interest point detector called SURF [3].

7.2 Emulating a binary-decision black box algorithm with WaldBoost

The main idea of the proposed approach is to look at an existing algorithm as a black box performing some useful binary decision task. The black box algorithm is run on a large dataset of images which provides almost unlimited number of training samples which are used to train a sequential classifier emulating the black box algorithm behaviour. The user's optimisation effort is thus transformed into a much simpler task of finding a suitable set of features which are used in the WaldBoost training.

The main components of the proposed learning system are shown in Figure 7.1. The black box algorithm provides positive and negative outputs that form a labelled training set. The WaldBoost learning algorithm (see Chapter 5) builds a classifier sequentially and when new training samples are needed, it bootstraps the training set by running the black box algorithm on new images. Only the samples undecided by the current classifier are used for further training. The result of the process is a WaldBoost sequential classifier which emulates the original black box algorithm.

The training loop uses the fact that the black box algorithm can provide practically unlimited number of labelled training samples. Note that this is in contrast to commonly used human labelled data which are difficult to obtain. The bootstrapping technique [78] is used to effectively update the training set.

In the context of fast black box algorithm emulation, what distinguishes training for different algorithms is the feature set $\mathcal{F}$ (see Algorithm 3). A suitable set has to be found for every emulated algorithm. The set $\mathcal{F}$ can be very large and does not need to be homogeneous, i.e. it may contain Haar-like features [81], LBP [56, 23], histograms of gradients, etc. The WaldBoost algorithm selects a suitable subset while optimising the time-to-decision. WaldBoost minimises the average number of evaluated measurements which is the same as minimisation of time-to-decision only when computational complexity of the different types of features is (roughly) the same. The condition is satisfied by the feature set $\mathcal{F}$ adopted in the experiments.

7.3 Emulated scale invariant interest point detectors

In order to demonstrate the approach, two similarity-invariant interest point detectors have been chosen: (i) Hessian-Laplace [55] detector, which is a state of the art similarity-invariant detector, and (ii) Kadir-Brady [38] saliency detector, which has been found valuable for categorisation [15, 88], but is about 100× slower than the Hessian-Laplace detector. Binaries of both detectors were downloaded from the web page [51]. We followed standard test protocols for evaluation as described in [53]. Both detectors are similarity-invariant (not affine), so the detection can be easily implemented by running a sequential test at each position and scale in the scanning window approach [81].

For both detectors, the set $\mathcal{F}$ includes the Haar-like features proposed by Viola and Jones [81], plus a centre-surround feature from [45], which has been shown to be useful for blob-like structure detectors [31]. Haar-like features were chosen for their high evaluation speed (due to integral image representation) and because they have a potential to emulate the Hessian-Laplace detections [31]. The only difference to the original Viola and Jones feature set is that the feature response is not normalised by a window standard deviation since the intensity contrast is important for both Hessian-Laplace and Kadir-Brady detectors.

For the entropy-based Kadir-Brady saliency detector emulation, however, the Haar-like features were not sufficiently accurate. To overcome this we introduced "variance" features based on the integral images of squared intensities. They are computed as an intensity variance in a given rectangle.

The input to the non-maximum suppression differs from that obtained in the original detectors. Instead of having a real-valued feature response over whole image, sparse responses are returned by the WaldBoost detector. The accepted positions get the real-valued confidence value f_T, but the rejected positions have the "confidence" f_t around the $\theta_A^{(t)}$ value depending on the time t when they have been rejected. These values are incomparable, thus a typical quadratic interpolation and a local maximum search cannot be applied. Instead, the non-maximum algorithm described in Section 6.1 is used.

7.4 Experiments

Two detectors are emulated in the experiments: Hessian-Laplace [55] and Kadir-Brady [38] saliency detector. The Hessian-Laplace detector's simplicity allows easier analysis of obtained results. The Kadir-Brady detector incorporates entropy measure to find salient regions. It performs rather poor in classical repeatability tests [53] but has been successfully used in several recognition tasks. However, its main weakness for practical applications is its very long computation time in order of minutes per image. Standard versions of the detectors provided by their authors were downloaded from the interest point detection web page [51].

To collect positive and negative samples for training, an emulated detector is run on a set of images of various sizes and content (nature, urban environment, hand drawn, etc.). To create the sample pool we used 1300 images randomly chosen from the non-skin image database introduced in [37]. The detector assigns a scale to each detected point. Square patches of the size twice the scale were used as positive samples. Negative samples were collected from the same images at positions and scales not covered by positive samples.

The size of the training set $\mathcal{T}$ was 10,000 (half positive and half negative samples) in all experiments. The training set was sampled from the pool $\mathcal{P}$ by the quasi-random weighted sampling + trimming method (QWS+) [39]. The QWS+ sampling has been shown to reduce the variance of hypothesis error estimate and to improve the classifier performance compared to other sampling strategies. Moreover, with QWS+ sampling, AdaBoost performance becomes relatively insensitive to the training set size.

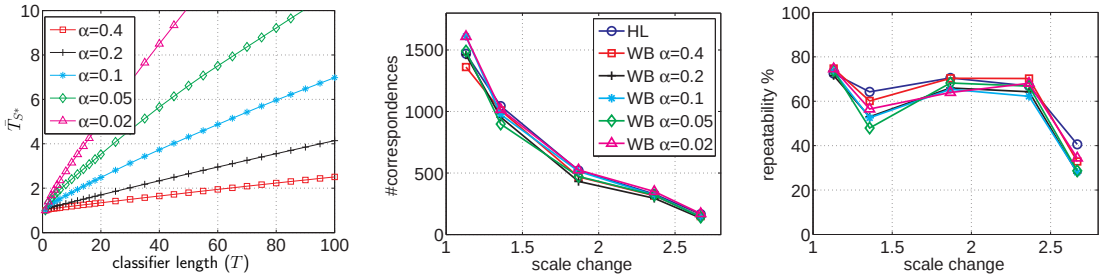
7.4.1 Hessian-Laplace emulation

The Hessian-Laplace detector was used with threshold 1000 to generate the training set. The value was empirically chosen to achieve similar number of detections as in [53]. The same threshold was used throughout all the experiments for both learning and evaluation.

The detector has been assessed in standard tests proposed by Mikolajczyk et al. [53]. The ground truth is given by a homography between the first and the other images in the sequence. The tests are based on two measures: (i) the repeatability measure, (ii) the matching score.

(i) Repeatability measure. To assess the quality of an interest point detector in varying acquisition conditions of the same scene the repeatability measure is used [53]. The measure is defined for two sets of elliptical regions – one set for one image. It is computed as *the ratio between the number of region-to-region correspondences and the smaller of the number of regions in the pair of images*. The mutual correspondence of two regions is claimed when the overlap error is smaller than some threshold. The measure takes into account several other technical issues such as uniqueness of matches and is fully defined by a Matlab script [51]. In all experiments, the overlap error threshold is fixed to 40 % as in most of the experiments in [53].

(ii) The matching score test aims at predicting performance of the detectors in matching and correspondence finding applications. The matching score, defined in [53], is the number of correct matches divided by the smaller number of correspondences in the common part of the two images. A pair of elliptical regions is counted as a *correct match* if (1) their overlap error is smaller than 40 %, and (2) their descriptors are sufficiently similar (for details, see [53]).



(a) (b) (c)

α	0.02	0.05	0.1	0.2	0.4
♯det	11472	9427	7106	4203	2624

(d)

threshold	500	1000
♯det	3549	2207

(e)

Figure 7.2: Selecting the false negative rate α. (a) The average evaluation speed for several values of α. All compared detectors are able to achieve similar number of correspondences (b) and repeatability score (c) – measured for $T = 20$ for all detectors on the BOAT sequence. (d) The number of detections of the WaldBoost emulator on the first image from the BOAT sequence as a function of the α parameter, (e) the number of detections of Hessian-Laplace as a function of the final threshold.

Selection of the false negative rate α. The value of α balances the trade-off between WaldBoost detector speed and precision. Figure 7.2a-c shows performance of the detector for several α values on the BOAT sequence. The value of α also significantly influences the number of detections before the final thresholding by γ (Figure 7.2d).

For a certain range of α values, it is possible to set the final threshold γ (see Algorithm 2) to reach the number of correspondences similar to that of the emulated detector (Figure 7.2b). With such threshold γ, the repeatability and the number of correct correspondences is almost identical for all tested values of α throughout the test sequence (Figure 7.2c).

Increasing α leads to faster evaluation (Figure 7.2a) but also to fewer detections (Figure 7.2d) before imposing the final threshold γ. In some applications it may be useful to produce more detections by changing the γ threshold.

Similarly to the original detector, the WaldBoost emulator imposes a threshold on the classifier response. We set α to 0.2 as a compromise: the classifier is already very fast (see Table 7.1) and yet the user can still control the number of detections by changing the γ threshold similarly to the original detector (Figure 7.2e). Thus the value $\alpha = 0.2$ is used in all following experiments. The final threshold γ is the same in all experiments and is set empirically so that the detector produces similar number of detections as the original Hessian-Laplace detector.

Classifier length. Empirically we set the length of the classifier to $T = 20$ (number of weak classifiers). Longer classifiers slow down the evaluation (see Figure 7.2a) and do not bring significant improvement in performance.

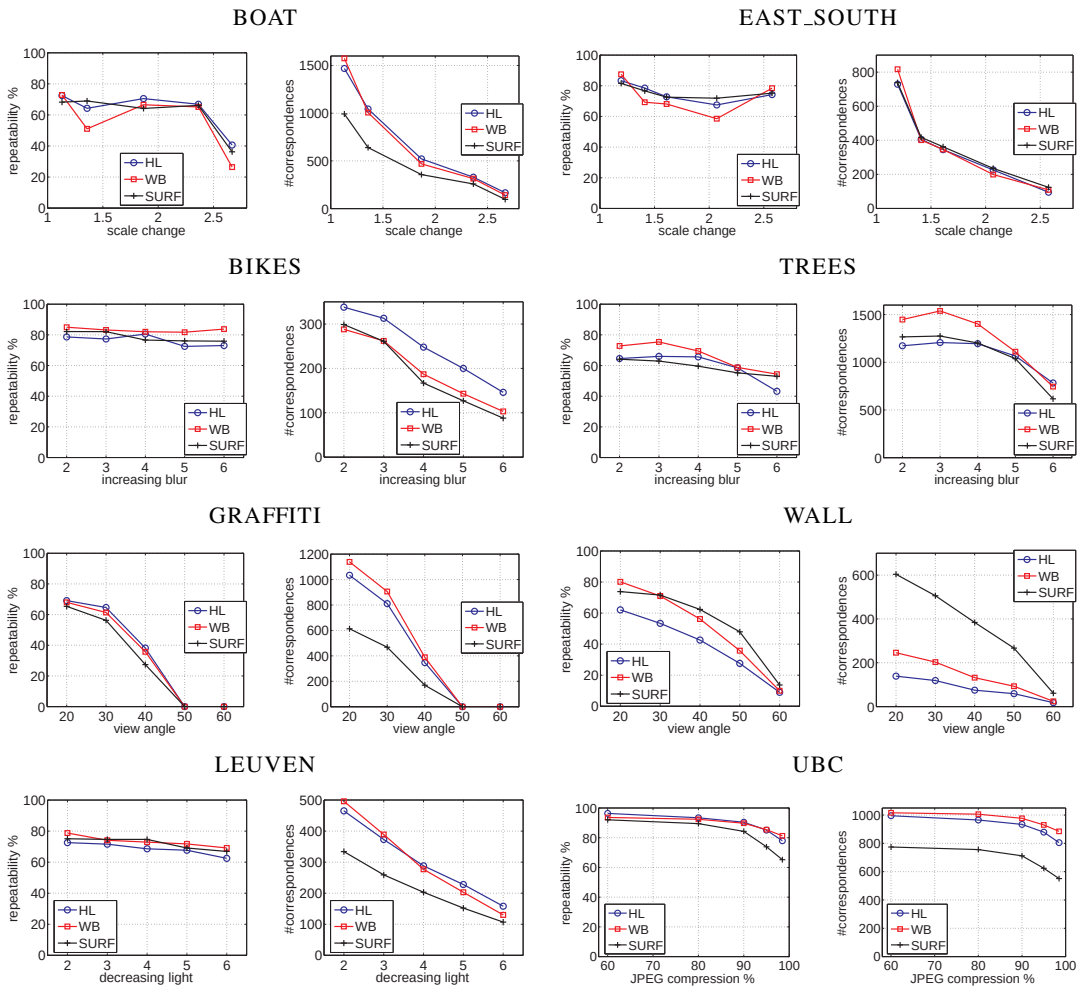


Figure 7.3: Repeatability comparison of the Hessian-Laplace detector, its WaldBoost emulation and the SURF detector on Mikolajczyk's dataset.

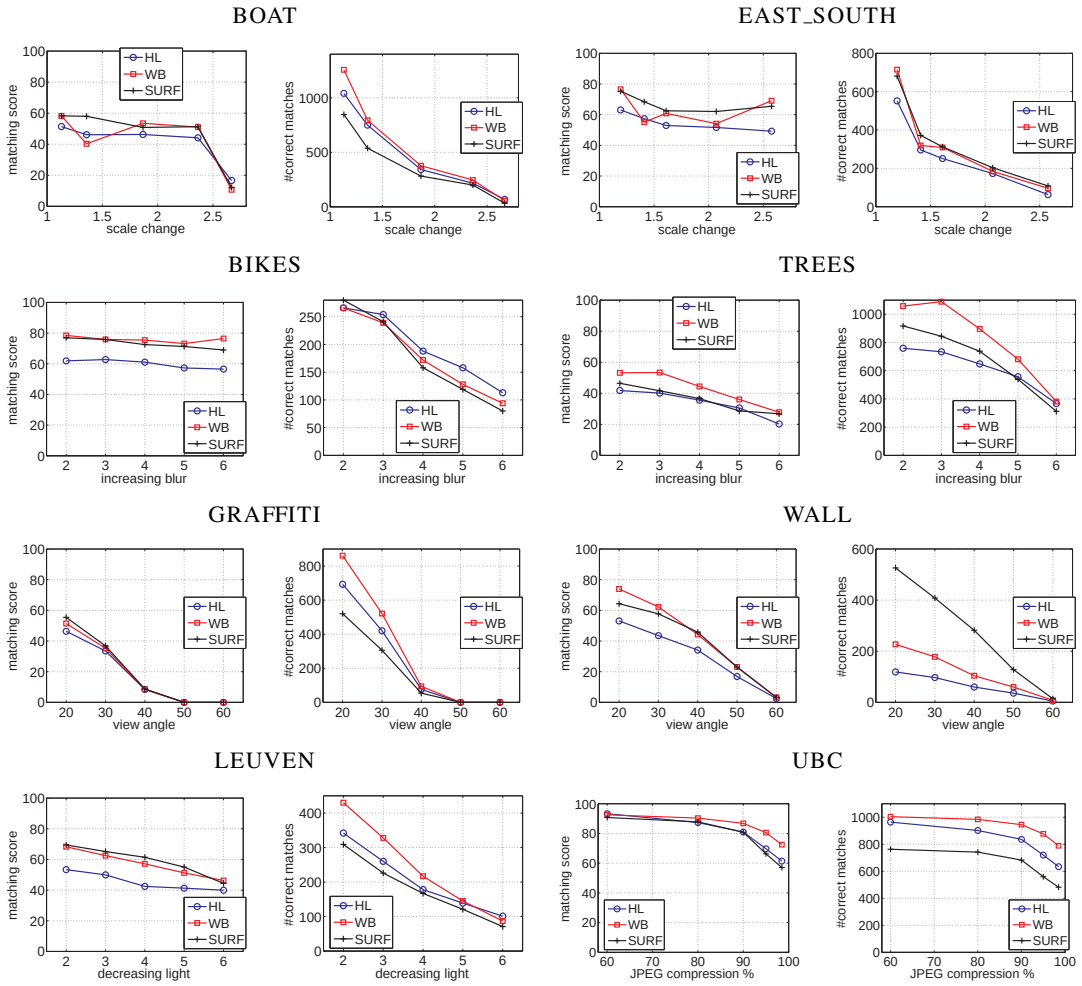


Figure 7.4: Matching score comparison of the Hessian-Laplace detector, its WaldBoost emulation and the SURF detector on Mikolajczyk's dataset.

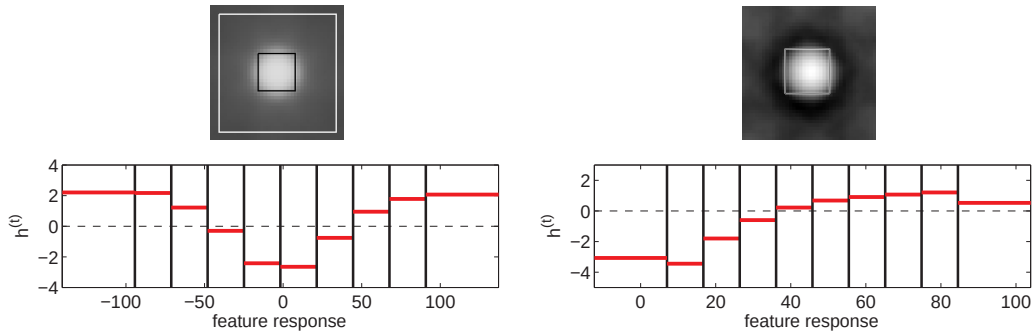


Figure 7.5: *Top row.* First centre-surround and variance feature found in WaldBoost Hessian-Laplace (left) and Kadir-Brady (right) emulated detectors. The background image is visualised as $E(|\mathbf{x}_i - 127.5|)$ and $E(\mathbf{x}_i)$ respectively, where $E()$ is the average operator and $\mathbf{x}_i$ is the i-th positive training example. *Bottom row.* Bin responses in the corresponding domain-partitioning weak classifiers (see Figure 4.3).

Repeatability. The repeatability measure of the trained WaldBoost detector has been compared with the original Hessian-Laplace detector on standard image sequences with variations in scale and rotation, blur, affine deformation, light change and JPEG compression from [54]. The results are shown in Figure 7.3. The WaldBoost detector achieves similar repeatability and number of correspondences as the original Hessian-Laplace detector.

Matching score. For the same sequences, the matching score of the Hessian-Laplace detector ant its WaldBoost emulator is shown in Figure 7.4. The WaldBoost detector achieves slightly better matching score than the original algorithm.

Speed. The WaldBoost classifier evaluates on average 1.7 features per examined position and scale. Unsurprisingly, this is much less than any reported speed for face detection [73]. The evaluation times are compared in Table 7.1. The WaldBoost emulator is about nine times faster than the Hessian-Laplace detector with a rather careful design [52].

Classifier structure. The Hessian-Laplace detector finds blob-like structures. The structure of the trained WaldBoost emulation should reflect this property. As shown in Figure 7.5, the first selected weak classifier is of the centre-surround type and gives high responses to blob-like structures with high contrast between central part and its surrounding (the feature value is average intensity in the central part minus average intensity in the surrounding part).

Coverage. The output of the trained WaldBoost emulation of Hessian-Laplace is compared to the original algorithm in Figure 7.6a. As in the repeatability experiment two sets of detections are compared – the original detections and the WaldBoost emulator detections (with $\gamma = -\infty$). Since the comparison works on a single image, the ground truth transformation matrix is identity.

The white circles show the original detections with a correspondence found among the WaldBoost detections. The black circles show the original detections not found by WaldBoost. Note that most of the missed detections have a correct detection nearby, so the corresponding image

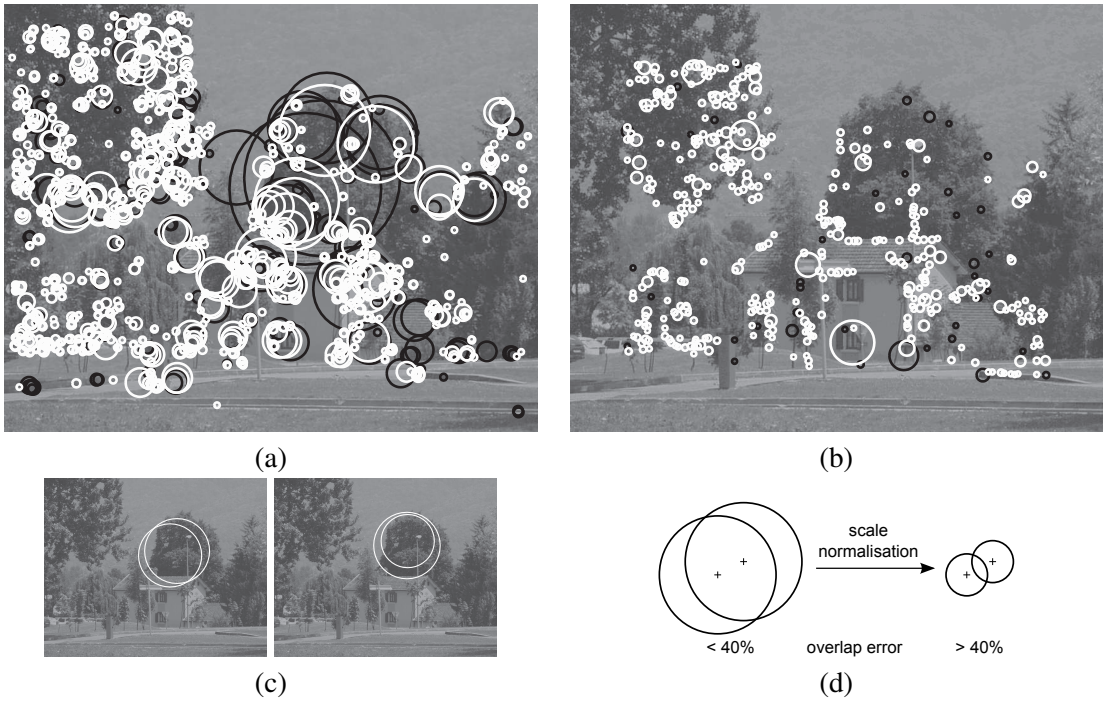


Figure 7.6: Comparison of the outputs of the original and WaldBoost-emulated (a) Hessian-Laplace and (b) Kadir-Brady saliency detectors. The white circles show repeated Hessian-Laplace detection. The black circles highlight the original detections not found by the WaldBoost detector. Note that for most of missed detections there is a nearby detection on the same image structure. The accuracy of the emulation is 80 % for Hessian-Laplace and 90 % for Kadir-Brady saliency detector. Note that the publicly available Kadir-Brady algorithm does not detect points close to image edges. (c) Missed Hessian-Laplace detections (left) and manually found corresponding WaldBoost detections (right). (d) They are not found as correspondences, because Mikolajczyk's overlap function prefers smaller detections (see the discussion in the text).

structure is actually found. The percentage of repeated detections of the original algorithm is 80 %.

The WaldBoost detector may seem to miss consistently the large regions. Figure 7.6c shows manually selected WaldBoost regions close to the original detections – the "tree blob" is in fact detected. The real problem is in the way the correspondence overlap is computed. To compute the overlap of two detected points, Mikolajczyk [53] first normalises their scale to 30 pixels. This way, the problem of unnecessary large regions which would almost always have large overlaps is avoided. However, as shown in Figure 7.6d, this normalisation returns small overlap when large regions are only slightly misplaced. This problem is general and *appears in*

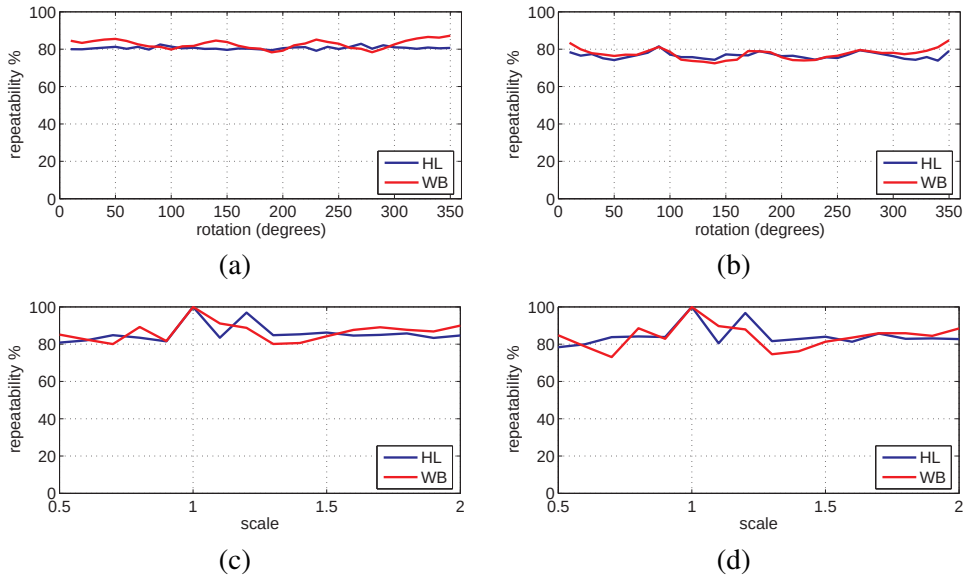


Figure 7.7: Rotation and scale invariance of the WaldBoost Hessian-Laplace emulator. *Top row:* Repeatability on *rotated* first images from (a) BOAT, and (b) EAST_SOUTH sequences for the Hessian-Laplace detector (HL) and its WaldBoost emulator (WB). *Bottom row:* Repeatability on *scaled* first images from (c) BOAT, and (d) EAST_SOUTH sequences.

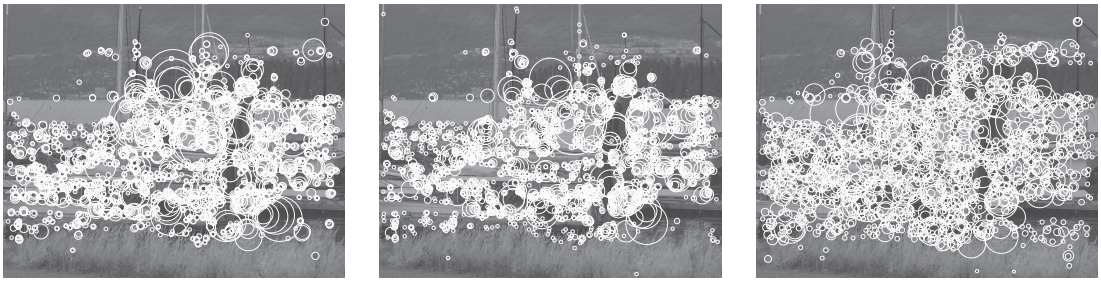
all region detection papers which use the Mikolajczyk's repeatability measure. To conclude, the real emulation accuracy is in fact higher than 80 %.

Rotational invariance. One of the properties of the emulated Hessian-Laplace detector which should be preserved is its rotational invariance. A learning approach can achieve rotational invariance with non-rotationally invariant features by introducing synthetically rotated positive samples into the training set. The results in Figure 7.7 (top row) show that the rotational invariance is preserved even without introducing synthetic training samples. This is probably a consequence of the large training pool which is available. Instead of introducing rotated samples synthetically, the statistics are covered by collecting huge number of samples.

Scale invariance. Similarly, the detector invariance to scale changes has been tested. The emulated detector achieves similar scale invariance as the original algorithm as shown in Figure 7.7 (bottom row).

Comparison to SURF. The WaldBoost emulator has been compared with the SURF detector [3] which is a simplification of the Hessian-Laplace detector, manually designed for maximum speed. The SURF is commonly used as a good compromise between speed, accuracy and repeatability.

The comparison of the repeatability and the matching score of all three detectors is shown in Figure 7.3 and Figure 7.4. All the detectors has been set to produce similar number of detections



(a) (b) (c)

Figure 7.8: Comparison of Hessian-Laplace (a), its WaldBoost emulator (b) and SURF detector (c) outputs on the first image from the BOAT sequence. WaldBoost returns similar distribution of points as the emulated Hessian-Laplace. The SURF points are distributed differently.

on the first image of the EAST_SOUTH sequence. Neither of the fast detectors approximates the original detector perfectly. Yet, both could be said to achieve similar statistics as the original Hessian-Laplace detector, deviating slightly at different sequences.

The evaluation speeds of the detectors are compared in Table 7.1. The WaldBoost detector achieves similar evaluation speed as the manually tuned SURF detector. However, since most of the computational components are the same in both detectors, the average evaluation time $\bar{T}_{S,-1} = 1.7$ for WaldBoost and $\bar{T}_{S,-1} = 3$ for SURF suggests that further code optimisation of the WaldBoost detector could lead to even faster implementation.

An important difference between the SURF detector and the Hessian-Laplace WaldBoost emulator is that the first one is *a simplification* while the other is *an emulation*. The SURF produces different set of regions compared to the Hessian-Laplace detector. This could be verified by computing the coverage as in Figure 7.6. For the SURF detector only 49.7 % coverage is reached compared to 80 % of the WaldBoost detector. The difference in detectors outputs is shown in Figure 7.8.

7.4.2 Fast saliency detector

The emulation of the Kadir-Brady saliency detector [38] uses the same image pool for training as the WaldBoost Hessian-Laplace emulator. The saliency threshold of the original detector was set empirically to 2 to collect a sample pool of a reasonable size. Higher value of threshold also helps to limit the positive examples only to those with higher saliency. As opposed to the Hessian-Laplace emulation, where rather low threshold was chosen, it is meaningful to use only the top most salient features from the Kadir-Brady detector since its response corresponds to the importance of the feature.

The Haar-like feature set was extended by the "variance" feature described in Section 7.3. The

	Hessian-Laplace	**Kadir-Brady**
original	0.9s	1m 48s
SURF	0.09s	—
speed-up	10×	—
$\bar{T}_{S,-1}$	3	—
WaldBoost	0.10s	0.76s
speed-up	9×	142×
$\bar{T}_{S,-1}$	1.7	2.2

Table 7.1: Speed comparison on the first image (850×680) from the BOAT sequence. The speed-up on another images is similar.

training was run for $T = 20$ (training steps) with $\alpha = 0.2$ and $\beta = 0$ as in the Hessian-Laplace experiment.

Publicly available version of Kadir-Brady detector has several drawbacks which need to be considered in the experimental evaluation. Due to relatively wide search for local maximum in the scale space, no detections near the image border are found. This results in a strip around image border where no detections are returned (see Figure 7.6b). Also the scale range of detections is limited. In all following experiments, WaldBoost emulator detections are filtered by the same restrictions for the comparison reasons. However, the WaldBoost emulator of the Kadir-Brady detector *does not have these restrictions* inherently.

Repeatability and matching score. The same experiments as for the Hessian-Laplace detector have been performed. The repeatability and the matching score of the Kadir-Brady detector and its WaldBoost emulation on BOAT and EAST_SOUTH sequences are shown in Figure 7.9. Similar performance to the Kadir-Brady detector is reached for similar number of correspondences and correct matches on both sequences.

Speed. The main advantage of the emulated saliency detector is its speed. The classifier evaluates on average 2.2 features per examined location and scale. Table 7.1 shows that the emulated detector is about 142× faster than the original detector.

Classifier structure. Our early experiments showed that the Haar-like features are not suitable to emulate the entropy-based saliency detector. With the variance features, the training was able to converge to a reasonable classifier. In fact, the variance feature is chosen for the first weak classifier in the WaldBoost ensemble (see Figure 7.5). The bin responses of the weak classifier show that higher variations are preferred.

Coverage. The outputs of the WaldBoost saliency detector and the original algorithm are compared in Figure 7.6b. The coverage of original detections is 90 %.

Rotational and scale invariance. Invariance to rotation and scale changes of the WaldBoost emulator and the Kadir-Brady detector are compared in Figure 7.11. Due to very different

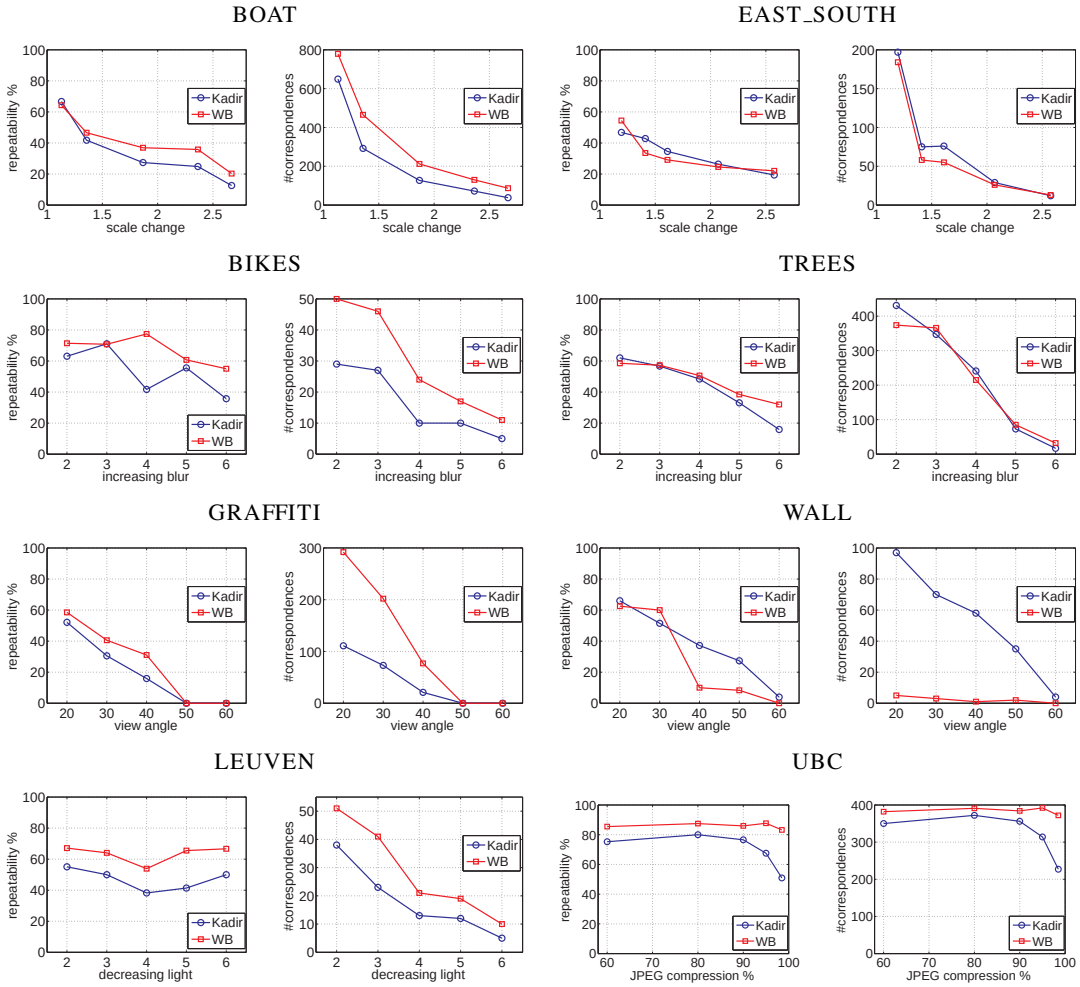


Figure 7.9: Repeatability comparison of the Kadir-Brady detector and its WaldBoost emulation on Mikolajczyk's dataset.

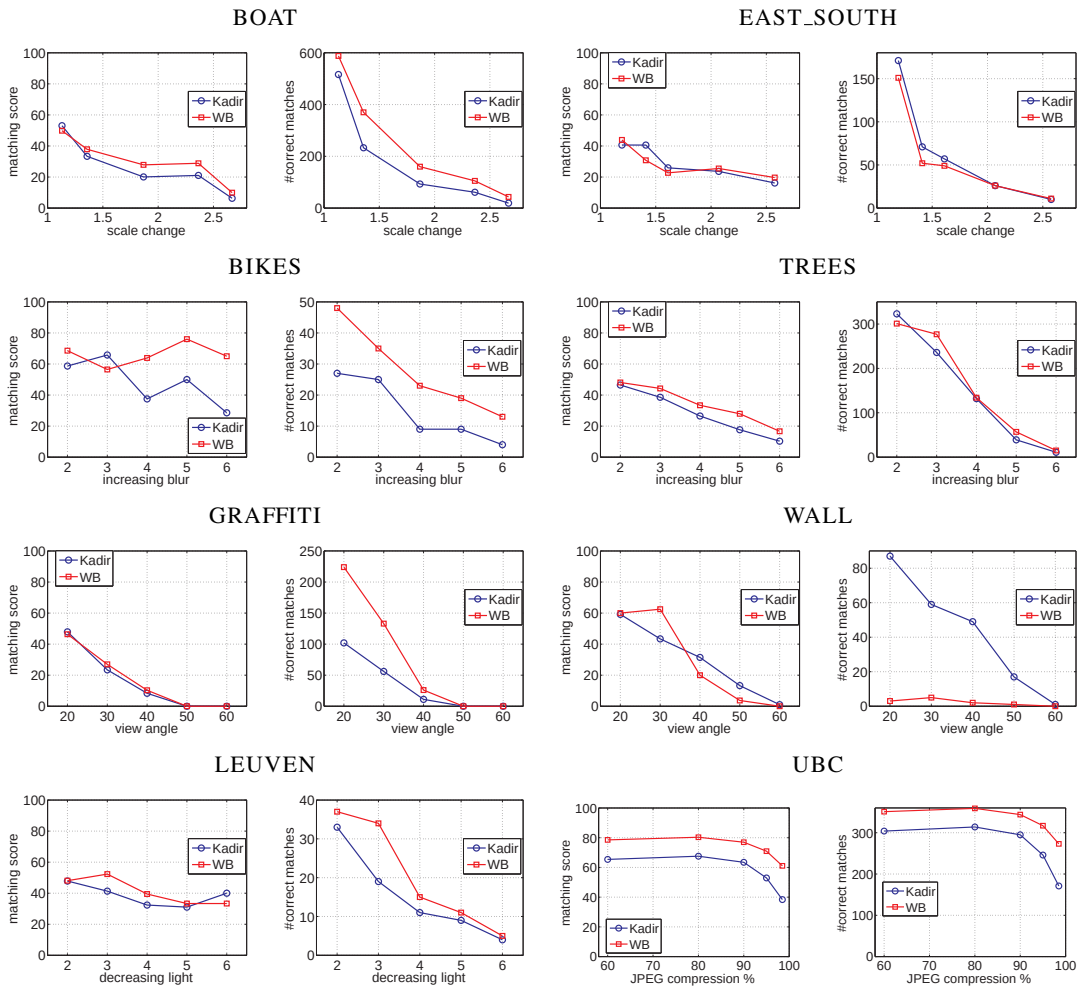


Figure 7.10: Matching score comparison of the Kadir-Brady detector and its WaldBoost emulation on Mikolajczyk's dataset.

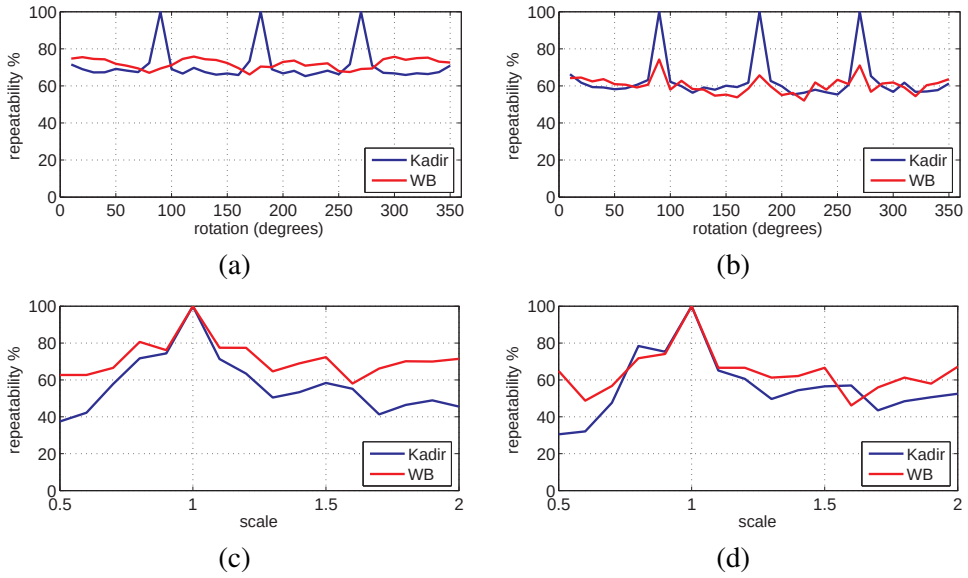


Figure 7.11: Rotation and scale invariance of the WaldBoost Kadir-Brady emulator. *Top row:* Repeatability on *rotated* first images from (a) BOAT, and (b) EAST_SOUTH sequences for the Kadir-Brady detector (Kadir) and its WaldBoost emulator (WB). *Bottom row:* Repeatability on *scaled* first images from (c) BOAT, and (d) EAST_SOUTH sequences.

approaches in computing the detectors responses (Haar-like features vs. entropy), the WaldBoost emulator is not able to reach perfect rotation invariance on images rotated by 90 degrees but is able to keep similar rotational invariance otherwise. Moreover, the feature-based approach of the WaldBoost emulator results in slightly better scale invariance of the detector. This can be probably explained by the instability of the entropy based Kadir-Brady detector especially at small scales where the probabilities are difficult to estimate. It is shown also in [33] that their difference-of-Gaussians detector is more robust to a range of transformations than the Kadir-Brady detector.

7.5 Summary

To conclude, a general learning framework has been proposed for speeding up existing binary decision processes by a sequential classifier which is learnt by the WaldBoost algorithm. Two interest point detectors, the Hessian-Laplace and the Kadir-Brady saliency detector, served as examples of emulated algorithms.

The WaldBoost emulator of the Hessian-Laplace detector was able to detect points with similar repeatability and slightly higher matching score while keeping the rotational and scale invari-

ance of the original detector. Moreover, the WaldBoost emulator was able to increase nine times the speed of detection compared to the original detector. When compared to the manually tuned SURF detector, similar repeatability, matching score and evaluation speed characteristics are reached. However the WaldBoost detector emulates the Hessian-Laplace detector significantly more closely.

Also, the WaldBoost training is able to emulate Kadir-Brady detector generally with similar repeatability, matching score and robustness to rotation changes, while improving slightly its scale invariance. But, most importantly, the decision times of the emulated detector are about 142 times lower than that of the original algorithm. That opens new possibilities for using the Kadir-Brady detector in time sensitive applications.

The proposed approach is general and can be applied to other algorithms as well. For future research, an interesting extension of the methodology would be to train an emulator which not only guarantees output similar to an existing algorithm but which also possesses some additional quality like insensitivity to certain acquisition conditions (e.g. motion blur) or maximum performance in a particular environment or task.

8 On-line WaldBoost

In this chapter, we introduce an on-line extension of the WaldBoost algorithm. On-line boosting [25] proved its usefulness in many practical applications like object tracking [27, 87] or background modelling [28] has been used to improve object detectors over time (e.g. [29, 84]). The on-line WaldBoost algorithm contributes to this field by solving two important problems of on-line learning: (i) optimisation of the classifier evaluation speed, and (ii) automatic determination of the classifier complexity necessary for given problem.

To overcome the first point, Wu and Nevatia [84] investigated to use the cascade approach in the on-line boosting framework. However, their approach uses many heuristic decisions and is not well founded in the theory.

All current approaches for on-line learning need the number of weak classifier to be given in advance [57, 25]. However, in tasks where the decision problem changes over time, like in object tracking, it is impossible to specify the classifier complexity in advance. A common approach is to train complex classifiers which can handle all situations but this is less effective when the task becomes easier.

Experiments on a visual object tracking task show that the on-line WaldBoost algorithm is able to automatically adapt the classifier complexity and speed to changing problem difficulty.

8.1 On-line Boosting for Feature Selection

The goal of training in both off-line [17] and on-line [57] boosting is to minimise the training error by selecting and combining a set of "weak" classification algorithms $h^{(t)}(\mathbf{x}) : \mathcal{X} \to \{+1, -1\}$ into a strong classifier $f_T(\mathbf{x})$[1]

$$f_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \ . \tag{8.1}$$

The binary classification is then based on the sign of the strong classifier response function f_T.

The main differences between off-line and on-line AdaBoost training is the way the training data are obtained and how the strong classifier is built. In the off-line training all the data are available in advance. The on-line training uses one training sample at a time. To build a classifier in the off-line training one weak classifier is added each training round, while in the on-line training the strong classifier is initialised at the beginning and is updated by each training sample.

[1] In this chapter we use the discrete version of AdaBoost [18] to avoid overfitting in the relatively simple classification problem.

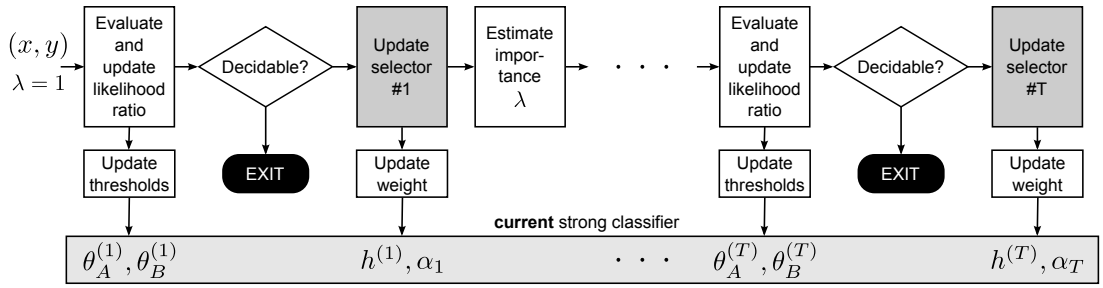


Figure 8.1: Training scheme of the on-line WaldBoost algorithm.

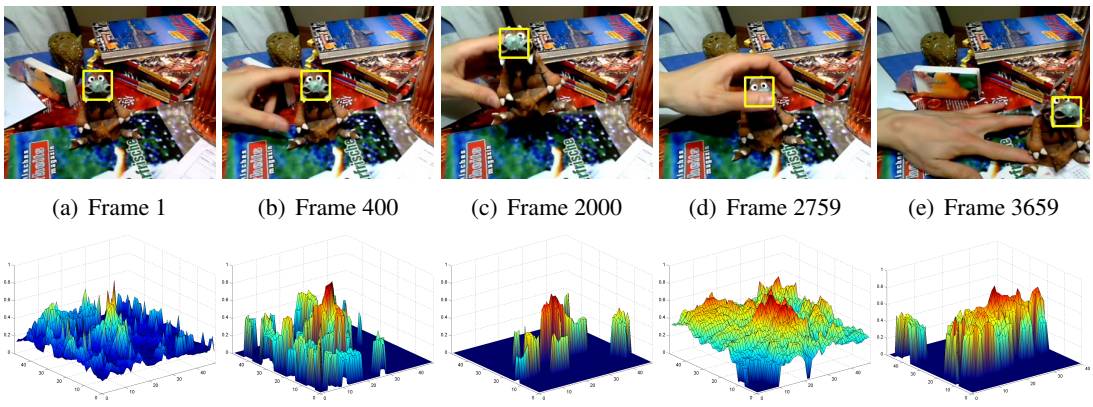
The base for the on-line WaldBoost algorithm is the on-line boosting for feature selection proposed by Grabner and Bischof [25]. The main idea is to perform on-line boosting on *selectors* rather than on the weak classifiers directly. Each selector $\mathcal{L}$ keeps a set of L weak classifiers $\mathcal{L}_t = \{h_1^t(\mathbf{x}), \ldots, h_L^t(\mathbf{x})\}$. The selectors are initialised with a random set of weak classifiers and in each selector one weak classifier, $h^{(t)}$, is chosen as active. To estimate the weak classifiers error an importance (difficulty) λ of a sample is propagated during training through the set of T selectors.

When a new training sample $(\mathbf{x}, y), y \in \{-1, +1\}$ arrives, the importance weight λ of the sample is initialised to $\lambda = 1$. The selectors are updated sequentially. In the updated selector, error of each weak classifier is evaluated on the training sample and their error estimate is updated depending on the outcome of the evaluation and the importance weight λ. The weak classifier with the smallest estimated error is selected in the selector. Next, the corresponding voting weight α_t of the selected weak classifier and the importance weight λ of the sample are updated and the next selector is updated. The weight λ increases if the sample is misclassified by the current selector or decreased otherwise. Finally, a strong classifier is build as a linear combination of T weak classifiers selected in individual selectors.

8.2 On-line WaldBoost

The proposed on-line WaldBoost algorithm combines the WaldBoost algorithm described in Chapter 5 and the on-line boosting for feature selection from Section 8.1. The general training scheme is shown in Figure 8.1. As in Section 8.1 the selectors are updated using the actual training sample and its importance, the weak classifier is chosen as the classifier minimising the estimated error in the selector, and the sample importance weight λ reflects the difficulty of the sample. The main difference is that the training can be terminated earlier if the SPRT conditions hold, i.e. the sample is used for updating only those selectors to which it is passed undecided.

In order to find the Wald thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$ the likelihood ratio $\hat{R}(\mathbf{x})$ from equation 5.8 has to be estimated. In the off-line training the statistics are computed on the sample pool. The on-line training offers an elegant way to compute an unbiased estimate of the statistics using the given sample only. The idea is to use the current training sample first as a test sample (not seen before) to update the SPRT statistics before it is used for updating the strong classifier.



(a) Frame 1 (b) Frame 400 (c) Frame 2000 (d) Frame 2759 (e) Frame 3659

Figure 8.2: Tracking of an object (1st row) and the classifier response map within the search window (2nd row). Values equal to 0 mean early rejection, i.e. saving of the computation time.

The probabilities $p(\rightarrow t|C)$ can be estimated by computing the portion of samples seen so far and not decided until t-th selector. The densities $p(f_t(\mathbf{x})|y = C, \rightarrow t)$ are estimated from the samples which are not decided until the t-th selector only. In our implementation they are approximated by Gaussians. Given these probabilities and α and β parameters, the thresholds $\theta_A^{(t)}$ and $\theta_B^{(t)}$ are estimated as in Chapter 5. However, since a feature-switch in selector $\mathcal{L}_s$ causes a wrong estimate of the statistics of subsequent selectors, the statistics are reseted for the selectors $\mathcal{L}_t, t \geq s$, i.e. $p(f_t(\mathbf{x})|y = C, \rightarrow t)$ is set to the uniform distribution and $p(\rightarrow t|C) = 0.5$ for $C \in \{-1, +1\}$.

This training scheme allows for classifier speedup in both training and evaluation compared to the original on-line boosting. Moreover, the number of selectors can be set to a high number and the real classifier complexity (i.e. number of weak classifiers used) is controlled automatically.

8.3 Experiments

The properties of the proposed on-line WaldBoost algorithm are demonstrated on the task of visual object tracking. It is formulated as a binary classification problem where the classifier is used to distinguish the object from the local background [27]. To allow adaptability to the object appearance changes, updates of the classifier are performed – here we replaced the on-line boosting algorithm with the on-line WaldBoost algorithm. For the on-line WaldBoost classifier $T = 50$ selectors were used which selects from a global feature pool of $L = 250$ weak classifier corresponding to Haar-like features (same parameters as in [27]). The Wald parameters were set to $\alpha = 0.02$ and $\beta = 0$.

Figure 8.2 shows a challenging tracking sequence including appearance changes of the object as well as object occlusions on a complex background. The second row depicts the confidence maps of the classifier. Since we use no motion model (cf. [87]), a confidence map is computed by evaluating the classifier at all positions within a local search region. The position of the object

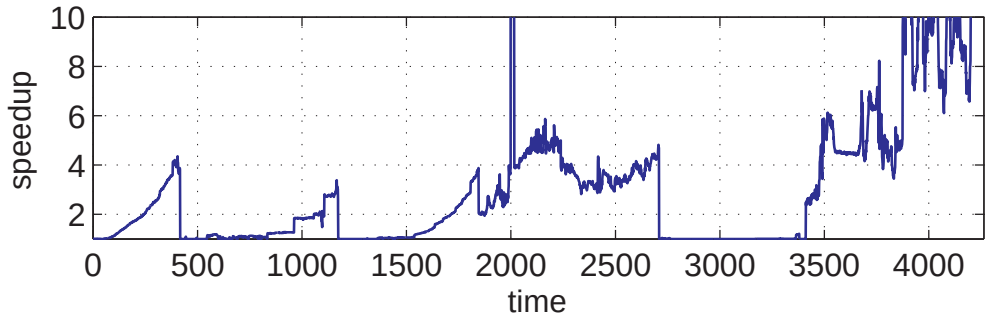


Figure 8.3: Speed-up compared to the non-sequential on-line boosting approach [27].

corresponds to the maximum in the confidence map. The values equal to zero show the positions rejected before reaching the end of the classifier sequence.

These early decisions lead to the speed-up shown in Figure 8.3. The speed-up is calculated as $T/\tilde{T}$, where $\tilde{T}$ is the average number of weak classifiers used before the decision is reached over the whole search region. If all values are equal to zero the object is considered to be lost. If the object is "stable" in the scene, the speed-up is continuously increasing since background patches can be discarded early. On average, we achieved a speed-up of a factor of 5 to 10 without suffering a loss in tracking quality, i.e. we never discard the maximum peak of the confidence map, so the tracking results are exactly the same as reported in [27]. The same tracking results are reached also for the other tracking sequences used in [27, 46]. However, the speed-up is not that big, since the object and the background changes a lot. Nevertheless, in the training the complexity of the classifier can still be determined automatically.

In general, the achieved speed-up depends on dynamically changing problem difficulty and how often the SPRT statistics have to be reseted (e.g. at frame 2706). Further, higher values of α lead to more speedup but having the risk of loosing the object if it changes its appearance too fast. The achieved speed-up (≥ 1) can be used for instance for extending the search region to handle faster movements or to include more degrees of freedom like scale.

8.4 Summary

In this chapter we have shown how to extend the WaldBoost algorithm to on-line learning settings. The proposed on-line WaldBoost training algorithm is able to control the classifier complexity depending on the problem difficulty. Moreover, the evaluation speed is increased through the sequential nature of the classifier compared to a non-sequential on-line classifier. We tested the on-line WaldBoost algorithm on a visual tracking problem. A significant speed-up was reached compared to the non-sequential on-line boosting. We are confident that other applications (e.g. improving object detectors) can benefit as well. In the future work we want to improve the statistics reseting strategy as well as to adapt the α parameter on-line.

9 Conclusions

In this thesis, the problem of learning sequential two-class classifiers with decision quality and evaluation time trade-off was studied. In contrast to commonly used learning algorithms, the time-to-decision parameter is explicitly considered in the task formulation. In the proposed learning formulation a sequential classifier which minimises the evaluation time subject to user defined upper bounds on the error rates is searched for.

As a solution to the formulated problem we proposed a learning algorithm called WaldBoost. The WaldBoost learning algorithm builds on the sequential probability ratio test (SPRT). We showed how to enlarge SPRT's capabilities to problems with dependent measurements and how the limitations of SPRT to a priori ordered measurements and known joint probability density functions can be overcome by using the AdaBoost algorithm. The AdaBoost algorithm selects and orders relevant measurements for SPRT and its properties allow simple decision thresholds estimation.

The WaldBoost algorithm was tested on the face detection problem. On a standard dataset, the results are superior to the state of the art methods in average evaluation time and comparable in detection rates. In the face detection context, the WaldBoost algorithm can be also viewed as a theoretically justifiable "boosted cascade of classifiers" proposed by Viola and Jones [81].

Another application of the WaldBoost algorithm described in the thesis is a framework for speeding up existing binary decision processes by learning their sequential WaldBoost emulators. Two interest point detectors, the Hessian-Laplace and the Kadir-Brady saliency detector, served as examples of emulated algorithms. The experiments show similar repeatability and matching scores of the original and emulating algorithms. For both, the Hessian-Laplace and the Kadir-Brady detectors, the WaldBoost emulation improved significantly the speed. The emulator was nine times faster for the Hessian-Laplace detector and about 142 times faster for the Kadir-Brady detector. In the case of the Kadir-Brady detector this speed-up opens new possibilities for using the detector in time sensitive applications. For the Hessian-Laplace detector, the achieved speed is similar to SURF, a commonly used Hessian-like fast detector; the WaldBoost emulator approximates the output of the Hessian-Laplace detector more precisely.

The proposed emulation approach is general and can be applied to other algorithms as well. For future research, an interesting extension of the methodology would be to train an emulator which not only guarantees output similar to an existing algorithm but which also possesses some additional quality like insensitivity to certain acquisition conditions (e.g. motion blur) or maximum performance in a particular environment or task.

Finally, we have shown how to extend the on-line boosting algorithm using Wald's sequential decision theory. The proposed on-line WaldBoost learning algorithm is able to control the classifier complexity depending on the problem difficulty. Moreover, the evaluation speed is

increased through the sequential nature of the classifier. We tested the on-line WaldBoost algorithm on a visual tracking problem. The evaluation speed is significantly improved compared to the non-sequential on-line boosting while preserving the tracking capabilities. We are confident that other applications (e.g. improving object detectors) can benefit from using the on-line WaldBoost algorithm as well.

Bibliography

[1] http://en.wikipedia.org/wiki/Twenty_Questions, Jan 2009.

[2] S. Baker and S. K. Nayar. Algorithms for pattern rejection. In *ICPR*, page 869, 1996.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *CVIU*, 110(3):346–359, 2008.

[4] B. Bhanu. Automatic target recognition: State of the art survey. *IEEE Transactions on Aerospace and Electronic Systems*, 22:364–379, 1986.

[5] Gilles Blanchard and Donald Geman. Hierarchical testing designs for pattern recognition. *Annals of Statistics*, 33(3):1155–1202, 2005.

[6] Mark Boddy and Thomas L. Dean. Decision-theoretic deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67(2):245–285, 1994.

[7] Lubomir Bourdev and Jonathan Brandt. Robust object detection via soft cascade. In *CVPR*, pages 236–243, 2005.

[8] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[9] S. Charles Brubaker, Jianxin Wu, Jie Sun, Matthew D. Mullin, and James M. Rehg. On the design of cascades of boosted ensembles for face detection. Technical report, Georgia Institute of Technology, 2005.

[10] S. Charles Brubaker, Jianxin Wu, Jie Sun, Matthew D. Mullin, and James M. Rehg. On the design of cascades of boosted ensembles for face detection. *International Journal on Computer Vision*, 77(1-3):65–86, 2008.

[11] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893, 2005.

[12] Thomas Dean and Mark Boddy. An analysis of time dependent planning. In *AAAI*, pages 49–54, 1988.

[13] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006.

[14] S. Esmeir and S. Markovitch. Anytime induction of low-cost, low-error classifiers: a sampling-based approach. *Journal of Artificial Intelligence Research*, 33:1–31, 2008.

[15] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *CVPR*, volume 1, pages 380–387, 2005.

[16] Franois Fleuret and Donald Geman. Coarse-to-fine face detection. *International Journal on Computer Vision*, 41:85–107, 2001.

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

[18] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[19] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, pages 121(2):256–285, 1995.

[20] Yoav Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3):293–318, 2001.

[21] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[22] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Department of Statistics, Sequoia Hall, Stanford Univerity, July 1998.

[23] B. Froba and A. Ernst. Face detection with the modified census transform. In *AFGR*, pages 91–96, 2004.

[24] B. Froba and C. Kublbeck. Robust face detection at video frame rate based on edge orientation features. In *AFGR*, pages 327–332, 2002.

[25] H. Grabner and H. Bischof. On-line boosting and vision. In *CVPR*, volume 1, pages 260–267, 2006.

[26] H. Grabner, Beleznai C., and H. Bischof. Improving adaboost detection rate by wobble and mean shift. In *Computer Vision Winter Workshop*, pages 23–32, 2005.

[27] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, volume 1, pages 47–56, 2006.

[28] H. Grabner, C. Leistner, and H. Bischof. Time dependent on-line boosting for robust background modeling. In *VISAPP*, 2007.

[29] H. Grabner, P.M. Roth, and H. Bischof. Is pedestrian detection realy a hard task? In *PETS*, 2007.

[30] H. Grabner, J. Šochman, H. Bischof, and J. Matas. Training sequential on-line boosting classifier for visual tracking. In *ICPR*, 2008.

[31] M. Grabner, H. Grabner, and H. Bischof. Fast approximated SIFT. In *ACCV*, pages I:918–927, 2006.

[32] Eienne Grossmann. Automatic design of cascaded classifiers. *Lecture notes in computer science*, 3138:983–991, 2004.

[33] J. S. Hare and P. H. Lewis. Salient regions for query by image content. In *CIVR*, pages 317–325, 2004.

[34] Chang Huang, Haizhou Ai, Shihong Lao, and Yuan Li. High-performance rotation invariant multiview face detection. *PAMI*, 29(4):671–686, 2007.

[35] Chang Huang, Haizhou Ai, Yuan Li, and Shihong Lao. Vector boosting for rotation invariant multi-view face detection. In *ICCV*, pages 446–453, 2005.

[36] M. Jones and P. Viola. Fast multi-view face detection. Technical report, Mitsubishi Eletric Research Laboratories, August 2003.

[37] Michael J. Jones and James M. Rehg. Statistical color models with application to skin detection. *International Journal on Computer Vision*, 46(1):81–96, 2002.

[38] T. Kadir and M. Brady. Saliency, scale and image description. *International Journal on Computer Vision*, 45(2):83–105, 2001.

[39] Zdeněk Kálal, Jiří Matas, and Krystian Mikolajczyk. Weighted sampling for large-scale boosting. In *BMVC*, 2008.

[40] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: object localization by efficient subwindow search. In *CVPR*, 2008.

[41] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *CVPR*, volume II, pages 775–781, 2005.

[42] S.Z. Li and Z. Zhang. FloatBoost learning and statistical face detection. *PAMI*, 26(9):1112–1123, 2004.

[43] S.Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *ECCV*, volume IV, pages 67–81, 2002.

[44] Rainer Lienhart, Alexander Kuranov, and Pisarevsky Vadim. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM*, 2003.

[45] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *ICIP*, 2002.

[46] J. Lim, D. Ross, R. Lin, and M. Yang. Incremental learning for visual tracking. In *NIPS*, volume 17, pages 793–800. MIT Press, 2005.

[47] Huitao Luo. Optimization design of cascaded classifiers. In *CVPR*, volume 1, pages 480–485, 2005.

[48] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, 2004.

[49] Jiří Matas and Ondřej Chum. Randomized RANSAC with sequential probability ratio test. In *ICCV*, volume II, pages 1727–1732, 2005.

[50] Jiří Matas and Jan Šochman. Wald's sequential analysis for time-constrained vision problems. In *ICRA*, April 2007.

[51] K. Mikolajczyk. http://www.robots.ox.ac.uk/˜vgg/research/affine, 2008.

[52] K. Mikolajczyk. Personal communication, 2008.

[53] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal on Computer Vision*, 65(1-2):43–72, 2005.

[54] Krystian Mikolajczyk. *Detection of local features invariant to affines transformations*. PhD thesis, INPG, Grenoble, July 2002.

[55] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *International Journal on Computer Vision*, 60(1):63–86, 2004.

[56] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 24(7):971–987, 2002.

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

[58] Ofir Pele and Michael Werman. Robust real time pattern matching using bayesian sequential hypothesis testing. *PAMI*, 30(8):1427–1443, 2008.

[59] Matthias Raetsch, Sami Romdhani, and Thomas Vetter. Efficient face detection by a cascaded support vector machine using haar-like features. In *DAGM*, 2004.

[60] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. In *Machine Learning*, volume 3, pages 287–320, 2001.

[61] Ronald L. Rivest. Learning decision lists. In *Machine Learning*, pages 229–246, 1987.

[62] S. Romdhani, P.H.S. Torr, B. Scholkopf, and A. Blake. Computationally efficient face detection. In *ICCV*, pages II: 695–700, 2001.

[63] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *ECCV*, volume 1, pages 430–443, 2006.

[64] H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *PAMI*, 20(1):23–38, 1998.

[65] H.A. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *CVPR*, page 963, 1998.

[66] Hichem Sahbi. *Coarse-to-fine support vector machines for hierarchical face detection*. PhD thesis, Computer Science, University of Versailles at France, April 2003.

[67] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

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

[69] Michail I. Schlesinger. Bystraja realizaciaja odnovo klassa linejnykh svjortok. *Teoret. i prikl. voprosy raspoznavanija izobrazhenij*, 1991.

[70] H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection. In *CVPR*, volume 2, pages 29–36, 2004.

[71] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

[72] David Siegmund. *Sequential Analysis. Test and Confidence Intervals*. Springer Series in Statistics. Springer-Verlag, New York, NY, 1985.

[73] Jan Šochman and Jiří Matas. WaldBoost - learning for time constrained sequential detection. In *CVPR*, volume 2, pages 150–157, 2005.

[74] Jan Šochman and Jiří Matas. Learning a fast emulator of a binary decision process. In *ACCV*, 2007.

[75] Jan Šochman and Jiří Matas. Learning fast emulators of binary decision processes. *Submitted to International Journal on Computer Vision*, 2009.

[76] J. Sun, J.M. Rehg, and A. Bobick. Automatic cascade training with perturbation bias. In *CVPR*, volume 2, pages 276–283, 2004.

[77] Kah Kay Sung and Tomaso Poggio. Learning human face detection in cluttered scenes. In *Computer Analysis of Images and Patterns*, pages 432–439, 1995.

[78] Kah Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. *PAMI*, 20(1):39–51, 1998.

[79] N. A. Thompson. A practitioners guide for variable-length computerized classification testing. *Practical Assessment Research & Evaluation*, 12(1), 2007.

[80] K. Toyama. Handling tradeoffs between precision and robustness with incremental focus of attention for visual tracking. In *AAAI Symposium on Flexible Computation in Intelligent Systems*, 1996.

[81] P. Viola and M.J. Jones. Robust real time object detection. In *SCTV*, 2001.

[82] Paul Viola and Michael J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.

[83] Abraham Wald. *Sequential analysis*. Dover, New York, 1947.

[84] B. Wu and R. Nevatia. Improving part based object detection by unsupervised, online boosting. In *CVPR*, pages 1–8, 2007.

[85] Bo Wu, Haizhou AI, Chang Huang, and Shihong Lao. Fast rotation invariant multi-view face detection based on real adaboost. In *FGR*, 2004.

[86] R. Xiao, L. Zhu, and H.J. Zhang. Boosting chain learning for object detection. In *ICCV*, pages 709–715, 2003.

[87] L. Xj, T. Yamashita, S. Lao, M. Kawade, and F. Q. Online real boosting for object tracking under severel appearance changes and occlusion. In *International Conference on Acoustics, Speech and Signal Processing*, pages 925–928, 2007.

[88] Long Zhu, Yuanhao Chen, and Alan L. Yuille. Unsupervised learning of a probabilistic grammar for object detection and parsing. In *NIPS*, pages 1617–1624, 2006.