Graz University of Technology

Institute for Computer Graphics and Vision

PhD Thesis

On-line Boosting and Vision

Helmut Grabner

Graz, August 18, 2008

1. Examiner and Advisor
Prof. Dr. Horst Bischof

2. Examiner
Prof. Dr. Aleš Leonardis

The aim of computer vision is to overfit to the real world.

Antonio Torralba (after his third beer).

Affirmation

Herewith I declare that I created this work by myself. I used no other than the listed references and marked the locations, where I assumed contently or literally same content from these references.

Acknowledgments

This PhD thesis was created 2004-2008 at the Institute for Computer Graphics and Vision at the Graz University of Technology, Austria.

The financial supported during that period was given in the first two years by the the project VITUS-II by the Austrian Federal Ministry of Transport, Innovation and Technology under P-Nr. I2-2-26p. The last two years, I was employed as a research and teaching assistant and hence payed from the University.

Sincere thanks are given to all members of the Institute for Computer Graphics and Vision and especially to my supervisor Horst Bischof, who guided me through these years of research and was always there with useful advices and suggestions. Thanks to Jiří Matas from Czech Technical University, Prague. I was visiting the lab twice for a few weeks each and benefit much from discussions with him and the members of his group, especially with Jan Šochman. Thanks to Peter Auer from the University of Leoben for good collaboration and useful advices form the field of machine learning. Also thanks to Aleš Leonardis from University of Ljubljana, who agreed to be the second examiner of this thesis and gave useful remarks.

My load of work was a lot reduced because of some great researchers where most of this work was jointly created, especially Michael Grabner, Peter M. Roth, and Christian Leistner. Further, I will thank all the members of the ICG for fruitful and critic discussions about my work.

I also want to thank my friends who often helped me to distract my mind and to get a clear head and new ideas for the next day of research.

Thanks to my family who always supported the way I chose. Finally, I want to give especial thanks to my girlfriend, Susanne, who supported me through all the ups and downs during the work on this thesis.

Graz, August 18, 2008 Helmut Grabner

Abstract

In this thesis, we introduce the *On-line Boosting for Feature Selection* algorithm. Boosting, a widely used machine learning algorithm, has become very popular in computer vision, showing impressive performance for detection and recognition tasks. Mainly off-line training methods have been used, which implies that all training data has to be given a priori. To train the classifier on-line and incrementally as new data becomes available has several advantages and opens new areas of applications in computer vision. However, in many practical applications only partially labeled or unlabeled data is available. Since the proposed algorithm is fully supervised, labels have to be generated for the unlabeled samples. Thus, different methods, from fully supervised learning to self-learning, are shown. In fact, we apply the algorithm on such diverse tasks as learning complex background models, visual tracking, and improving object detectors over time. All approaches benefit significantly from the on-line training, which is shown by various experiments. Extensive evaluations have been done, whereas results demonstrate the applicability of the proposed algorithm for all these different applications. Nevertheless, it cannot be ensured that always correct updates are made and hence the system may drift, *i.e.*, starts learning something wrong. Finally, some suggestions of limiting or avoiding the problem are discussed.

Keywords: computer vision, machine learning, on-line learning, boosting, feature selection, object detection, object tracking, background modeling, drifting, supervision.

Kurzfassung

In dieser Arbeit wird der Algorithmus *Online Boosting für Merkmalsselektion* vorgestellt. Boosting, ein viel verwendeter Algorithmus aus dem maschinellen Lernen, hat sich auch im Bereich des maschinellen Sehens (Computer Vision) etabliert. Eindrucksvolle Resultate werden sowohl bei der Objektdetektion als auch bei der Objekterkennung erzielt. Hauptsächlich wurden sogenannte offline Lernmethoden verwendet, die voraussetzen, dass alle Trainingsdaten bereits zu Beginn vorhanden sind. Wird der Klassifikator jedoch online oder inkrementell trainiert, wenn neue Daten vorhanden sind, ergeben sich viele Vorteile und neue Anwendungsmöglichkeiten. Bei vielen praktischen Anwendungen ist jedoch die Klassenzugehörigkeit zu den Daten nur teilweise oder gar nicht bekannt. Da der vorgestellte Algorithmus aber immer auch die Klassenzugehörigkeit benötigt, muss diese für die fehlenden Datenpunkte generiert werden. In dieser Arbeit werden verschiedene Methoden, von vollständig überwachtem Lernen bis hin zum eigenständigen Lernen, untersucht. Im speziellen wird der Algorithmus auf sehr unterschiedliche Bereiche des maschinellen Sehens angewendet, wie des Lernen von komplexen Hintergrundmodellen, das Verfolgen von Objekten (Tracking) und die kontinuierliche Verbesserung von Objektdetektoren. Alle Verfahren profitieren signifikant vom online Lernen was anhand etlicher Experimente veranschaulicht wird. Die Eignung des vorgestellten Algorithmus für all diese Anwendungen wird durch umfangreiche Auswertungen demonstriert. Da jedoch nicht immer gewährleistet werden kann, dass korrekte Daten vorhanden sind, neigt das System zu driften, d.h. etwas Falsches zu lernen. Zum Abschluss werden einige Vorschläge für die Limitierung bzw. das Verhindern dieses Problems diskutiert.

Stichworte: maschinelles Sehen, maschinelles Lernen, online Lernen, Boosting, Merkmalsselektion, Objektedetektion, Objektverfolgung, Hintergrundmodellierung, Drift, Supervision.

Contents

Chapter 1

Introduction

The chapter sets the frame which motivates this thesis. We ask ourself the following questions: Why is it worth doing research on computer vision? What are the historical roots of computer vision? What are the long term aims? Why is machine learning highly used in the last years? What are the main issues of learning nowadays? How is this work related to it? What did we achieve over the last years? And thus, what are the contributions of this thesis.

1.1 Motivation

An image contains a lot of informations encoded in thousands of pixels. Each of them may have highly relevant informations of what is happening in this image. But, how can these informations be extracted? David Marr started his classical book *Vision* [80] with the following question:

> *What does it mean, to see? – The plain man's answer (and Aristotle's, too) would be, to know what is where by looking.*

(a) typical holiday picture (b) art[2]

Figure 1.1: What does it mean to see?

By taking a look at Figure 1.1, what does the human care about in an image and thus what are the tasks for computer vision? Following Pietro Perona[1] the following points should be considered: *Verification* – Is an object present in a given sub-image?; *Classification* – What object (apple, car, person,...) is it?; *Detection* – Are there people?; *Naming* – What is where?; *Identification* – Is this me?; and *Categorization* – Streets, Trees,... An other line of research focus on *Scene Understanding* – What is the picture telling me? [21]

All the considerations above work on a still images. When using a video or image sequence, motion plays an essential role and the tasks get extended using the dynamic behavior, *e.g.*, *Dynamic Classification* – How does the object move? (*e.g.*, jumping, running,...). Besides this, a new question comes up and is formulated by the following task: *Tracking* – Where is the object in the next frame?

The temporal information can also be helpful in order to detect or categorize object (*e.g.*, focus of attention [61]). In general, high level information or context can help to verify decisions and it can be used to reduce the search space and speed up recognition and detection *e.g.*, [50, 124]. However, the temporal behavior of an object can sometimes be obtained by a single image [115], *e.g.*, jumping of a person. This is possible since we have a lot of experiences and know how an object should behave in a scene (*e.g.*, taking into account all the physical rules such as gravity or causality). For example, 3D structure can be obtained from a single image as well [51].

The term *object* is used quite often above, however, it is not defined up to now. By

[1]Slides are on-line available http://www.mis.informatik.tu-darmstadt.de/events/iwoc-iccv07/, (April 29, 2008)

[2]From *Best Pictures On The Internet 2007 Awards*, http://refreshyourself.wordpress.com/2008/03/26/best-pictures-on-the-internet-2007-awards/ (August 18, 2008)

Figure 1.2: Inter-class and inner-class variability: The subfigures shows Jaroslaw Kaczyński (prime minster of Poland 2006-2007) and his identical twin brother Lech Kaczyński, (currently President of Poland) at different ages[5].

taking a look into a dictionary[3] one gets:

> *Something perceptible by one or more of the senses, especially by vision or touch; a material thing.*

Hence, in the standard understanding an object is a (rigid) body with six degrees of freedom. The appearance can vary in a wide range[4]. Thus, challenges are invariance against view point variations, scale, illumination, deformations of the object as well as robustness to background cutter and occlusions. Furthermore, one should recognize that we live in a 3D world and that there are problems with local ambiguities.

Regarding object categorization, where it is considered that $10,000$ to $30,000$ object categories are present, one has to deal with high interclass variance and low intraclass variance, *e.g.*, the class of bikes and motorbikes [99]. Furthermore, there is no dichotomy into (specific object) recognition and categorization[5], but a continuum of problems. Nowadays, algorithms address one or the other, sometimes using quite different techniques. An example is shown in Figure 1.2, where photos of the same twins are depicted at different ages. Both are persons, look similar but are individual people. Unless all these issues, a further problem arises when connecting words to objects. Sometimes one word corresponds to more than one object. Hence, if this is the case it is necessary to also take a look at the context and semantic. An example is shown in Figure 1.3.

However, up to now there is no closed theory of visual recognition and no common used definition of "an object" in the vision community. Concerning the standard

[3] `http://www.thefreedictionary.com/object`, (April 30, 2008)

[4] The following considerations are based on an excellent tutorial entitled *Recognizing and Learning Object Categories* by Li Fei-Fei, Rob Fergus and Antonio Torralba given at ICCV 2005 and CVPR 2007. Slides and further material are on-line available at `http://people.csail.mit.edu/torralba/shortCourseRLOC/`, (April 28, 2008)

[5] The considerations here are triggered by discussions with Jiří Matas after the tutorial he organized jointly with Krystian Mikolajczyk at ICCV 2007 on *Visual Recognition*. Pictures from Figure 1.2 are thankfully provided by them.

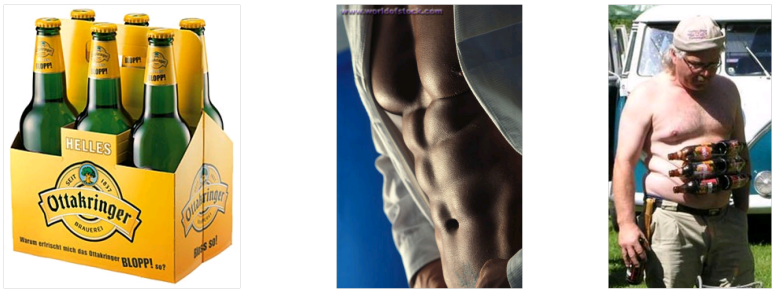


Figure 1.3: "Six pack" (Images from Google Image Search).

understanding from the dictionary, are clouds in the sky objects? To overcome this, the following attempt is given by Jiří Matas[5]:

> *View Visual Recognition as the process of associating parts of images with parts of (other) images. The association may occur at very different levels of abstraction, like the two image parts are shifted versions of each other (*e.g.*, stereo), the observation in two image parts are consistent with 3D rigid body motion, responses of some filters have similar statistics (*e.g.*, in texture recognition), or the configuration of some general characteristics is similar (*e.g.*, in categorization). They can be established through supervision (same label) or for example on basis of some non-visual experience (*e.g.*, touch, taste). An object is any representation derived from associated image parts,* i.e.*, an "object" may be a scene, an image, a rigid object, a class, a behavior* etc.

There might be no good theory of visual recognition, but at least the goals should be clear. The approaches should take the following points into account:

Generality: Recognize a "broad range" of objects, *i.e.*, be general in the space of possible objects. This is maybe the critical weakness of current recognition methods.

Robustness: The recognition should be robust to occlusion of the object and should allow to recognize the object under different lightning conditions, in cluttered background scenes and from different views.

Efficiency: The response time should be insensitive to the number of objects to be recognized, *i.e.*, scales sub-linear with respect to the number of objects [125] and categories.

Constructive: The method should be able of learning new objects via observation (possibly active or unsupervised) and include it them the current models.

1.2 Object Recognition System

There are many requirements as listed above. However, it is unclear how to model the objects and categories. It seems that we are trying to solve problems, that do not have a solution (vision is an ill-posed problem). Or the other way around, we are nowadays not able to formulate the task in a popper way to solve it. Thus, we *learn*, what distinguishes them rather than manually specify the difference. This is done using data, statistics, and machine learning algorithms. Alyosha Efros[6] called this playful:

> *Google Intelligence – The Artificial Intelligence for the Post-modern World.*

In other words, what happened before is likely to happen again. In general, a recognition system involves the following three main points.

Representation: An object can be represented in many different ways. It can be done global, local, part-based or hierarchical. Which again can be described using only their appearance or both, location and appearance. To model location, this can be done explicitly via probability density functions or implicitly, *i.e.*, by using a voting space [65].

As basis edges, patches, filter responses or interest points are proposed as features. Descriptors for these features should be invariant against some of the above mentioned issues but at the same time powerful enough to allow of distinguishing the individual parts[7]. This is realized by appearance based representation (*e.g.*, SIFT [79]), subspace methods [128] or classifier [130].

Learning: Given a set of training data, the task of learning is to form a model, which is able to classify these data (*e.g.*, in object vs. background or in individual categories). Many different learning methods were proposed, *e.g.*, *Principle Component Analysis* (PCA) [128], *Boosting* [130], *Support Vector Machines* [88], *Bayesian Networks* [117], or *Neural Networks* [109].

When using learning, the knowledge is encoded by the given training samples. Thus, we have to add a further requirement (goal) for any object recognition system to the listed points mentioned in the last section.

Training data: Minimize the number of training images needed to build a model. Concerning robustness, if possible this can be achieved via geometric and photometric invariance. However, it seems impossible to learn all possible backgrounds and occlusions. Hence, this should be done by

[6] Invited Talk at the 3D Representation for Recognition Workshop held in conjunction with ICCV 2007.

[7] Consider the letters `W` and `M`, if the representation is rotation invariant we are not able do distinguish between them anymore.

the representation. Concerning model construction, it should be possible to learn new objects only from few examples (one-shot learning [22]) and in an incremental manner [70].

Note, everything which is not handled by the (invariant) representation has to be learned. However, the representation has to be powerful enough to allow the learning algorithm to solve the task. The issues, which have to be considered in this step are: the level of supervision (fully labeled, weakly labeled, unlabeled), label errors (wrong labels), label jitter (not well aligned objects), learning methods (discriminative vs. generative), one/multiple class problems, including prior knowledge and context, incremental learning (complete re-training is prohibitive), active learning, learning from few (one) examples, *etc.*

Recognition: Once a model (a generative model or a discriminate classifier) is trained, it can be applied on novel data. How this is done is (i) highly task depended and (ii) depends on the learned model. For example, the task can vary from just detecting the presents of an object up to accurate localize it. The localization can either be by a bounding box or by a pixel level segmentation. Further, global or a local object models may used, which in general needs different techniques for invariant (*e.g.*, rotation, scaling) and robust (*e.g.*, occlusions) recognition.

The increase of computational power allows more powerful machine learning techniques to be used and are quite common nowadays. Besides novel methods for local image representations, there was a significant progress in using advanced machine learning methods. Further, if enough labeled training data exists these approaches can obtain very high recognition performances. For example, in object categorization (see [99] for a good overview), in the recent years, there was a significant progress on methods for visual categorization. For example, the performance on the Caltech 101 dataset was in 2004 approximately 16%, now the best performing approaches obtain close to 70% [27]. In most of these approaches machine learning plays an important role. However, one should care about the human visual system [118] or even learn from the observations and experiences from cognitive scientist, *e.g.*, [17, 98].

1.2.1 Learning and Vision: A Historical View[8]

In this section, we take a brief historical look back in time and partially review the connections between computer vision and machine learning. Early papers on image

[8]This section does not claim to be completive. The goal is to give the reader a rough feeling of how the community evolves and how learning techniques are used. `http://www.icg.tugraz.at/News/historyOfCV`, (May 11, 2008), [29].

analysis appeared in general electrical engineering conferences. By 1960's journals and conferences get established [104] (see Table 1.1).

Year	Event
since 1960	Digital image processing by computer
1968	Journal on Pattern Recognition, Pergamon, now Elsevier.
1969	First Textbook: *Picture Processing by Computer* by A. Rosenfeld.
1970	Journal on Artificial Intelligence
1972	IEEE International Conference on Pattern Recognition (ICPR), Washington, D.C.
1977	Conference on Pattern Recognition and Image Processing (PRIP), Troy, New York.
1978	Textbook: *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information* by David Marr, posthumous published in 1982 [80].
1979	IEEE Journal of Pattern Recognition and Machine Intelligence (PAMI). See [12] for a nice reflection.
1980	The National Conference an Artificial Intelligence (AAAI), Stanford.
1983	IEEE Conference in computer vision and Pattern Recognition (CVPR) , Arlington, Virginia (PRIP was the precursors to it).
1987	Neural Information Processing System Conference (NIPS), Denver.
1987	IEEE International Conference of Computer Vision (ICCV), London.
1987	International Journal of Computer Vision (IJCV), Springer.
1993	10th International Conference on Machine Learning (ICML). Former known as International Workshop on Machine Learning (ML).

Table 1.1: Historical events in vision and graphics.

At the beginning less than 100 people where submitting papers and participated to those conferences. Nowadays, the *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* and the *IEEE International Conference on Computer Vision (ICCV)* together with the *European Conference of Computer Vision (ECCV)* are currently the three major conferences in the field of computer vision, each with a few hundred to more than 1000 participants. The submissions (and the number of participants) are still increasing while keeping low acceptance rate (around 20%). An overview of the submission statistis[9] is visualized in Figure 1.4.

[9]The data was collected using the following links: http://iris.usc.edu/Information/ieee/history.html, http://lear.inrialpes.fr/people/triggs/events/iccv03/conf-stats, http://vrlab.epfl.ch/~ulicny/statistics/, http://www.adaptivebox.net/research/bookmark/CICON_stat.html and http://tab.computer.org/pamitc/conference/history.html, (February 20, 2008). The links also provide information of other conferences and journals (including prominent machine learning ones), *e.g.*, ICPR, NIPS, ICML, PAMI.

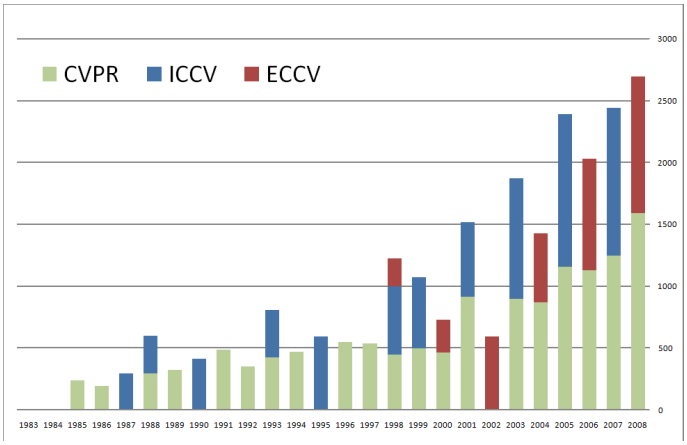


Figure 1.4: Number of submissions to the three major computer vision conferences.

One of the godfathers of computer vision, Azriel Rosenfeld, started in the year 1969 reviewing and classifying the literature related to computer vision and image analysis. From that on he published every year up to the year 2000 an article[10]. Even the naming of the journals and titles are interesting showing how the terms changes (see Table 1.2). *Image Processing* takes as input as well as output an image whereas in *Computer Vision* (or *Image Analysis*) the output is a descriptive data [104][11].

1.2.2 Two Selected Examples

Even in the very beginning of computer vision machine learning methods where applied. Nowadays, computer vision and machine learning are two fields, which are very closely related to each other. This is reflected by the publications appearing in the listed conferences and journals. Besides many excellent papers, from my point of view, in the last decade there are two outstanding papers which have an enormous impact for visual perception. We briefly mention them in the context of how they represent, "learn", and finally recognize objects on novel images.

Object Recognition: Based on the work of Schmid and Mohr [116], David Lowe developed SIFT (Scale Invariant Feature Transform) for (specific) object recognition. It is widely used by many researchers, hence the original conference paper (ICCV 1999) [79] has about 1,030 cites and the IJCV version [78] which appeared in 2004 has 2,070 cites[12].

[10] Nowadays, web based databases exist, *e.g.*, `http://www.visionbib.com/bibliography/` or `http://www.informatik.uni-trier.de/~ley/db/conf/`, (May 2, 2008).

[11] If the input is descriptive data and the output is an image this is called *Computer Graphics* or *Image Synthesis*.

[12] All cites on Google scholar, (May 11, 2008).

Years	Title	Appears in
1969	Picture Processing by Computer: Survey	ACM Computing Surveys
1973	*Progress* in Picture Processing: 1969-71	ACM Computing Surveys
1972-1981	Picture Processing: 19xx	Journal of Computer Graphics and Image processing
1982-1986	Picture Processing: 19xx	Journal of *Computer Vision*, Graphics and Image Processing
1987-1993	Image Analysis and *Computer Vision*: 19xx	Journal of Computer Vision, Graphics and Image Processing
1994-1999	Image Analysis and Computer Vision: 19xx	Journal of Computer Vision and *Image Understanding*
2000	Classifying the Literature Related to Computer Vision and Image Analysis	Journal of Computer Vision and Image Understanding

Table 1.2: Reviewing and classifying the literature related to computer vision by A. Rosenfeld. The evolution is even reflected by the words used, beginning from *Picture Processing* to *Image Understanding*.

Objects are described by local features. For that purpose, interest points are extracted from the image, which are then independently described using a fixed mapping (feature vector). The feature vector is saved and associated to objects. Hence, the learning part is reduced to a simple nearest neighbor matching. The power of the approach lies in the representation, *i.e.*, the description of the object's keypoints and the voting schema.

Fast Object Detection: The seminal work of Paul Viola and Michael Jones [130] published in Proceedings of CVPR in the year 2001 and had great success from then on. This paper has about 1,700 cites and the IJCV [132] version, which appeared 2002 has about 925 cites[12], which of course reflects the enormous impact[13].

The main idea of the Viola and Jones face detector is to build a discriminative classifier witch can distinguishes very efficiently between image patches containing a faces or not. This is achieved by combining simple image features, which were selected via *AdaBoost*, a prominent machine learning algorithm[14].

[13]It seems since both of these paper were published in IJCV currently this journal has the highest impact factor of 6.085 in our community followed by PAMI with 4.306. Thomans Journal Citation Report, http://admin-apps.isiknowledge.com/JCR/JCR?RQ=HOME, (May 2, 2008).

[14]*AdaBoost* was proposed by Freund and Schapire [24] in 1997; the paper has about 2,650 cites[12].

Thus highly machine learning is used in their approach. Detection is done by applying the classifier subsequently to the whole image using a sliding windows. This exhaustively search is only feasible, since the image features can be calculated very efficiently and the evaluation is cascaded in order to quickly decide to the background class. Since usually overlapping detections are achieved, they have to be combined in a post processing step, *e.g.*, via a simple *Non-Maxima-Suppression*.

To sum up, machine learning techniques are used extensively in many computer vision applications by researchers for more than the last 30 years. Nowadays, both learning from few examples, and learning from a huge database with partially labeled or unladed data are hot topics. In order to efficiently cope with these problems, incremental and on-line learning methods plays an important role.

1.3 Problem Statement

A lot of research is focused on developing on-line learning methods. A subset of cites why it is worth doing on-line or incremental learning are given in the following:

- *There are a number of interesting situations where learning must take place over time, in a kind of continuous fashion rather than as an one-shot experience.* [28]
- *In particular, incrementally learning a model, which is computationally efficient for large-scale problems as well as adaptable to reflect the variable state of a dynamic system, is an attractive research topic with numerous applications such as adaptive background modeling and active object recognition.* [72]
- *Learning representations of objects and scenes is an essential part of any cognitive vision system. In the real world, learning is usually a continuous, never ending process, thus requiring incremental methods for updating previously learnt representations.* [119]
- *To learn concepts over massive data streams, it is essential to design inference and learning methods that operate in real time with limited memory. ... Compared to batch methods, online learning methods are often simpler to implement, faster, and require less memory. For such reasons, these techniques are natural ones to consider for large-scale learning problems.* [14]

As showed in the previous paragraph, there is an essential need for on-line algorithms, that are able to learn continuously. For this thesis, the following task should be considered:

> *It should be investigated how computer vision applications can benefit from* **on-line** *machine learning. In particular, a general algorithm, which is applicable for many computer vision problems should be developed.*

Due to its success for computer vision tasks, our approach is heavily inspired by the seminal work of Viola and Jones [130] mentioned earlier. In a nutshell: The assumption is that a small subset of simple image features is sufficient to distinguish between two classes (*e.g.*, face vs. background). The feature selection is done by *AdaBoost*, which have been former proposed by Tieu and Viola [123]. However, their approach work off-line, which limits the usage for many applications. For example, tracking requires adaptive techniques for adjusting to possible variations of the target object over time. This can only be handled with adaptive methods, which are able to incrementally update their representations.

To overcome the limitations of off-line/batch learning authors propose a "pseudo-on-line" batch processing mode, where collecting a set of training examples and then run the off-line algorithm again and again. Javed *et al.* [55] propose to use on-line learning in a co-training framework for object detection. In their work a classifier is first trained off-line on a generic scenario, which is later adapted and refined on-line. A similar quasi on-line approach is used by Avidan for the application of visual tracking [5].

In comparison, we propose a general algorithm for on-line feature selection by directly extending the *Off-line Boosting for Feature Selection* algorithm introduced by Tieu and Viola [123] to the on-line case. In order to develop the *On-line Boosting for Feature Selection* algorithm we take an on-line variant of *AdaBoost* [92]. Note, other on-line (ensemble) algorithms, like *Winnow* [75] or the *Weighted Majority* algorithm [76] use a fixed set of weak classifiers (the experts), that are trained and combined using weights in an on-line manner. In other words, they send identical training sequences to each expert and hence the diversity is not enforced by the samples itself. In comparisons, the on-line boosting algorithm overcomes these limitations. This was one reason for us to investigate in on-line boosting and not any other on-line machine learning algorithm. The second reason was the great success of its off-line counterpart, which already have been applied to the feature selection task by Viola *et al.* [123, 130].

1.3.1 Our Contribution

Our work presents a **general approach**, which allows to perform **on-line learning** for **feature selection** using boosting. The proposed algorithm is a supervised two class classification algorithm, *i.e.*, it needs positive and negative labeled training examples. In fact, it selects proper features in order to discriminate between these two classes. In the following we focus on computer vision applications:

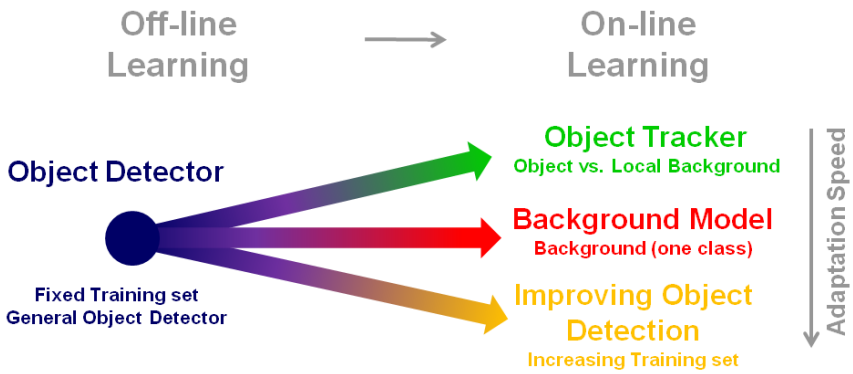


Figure 1.5: On-line learning opens new areas of applications in computer vision. For example, detection and tracking can be viewed as the same problem, depending on how fast the classifier adapts to the current state.

Generality: Many computer vision problems can be formulated as binary classification problems, such as improving detectors (*e.g.*, adaptation to a specific scene), tracking, recognition, and background modeling (see Figure 1.5).

On-line Learning: The benefit of on-line learning is twofold. It facilitates (i) learning from large databases, and (ii) learning when the data is not completely available at the beginning. Hence, the classifier can change over time, *i.e.*, adapt itself. On the right hand side of Figure 1.5 the adaptation speed variates from fast (top) to slow (bottom), whereas on the left hand side the off-line classifier cannot adapt at all.

Feature Selection: A subset of simple image features is selected in order to solve the classification problem, which makes it applicable for computer vision applications.

Summarizing, feature selection in combination with on-line learning allows to adapt the model by exchanging features, *i.e.*, a subset of features is selected, which is currently useful. Hence, the classifier focuses on a "simpler" subproblem. For example, in tracking the task is narrowed down to differentiate between the current object appearance and the local background (see Section 8.1).

1.4 Outline

The thesis is organized in two main parts. Part I introduces the *On-line Boosting for Feature Selection* algorithm. Part II demonstrate the applicability of the proposed algorithm for different computer vision applications. More precise, the parts are structured as follows:

Part I: First, in Chapter 2, we briefly review the two main concepts, which are (i) ensemble methods, in particular boosting, and (ii) feature selection. We summarize how off-line boosting can be used to perform feature selection and which features can efficiently be used for computer vision applications. In Chapter 3, we develop the *On-line Boosting for Feature Selection* algorithm based on an on-line version of boosting. We show extensions, which have been made in order to make the algorithm more suitable for practical applications and to increase the performance. Finally, we discuss our approach and show the principle behavior by a 2D-toy experiment.

Part II: In this part the developed algorithm is applied on different computer vision tasks. However, in many practical applications only partially labeled or unlabeled data is available. Since the proposed algorithm is fully supervised learning algorithm, labels have to be generated for the unlabeled samples in order to apply it. We focus mainly on the role of supervision, *i.e.*, how new (unlabeled) data can be incorporated into the already existing model. First, in Chapter 5, a brief review is given. The rest of the part is structured according to specify methods, where a chapter is used for (i) fully supervised learning, (ii) learning using a verifier, (iii) self-learning and (iv) learning with fix update rules. The update strategies are discussed for multiple applications like improving object detectors, background modeling, and visual tracking. For each of them we shortly summarize the method and show selected experiments as well as discuss problems and limitations.

Finally, in Chapter 10, we give a conclusion by summarizing open questions. Furthermore, ongoing work is briefly presented which gives some ideas to overcome the limitations shown before.

Part I

On-line Boosting for Feature Selection

Contents of Part I

Chapter 2

Preliminaries

Our goal is to develop a novel on-line feature selection algorithm based on boosting. Before we can introduce this algorithm we have to review related work and the basic components. First, we define on-line and off-line machine learning. Second, we take a look on feature selection methods. Third, ensemble methods and boosting are reviewed. Finally, we demonstrate how off-line boosting is used to perform feature selection and show how it can be applied to computer vision.

2.1 Off-line and On-line Machine Learning

2.1.1 Definitions

Expert Systems builds a model using given rules and facts, which are provided by a human expert and his experience. Using logical inference, the system is able to provide answers to questions. The difficult part is to build the (complex) knowledge basis. In contrast, *Machine Learning* models the behavior of input/output correspondences using given samples, *i.e.*, no expert is needed. In this thesis, we focus on supervised machine learning, *i.e.*, the target value is known. However, especially in Part II of this thesis applications are considered, that have the need of unsupervised learning. Nevertheless, the learning algorithm itself is supervised and labels are generated by some sort of supervision.

In the following, basic definitions are given based on [19, 82, 129].

Sample: A sample $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ is a set of features $\mathbf{x} \in \mathcal{X}$, which is assigned a target value $y \in \mathcal{Y}$.

Learning Problem: The learning problem is defined on a probability density function P on the set of samples $\mathcal{X} \times \mathcal{Y}$. If $\mathcal{Y}$ containers a finite number of elements the task is considered as classification problem, otherwise as regression problem.

In this thesis, we focus on the binary classification problem, *i.e.*, $\mathcal{Y} = \{-1, +1\}$. Furthermore, we consider $\mathcal{X} = \mathbb{R}^d$, where d is the dimensionality of the input vector (*e.g.*, $\mathbf{x} \in \{0, \ldots, 255\}^d$ is a 8 bit gray-value image with d pixels).

Hypothesis: Learning is formally the estimation of a function $f : \mathcal{X} \to \mathcal{Y}$. Particular knowledge is given by a samples set, *i.e.*, supervised learning is to predict the true label y_i correctly via $\hat{y} = f(\mathbf{x}_i)$. By extending this particular knowledge of f encoded by the samples $(\mathbf{x}, y) \subseteq \mathcal{X} \times \mathcal{Y}$ to the full space $\mathcal{X}$ a hypothesis is build. A hypothesis $H : \mathcal{X} \to \mathcal{Y}$ now models the underlying data generating process.

Hypothesis Class: A hypothesis class $\mathcal{H}$ is a set of hypotheses, which are considered by a special learning algorithm and have the same complexity, *i.e.*, *VC-Dimension* [129]).

Learning Algorithm: A learning algorithm A is a function $A : (\mathcal{X} \times \mathcal{Y})^\star \to \mathcal{Y}^\mathcal{X}$, where $(\mathcal{X} \times \mathcal{Y})^\star = \bigcup_{l=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^l$ is the set of all samples and $\mathcal{Y}^\mathcal{X}$ is the set of all functions $\mathcal{X} \to \mathcal{Y}$. Machine learning can be considered as the search for an optimal hypothesis within a given hypothesis class $H^\star \in \mathcal{H}$, which fits best to the given samples.

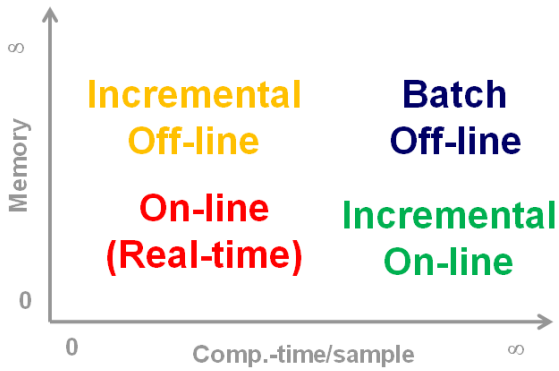


Figure 2.1: Resources typically needed (allowed) for the learning methods.

There are many ways the categorize learning methods. The distinction are overlapping and can be confusing and the used terminology is very inconsistent (*e.g.* [3, 28, 127]). The term *batch* learning is used quite consistently in the literature but *incremental* is often used for *on-line*, *constructive*, *adaptive*, *real-time* or *sequential* learning. Here, we follow mainly the definitions given by Warren S. Sarle[1]. The two distinct concepts are on-line vs. off-line and batch vs. incremental. The terms are defined in the following, and an illustration is given in Figure 2.1.

Batch Learning: After initializing the model, learning processes *all* the training data and updates the model. This is repeated until a certain stopping criterion (the average error, number of iterations, gradient is to small,...) is met. The iterations are also called epochs.

Incremental Learning: After initializing the model, learning processes in each iteration only *one* training example and then directly performs an update of the model.

Off-line Learning: In off-line learning *all* the data is given in advance. The stored samples can be accessed repeatedly. Therefore, batch learning is always off-line.

On-line Learning: In on-line learning each training sample is discarded after it has been processed and the model is updated. Furthermore, since an on-line algorithm has to deal with limited resources, the computational effort for processing one example and the consumed memory should stay constant. On-line learning is always incremental, but note, incremental learning can be done on-line or off-line.

[1] Web-Archive: *ai-faq/neural-nets/part2*, `ftp://ftp.sas.com/pub/neural/FAQ2.html`, (February 20, 2008)

However, the overall goal, is to minimize an objective function. For off-line learning this is done over summing the loss of all training instances $\sum_{l=1}^{L} \text{loss}(H(\mathbf{x}_l), y_l)$ where $\text{loss}(\hat{y}, y)$ is a given loss function. A loss function loss : $\mathbb{R} \times \mathcal{Y} \rightarrow \mathbb{R}_+$, which takes the expected target $\hat{y}$ and the true target y, whereas in on-line learning usually the cumulative loss is minimized $\sum_t \text{loss}(H_{t-1}(\mathbf{x}_t), y_t)$ (this setting is broadly called the *Mistake Bound* learning model [9]). Many different loss-functions are defined, *e.g.*, the misclassification error (zero-one loss), the exponential loss (used in boosting), the Hinge loss (used by support vector machines) or the squared error [26].

2.1.2 Off-line Learning

In off-line learning all training samples must be given in advance. More formally, a fixed training set $\mathcal{S}^{fix} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_L, y_L)\}$ with L labeled samples is given. The samples are drawn from a distribution P, which is defined on the whole set of samples $\mathcal{S} = \mathcal{X} \times \mathcal{Y}$. A machine learning algorithm looks for a hypothesis H within a given hypothesis class, which fits best to the examples. A hypothesis extrapolates from the specific samples $\mathcal{S}^{fix} \subseteq \mathcal{S}$ to a complete mapping $H : \mathcal{X} \rightarrow \mathcal{Y}$. Thus, it is a model for the underlying process, which generates the data (see Figure 2.2 (a)).

2.1.3 On-line Learning

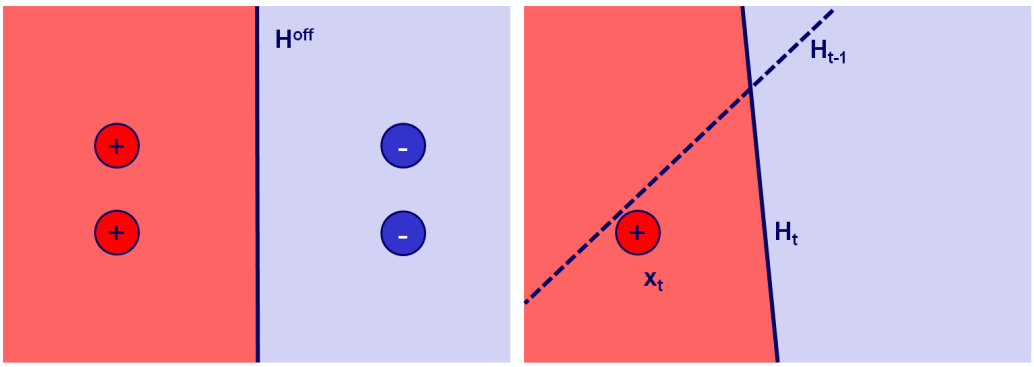
Similar to the definition of Giraud-Carrier [28] we define[2]:

On-line Learning Task: A learning task is on-line, if the training examples used to solve it are not available a priori but become available over time, usually one at a time. Learning my need to go on (almost) indefinitely.

On-line Learning Algorithm: A learning algorithm is on-line if, for any given training sample $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$, it produces a sequence of hypotheses $h_0, h_1, ..., h_T$, such that h_t depends only on h_{t-1} and the current sample $(\mathbf{x}_t, y_t)$. The main characteristics of an on-line learning algorithm are: no re-processing of previous examples is necessary since each h_t is viewed as a best approximation so far of the target application, the learner can, at any time, produce an answer to a query and the quality of its answer improves over time. Hence, there is no separation on the training and recognition stage, they run in a loop.

In other words, on-line learning algorithms, contrary to off-line learning algorithms, see each sample only once. In supervised on-line learning we are dealing with a

[2]In Giraud-Carriers words it is an incremental learning task, but he does not distinguish between incremental and on-line learning.



(a) off-line (b) on-line

Figure 2.2: In off-line learning (a) all training data is available from the beginning, whereas in on-line learning (b) a current hypothesis H_{t-1} is updated by a new training sample $\mathbf{x}_t$ in order to get a new hypothesis H_t.

sequence of labeled samples , which are drawn from the distribution P. An on-line learning algorithm takes as input a hypothesis $H_{t-1} : \mathcal{X} \rightarrow \mathcal{Y}$ and a new training example $(\mathbf{x}_t, y_t)$. The algorithm returns an updated hypothesis

$$H_t = \text{update}(H_{t-1}, (\mathbf{x}_t, y_t)). \tag{2.1}$$

At any time t a model is available, since the hypothesis models the data generating process (see Figure 2.2 (b)).

In general, off-line classifiers tend to perform better since they are able to build statistics based on all examples at once. A lossless on-line learning algorithm is an algorithm that returns a hypothesis exact to what the corresponding off-line algorithm would return given the same training data. However, on-line algorithms have advantages or are even necessary in the following cases:

Large Training Data: The whole data does not fit into memory at once. On-line algorithm can thus cope with and even benefit from a large amount of data. The on-line algorithm is able to sample from the overall distribution and not only from the fixed subset as it is the case in the off-line versions. The explosive increase of data and information makes on-line learning algorithm more and more important for large scale learning approaches.

Availability: Not all the data is available at the beginning. Further, the data generation process itself may change over time, *i.e.*, the distribution P is a time depended one P_t. Hence, the goal of an on-line learner is to forget irrelevant information and specialize to the current situation, which usually is an easier but time dependent sub-problem [9, 127].

2.2 Pattern Recognition System

Following Duda *et al.* [19] a pattern recognition system can be subdivide into

(i) sensing,

(ii) segmentation,

(iii) **feature extraction/selection**,

(iv) **classification**, and

(v) post-processing.

In this thesis, we focus mainly on the points (iii) and (iv). The features and the classifier are somehow coupled (see also Section 1.2). On the one hand, if an ideal feature is given the classifier can be very simple, *e.g.*, the decision is if the feature is present or not. On the other hand, if we have access to a powerful classifier the need for clever feature extraction/selection is not necessary.

2.2.1 Feature Selection

Feature selection aims to determine useful features after feature extraction based on appropriate rules. Feature selection reduces the dimensionality of the feature space and removes the redundant, irrelevant or noisy data. Thus, it speeds up, improves the data quality, and increases the accuracy of the final hypotheses [8]. Standard feature selection methods can be broadly divided into three methods (following [45]), which are described in the following:

Filter Methods: They typically use some kind of heuristics to estimate the relative importance of different features. This can either be done by evaluating each feature separately or as a set (combination) of features. The algorithm then chooses a subset of n features. One can argue that a disadvantage is that the algorithm will select redundant features, because similar features will achieve similar weights. Summarizing, filter methods find a good subset before a machine learning algorithm is applied.

Wrapper Methods: This methods directly evaluate the performance of a subset of features by measuring the performance of a model trained on that subset. Thus, the extremal case is an exhaustive search of all possible feature subsets ($2^{|\mathcal{F}|}$), which, indeed, is enormous. The advantages are that it delivers the optimal subset for the used learning algorithm, it captures the combination of the features, and removes redundant features. The disadvantage is, that it is very inefficient.

Filter-Wrapper-Hybrid: Embedded methods perform variable selection in the process of training and are usually specific to given learning machines. These approaches, after feature extraction, somehow combine the steps of feature selection and classifier training in one framework. At each stage an evaluation function is used to select an attribute, that has the best ability to discriminate among the classes. In each iteration, with an user supplied feature, the algorithm ranks the features, that have been selected so far and adds the highest ranking feature to the feature subset. For example, *FeatureBoost* [87] is such a hybrid algorithm. The goal is to search for alternate hypotheses amongst the features. A distribution over features is kept and updated at each iteration.

The approaches used in this thesis, *Off-line Boosting for Feature Selection* and its counterpart *On-line Boosting for Feature Selection*, are filter-wrapper-hybrid approaches.

2.3 Ensemble Methods and Boosting

Ensemble methods can improve the performance of a given learning algorithm through the combination of base models. The fusion is a mapping from local decision (meta-feature) to the final decision (see Figure 2.3).

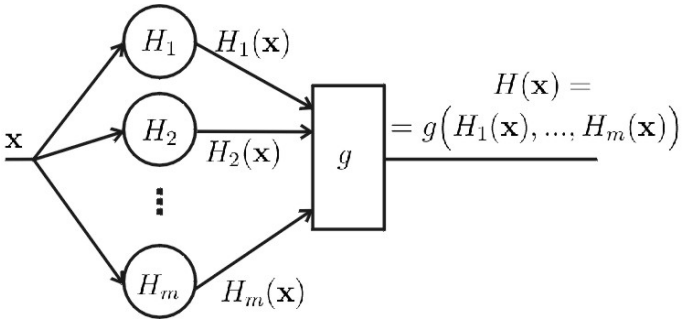


Figure 2.3: Using a proper combination of individual hypotheses $H_1(\mathbf{x}), \ldots, H_n(\mathbf{x})$ a *master*-hypothesis $H(\mathbf{x})$ is constructed, which improves the performance. In order to construct different hypotheses, this can be done by applying the same learning algorithm on (i) different subset of the training data (bagging); (ii) different weighted training data (boosting) or (iii) using different learning algorithm (stacking).

Ensemble construction is one of the fields of machine learning, that is receiving most research attention, mainly due to the significant performance improvements over single classifiers, that have been reported with ensemble methods. The main steps are model generation and combination [18]. The individual classifiers should be as accurate as possible, but simultaneously should disagree as much as possible.

These two objectives are somehow conflicting. If the classifiers are more accurate, it is obvious that they must agree more frequently.

Many methods have been developed to enforce diversity on the classifiers, *e.g.*, using different combination schema, different classifier models, different feature subsets, or different training samples. The first method is also known as "voting classifier ensembles". A sample is presented to all "experts" (base models) of the ensemble and their outputs are combined in some manner (*e.g.*, voting, averaging) in order to yield the finial decision (*e.g.*, the *Mixture of Experts* [54] approach). When using different training samples (training datasets), the learning algorithms run several times, each time with a different partition of the whole training set. *Bagging* and *Boosting* corresponds to that group.

Bagging [13] (after bootstrap aggregating) just generates different bootstrap samples from the training set. Boosting methods adaptively change the distribution of the training set based on the performance of the previous classifiers. Unlike bagging, which is largely a variance reduction method, boosting appears to reduce both bias and variance.

2.3.1 Discrete *AdaBoost*

We focus on discrete *AdaBoost* (adaptive boosting), a specific boosting algorithm for classification proposed by Freund and Schapire [24] (a good overview is given in [25]). Boosting is very popular and, hence, various variants have been developed (*e.g.*, *Real-Boost* [24], *LP-Boost* [16]). *AdaBoost* has been analyzed carefully (*e.g.*, [112]) and tested empirically by many researchers. For instance, following the overview given in [111], boosting has been used for text recognition, text filtering, routing, "ranking" problems, learning problems in natural language processing, medical diagnostic, and customer monitoring and segmentation.

In general, boosting is a method for improving the accuracy of any given learning algorithm. This is done by combining (a weighted voting) of N hypotheses which have been generated by repeating training with different subsets of training data. Let us first define some terms:

Weak classifier: A weak classifier has only to perform slightly better than random guessing, *i.e.*, for a binary decision task, the error rate must be less than 50%. The hypothesis $h : \mathcal{X} \rightarrow \{-1, +1\}$ generated by a weak classifier is obtained by applying a learning algorithm.

Strong classifier: Given a set of N weak classifiers, a strong classifier is computed as linear combination of the weak classifiers. The value $f(\cdot)$ (which is related

to the margin) can be interpreted as a confidence measure.

$$H(\mathbf{x}) = \text{sign}(f(\mathbf{x})), \qquad \text{where} \qquad f(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n h_n(\mathbf{x}) \tag{2.2}$$

In addition, it can be easily shown [26] (see Appendix A) that boosting develops the likelihood estimator

$$H(\mathbf{x}) = \frac{1}{2} \log \left(\frac{P(y=1|\mathbf{x})}{P(y=-1|\mathbf{x})} \right) \tag{2.3}$$

or equivalently

$$P(y=1|\mathbf{x}) = \frac{\exp(H(\mathbf{x}))}{\exp(H(\mathbf{x})) + \exp(-H(\mathbf{x}))}. \tag{2.4}$$

The basic algorithm, shown in Algorithm 1, works as follows: Given a training set $\mathcal{S}$ with positive and negative labeled samples and an initial uniform distribution $\mathbf{w}_0 = (w_{0,1}, \ldots, w_{0,L})$, $w_{0,l} = \frac{1}{L}$ over the examples $l = 1, \ldots, L$. A weak classifier $h(\mathbf{x})$ is trained based on the training set $\mathcal{S}$ and weights $\mathbf{w}(\mathbf{x})$, *i.e.*,

$$h_n = \arg\min_{h_n} \sum_{l=1}^{L} w_{n,l} \cdot \begin{cases} 1 & h_n(\mathbf{x}_l) \neq y_l \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

Based on the weighted error e_n (with respect to $\mathbf{w}_n$) of the weak classifier h_n it gets assigned a voting-weight

$$\alpha_n = \frac{1}{2} \ln \left(\frac{1-e_n}{e_n} \right). \tag{2.6}$$

Finally, the weight distribution

$$w_{n+1,l} = w_{n,l} \cdot \begin{cases} \exp(-\alpha_n) & h(\mathbf{x}_l) = y_l \\ \exp(\alpha_n) & h(\mathbf{x}_l) \neq y_l \end{cases} \tag{2.7}$$

is updated such that the probability increases for the samples, that were misclassified. If the sample is classified correctly the corresponding weight is decreased. Therefore, the algorithm focuses on the difficult examples. The process is repeated and at each boosting iteration a new weak hypothesis is added until a certain stopping condition is met (*e.g.*, a given number of weak classifiers are trained).

This is a shift in mind: instead of trying to design a learning algorithm that is accurate over the entire space, we can instead focus on finding learning algorithms, that only need to be better than random guessing and combine them.

Freund and Schapire [24] proved strong bounds on the training and generalization error of *AdaBoost* (see Appendix A). For the case of binary classification the training

Algorithm 1 Off-line *AdaBoost* (Freund and Schapire, [24])

Require: training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\} | y_i \in \mathcal{Y} = \{-1, +1\}$

//initialize weights

$w_{0,l} = 1, \quad l = 1, ..., L$

for $n = 1, 2, .., N$ **do**

//normalization of weights such that $\mathbf{w}_n$ is a distribution

$w_{n,l} = \frac{w_{n-1,l}}{\sum_{i=1}^{L} w_{n-1,i}}$

//train weak classifier with respect to $\mathbf{w}_n$

$h_n = \arg\min_{h_n} \sum_{l=1}^{L} w_{n,l} \cdot \begin{cases} 1 & h_n(\mathbf{x}_l) \neq y_l \\ 0 & \text{otherwise} \end{cases}$

//calculate error

$e_n = \sum_{l:h_n(\mathbf{x}_l) \neq y_l} w_{n,l}$

if $e_n = 0$ or $e_n > \frac{1}{2}$ **then**

exit

end if

//calculate voting factor

$\alpha_n = \frac{1}{2} \ln \left(\frac{1-e_n}{e_n} \right)$

//update prob. distribution

$w_{n+1,l} = w_{n,l} \cdot \begin{cases} \exp(-\alpha_n) & h(\mathbf{x}_l) = y_l \\ \exp(\alpha_n) & h(\mathbf{x}_l) \neq y_l \end{cases}$

end for

//output the final (strong) classifier

$H(\mathbf{x}) = \text{sign} \left(\sum_{n=1}^{N} \alpha_n \cdot h_n(\mathbf{x}) \right)$

error drops exponentially fast with respect to the number of boosting rounds N (*i.e.*, number of weak classifiers). One may suggest that boosting will overfit if it run for too many rounds, *i.e.*, as N becomes large. Empirically, it was shown that boosting often does not overfit, even when running for thousands of rounds. Moreover, it was observed that *AdaBoost* continues to drive down the generalization error long after the training error had reached zero. In response to these empirical findings, Schapire *et al.* [112] gave an alternative analysis in terms of the margins of the training examples. Large margin on the training set can be translated into a superior bound on the generalization error. More recently, an analysis of the behavior of *AdaBoost* was presented by Rudin *et al.* [110].

Furthermore, boosting can be viewed as a maximum margin classifier which achieves a hard margin (*i.e.*, focuses on the "hard" to learn samples) and so it is also related to *Support Vector Machines* (SVM) [102]. However, since hard margin is achieved the algorithm is very noise sensitive. Modification of the algorithm are proposed to overcome this issue, *e.g.*, [56, 90].

2.4 Off-line Boosting for Feature Selection

As mentioned earlier, feature selection and feature combination is highly relevant in machine learning. Compared to other approaches, boosting has many advantages; therefore it is quite popular [103].

Tieu and Viola [123] introduced boosting methods for feature selection. Also other authors investigated in this direction, *e.g.*, [87, 103]. Boosting is used as a meta learning algorithm for feature subset selection. The idea behind these approaches is that each feature corresponds to a single weak classifier and boosting selects from the features. A pool of possible features $\mathcal{F}$ is given. Since this feature pool can be very large, for computational reasons the algorithm focuses on a subset $\mathcal{F}_{sub} = \{f_1, ..., f_k\} \subseteq \mathcal{F}$.

Training proceeds in a similar manner to standard boosting, as shown in Algorithm 2. In each iteration n the algorithm selects one new feature and adds it (with the corresponding voting factor) to the ensemble. All features are evaluated and the best one is selected, which forms the weak hypothesis h_n. The weight α_n is set according to the error of h_n. Finally, a strong classifier H is computed as weighted linear combination of the weak classifiers. As already stated the training error drops exponentially fast over the boosting iterations, which are now equivalent to the number of selected features.

2.5 Image Features

The main reason why in computer vision features are used instead of raw pixel values as input to a learning algorithm is to reduce the intra-class variability while increasing the extra-class variability. In addition, "ad-hoc" knowledge can be included.

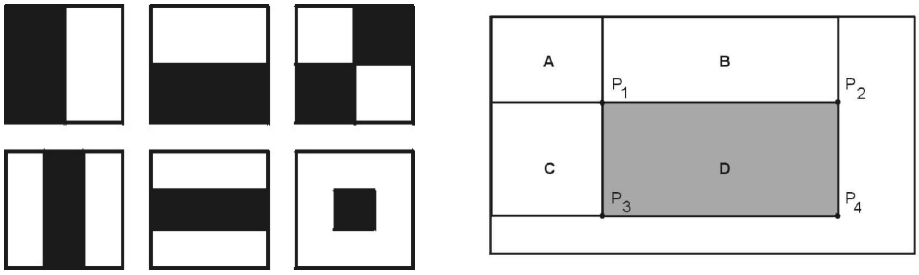
The *Off-line Boosting for Feature Selection* algorithm was used in the seminal work of Viola and Jones [130] in order to build a real-time face detector. The main assumption is that a small number of simple image features is sufficient to distinguish the object appearance from background. To be precise, Haar-like features are used as shown in Figure 2.4 (a). These six different prototypes are scaled independently in height and width and form a specific feature. The feature value of a Haar-like

Algorithm 2 Off-line *AdaBoost* for Feature Selection (Tieu and Viola [123])

Require: training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\} | y_i \in \mathcal{Y} = \{-1, +1\}$

```
//initialize weights
w_{0,l} = 1,    l = 1, ..., L

for n = 1, 2, .., N do
    //normalization of weights such that w_n is a distribution
    w_{n,l} = w_{n-1,l} / sum_{i=1}^{L} w_{n-1,i}

    // train all weak classifiers
    for m = 1, 2, .., M do
        //train weak classifier with respect to w_n
        h_{n,m} = arg min_{h_n} sum_{l=1}^{L} w_{n,l} * { 1  h_n(x_l) != y_l
                                                        { 0  otherwise

        // calculate error
        e_{n,m} = sum_{i:h_{n,m}(x_i) != y_i} w_{n,i}
    end for

    // choose weak classifier with the lowest error
    m+ = arg min_m (e_{n,m})
    e_n = e_{n,m+};  h_n^{sel} = h_{n,m+}
    if e_n = 0 or e_n > 1/2 then
        exit
    end if

    // calculate weighting factor
    alpha_n = 1/2 * ln((1 - e_n) / e_n)

    // update weight distribution
    w_{t+1,i} = w_{t,i} * { exp(-alpha_t)  h(x_i) = y_i
                          { exp(alpha_t)   h(x_i) != y_i
end for
```

feature is calculated as the sum of pixels within rectangular regions, which are either positive (black regions) or negative (white regions) weighted. Note, the computation of all these feature types can be done very efficiently using integral images [130] and integral histograms [100] as data structures. This allows for exhaustive template matching for the whole image. An integral image, denoted as II, sums up all the pixel values from the upper left up to the current position. More formally, it is



(a) Haar-like features (b) integral image

Figure 2.4: The value of the classical HAAR-like feature (a) is the difference of pixel values between the white and the black marked region, which can be efficiently calculated using the integral image (b). Efficient calculation of the sum over a rectangular area. The value of the integral image at Position P_1 is the sum of the pixel values in region A. P_2 corresponds $A+B$, P_3 to $A+C$ and P_4 to $A+B+C+D$. Therefore, the sum over the area D can be calculated by $P_4 + P_1 - P_2 - P_3$.

defined on an image I as

$$II(x,y) = \sum_{x'=1}^{x} \sum_{y'=1}^{y} I(x',y'). \tag{2.8}$$

The pre-calculation of an integral image for all pixels can be efficiently implemented in one pass over the image. Afterwards, any sum of any rectangular region can be computed by only 4 memory accesses and 3 additions, see Figure 2.4 (b) for an example. This allows to do exhaustive template matching when scanning the whole image. Since these features can only be calculated over rectangular regions the generic angles have to fit to the object models appearance. This problem can be solved by computing the integral structures at generic angles. An approximation for a given angle is easily computed by the use of integral image. Lienhard [74] introduced an additional set of rotated features. Barczack *et al.* [7] proposed a method to convert a previous trained classifier to work at any angle, so rotated objects can be detected. Another approach is used by Wu *et al.* [138] for fast rotation invariant face detection.

In order to select a "good" subset from this highly over-complete feature set, boosting for feature selection is applied on a large set of positive labeled images (about 5000). Negative image are bootstrapped from a set of background images, *i.e.*, images which do not contain the object of interest (*i.e.*, the current classifier is applied on them and selects those sub-patches were it has a high response). In order to use boosting for feature selection, each image feature corresponds to a weak classifier. To obtain a weak classifier h_j from a feature j, we model the probability distribution of this feature for positive and negative samples with $f_j(\mathbf{x})$ evaluating this feature

on the image $\mathbf{x}$. A hypothesis is achieved by a simple decision stump [130] and hence we get for a feature j

$$h_j(\mathbf{x}) = p_j \text{sign}(f_j(\mathbf{x}) - \theta_j), \tag{2.9}$$

where $\theta_j := 0.5|m^+ + m^-|$ is the threshold and $p_j := \text{sign}(m^+ - m^-)$ the parity of the decision. The values

$$m^+ = \frac{1}{|X^+|} \sum_{\mathbf{x} \in \mathcal{X}^+} f_j(\mathbf{x}) \qquad \text{and} \qquad m^- = \frac{1}{|X^-|} \sum_{\mathbf{x} \in \mathcal{X}^-} f_j(\mathbf{x}) \tag{2.10}$$

are the calculated means of the feature responses for all positive $\mathcal{X}^+$ and negative $\mathcal{X}^-$ samples.

The work of Viola and Jones, paved the way for boosting in the area of computer vision. Many authors analyzed and used this approach with different extensions. Some of them considers other (more sophisticated) feature types (*e.g.*, [73, 86, 133, 142]). Other extensions address the problem of the learning algorithm in order to make it more stable and noise invariant, *e.g.*, [101]. Others take the fact into account, that the learning problem is highly unbalanced, *i.e.*, much more negative image patches are available [131]. Other improvements focus an the efficiency to further speed up the recognition process (*e.g.*, [11, 120, 125]).

Chapter 3

On-line Boosting for Feature Selection[1]

In this chapter we introduce the novel *On-line Boosting for Feature Selection* algorithm. In order to develop this algorithm we take an on-line variant of *AdaBoost* [92]. Thus, we first, review this method, which is then extended to perform feature selection. We refer to the proposed algorithm in this chapter as the "original" one. All further improvements are yielded as extensions and are described in Chapter 4. Proof of concept is done by showing an illustrative experiment. By defining simple image features, we later make it applicable for various computer vision applications, which are indeed shown in the second part of this thesis.

[1] Adapted and extended from H. Grabner, and H. Bischof. On-line Boosting and Vision. *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 260-268, 2006. [30]

3.1 On-line Boosting

To obtain an on-line boosting algorithm (*i.e.*, an algorithm, that operates on a single example and discards it after updating), each of the boosting steps described in Section 2.3.1 has to be done on-line. Thus a set of weak classifier $(h_1, \ldots, h_n)$ and their corresponding voting weights $(\alpha_1, \ldots, \alpha_n)$ are given, which form the current strong classifier $H(\mathbf{x})$. This (current) classifier is updated whenever a new sample arrives [63]. On-line updating the weak classifiers is not the problem because many on-line learning algorithms for generating a hypothesis can be used. The crucial step is the computation of the weight distribution for the samples, because we do not know a priori the difficulty of a sample (*i.e.*, we do not know if we have seen the sample before).

We use ideas proposed by Oza and Russell [92] and the experimental comparisons they did [91]. The basic idea is that the importance (difficulty) of a sample can be estimated by propagating it through the set of weak classifiers. One can think of this as modeling the information gain with respect to the first n classifier and code it by the importance weight λ_n (initialized by $\lambda_0 = 1$). The update of the $n+1$-th weak classifier is done with respect to the importance weight λ_n of the current sample. Either λ_n is used as a learning rate in the learning algorithm or by k-times repeated updating $k \sim \text{Poisson}(\lambda)$ as proposed by Oza. For updating the weak classifiers, any on-line learning algorithm can be used. In fact, the error of the n-th weak classifier is estimated by

$$\hat{e_n} = \frac{\lambda_n^{wrong}}{\lambda_n^{wrong} + \lambda_n^{corr}}, \tag{3.1}$$

where the weights λ_n^{wrong} and λ_n^{corr} are the sum of importances of correctly and incorrectly classified samples seen so far. In fact at the n-th selector the importance λ_{n-1} is added to one of them, *i.e.*

$$\lambda_n^{corr} = \lambda_n^{corr} + \lambda_{n-1} \qquad h_n(\mathbf{x}) = y \tag{3.2}$$

$$\lambda_n^{worng} = \lambda_n^{wrong} + \lambda_{n-1} \qquad h_n(\mathbf{x}) \neq y. \tag{3.3}$$

With respect to the estimated error $\hat{e_n}$ the voting weight

$$\alpha_n = \frac{1}{2} \log \left(\frac{1 - \hat{e}_n}{\hat{e}_n} \right) \tag{3.4}$$

can be calculated. The update of the importance λ for the $n+1$-th weak classifier is done again according to the error and the decision of the weak classifier

$$\lambda_n = \lambda_{n-1} \cdot \begin{cases} \frac{1}{2(1-\hat{e}_n)} & h_n(\mathbf{x}) = y \\ \frac{1}{2(\hat{e}_n)} & h_n(\mathbf{x}) \neq y. \end{cases} \tag{3.5}$$

Algorithm 3 On-line *AdaBoost* (Oza and Russell [92])

Require: training example $(\mathbf{x}, y)$, $y \in \{-1, +1\}$
Require: strong classifier H (initialized randomly)
Require: weights λ_n^{corr}, λ_n^{wrong} (initialized with 1)

initialize the importance weight $\lambda_0 = 1$
// for all selectors
for $n = 1, 2, .., N$ **do**

// update each weak classifier
$h_n = \text{update}(h_n, \langle \mathbf{x}, y \rangle, \lambda_{n-1})$

// estimate errors
if $h_n(\mathbf{x}) = y$ **then**
$\lambda_n^{corr} = \lambda_n^{corr} + \lambda_{n-1}$
else
$\lambda_n^{wrong} = \lambda_n^{wrong} + \lambda_{n-1}$
end if
$\hat{e}_n = \frac{\lambda_n^{wrong}}{\lambda_n^{corr} + \lambda_n^{wrong}}$
if $\hat{e}_n = 0$ or $\hat{e}_n > \frac{1}{2}$ **then**
exit
end if

// calculate voting weight
$\alpha_n = \frac{1}{2} \cdot \ln\left(\frac{1-\hat{e}_n}{\hat{e}_n}\right)$

// update importance weight
if $h_n(\mathbf{x}) = y$ **then**
$\lambda_n = \lambda_{n-1} \frac{1}{2 \cdot (1-\hat{e}_n)}$
else
$\lambda_n = \lambda_{n-1} \frac{1}{2 \cdot \hat{e}_n}$
end if

end for

The on-line algorithm shown in Algorithm 3 requires that the number of weak classifiers is fixed at the beginning. Note, the interchange of roles: In the off-line case all samples are used to update (and select) one weak classifier, whereas in the on-line case one sample is used to update all weak classifiers and the corresponding voting weight.

Oza has proven, if off-line and on-line boosting are given the same training set, then the weak classifiers (Naive Bayes classifiers) returned by on-line boosting converges statistically to the one obtained by off-line boosting as the number of iterations $N \rightarrow \infty$. The proof sketch and the argumentation are the following: The first weak classifier gets the samples exactly as the off-line classifier. Since any on-line learning algorithm will yield the same results when it converged. The main point in boosting is to adaptively re-weight the samples with respect of the error of the former weak classifiers. Thus the weighting of the training samples change from weak classifier to weak classifier. If the first weak classifier is converged, then also the estimated error and the statistic will converge and thus the second weak classifier will be given the same examples and the same weight distribution as in the off-line case. Summarizing, after the n-th weak classifier is converged it delivers the correct weight distribution for the $n+1$-th classifier. Thus, training and convergence will start from the beginning. Therefore, for repeated presentation of the training set on-line boosting and off-line boosting deliver the same result, which yields to the following theorem:

Theorem (Oza [89], page 85): Let $h_n^{offline}(\mathbf{x})$ as the n-th weak model returned by off-line *AdaBoost* and define $h_n^{online}(\mathbf{x})$ as the n-th weak model returned by the on-line algorithm. Given the same training set, if $h_n^{online}(\mathbf{x})$ and $h_n^{offline}(\mathbf{x})$ for all $n \in \{1, 2, ..., N\}$ are Naive Bayes classifier, then $h^{online}(\mathbf{x}) \xrightarrow{P} h^{offline}(\mathbf{x})$.

Another interpretation is given in Appendix B, where we obtain similar results. Only the update of the importance weight λ, which is propagated through the set of selectors, differs.

3.2 On-line Boosting for Feature Selection

The approach of Oza and Russell, reviewed in the last section, is not directly applicable to feature selection. The essential novelty of our approach is, that we propose an on-line boosting algorithm for solving the feature selection task. For this purpose we need a further concept.

Selector: Given a set of M weak classifiers with hypothesis $\mathcal{H}^{weak} = \{h_1, ..., h_M\}$, a selector selects exactly one of those $h^{sel}(\mathbf{x}) = h_m(\mathbf{x})$ where m is chosen according to an optimization criterion. In fact, we use the estimated error e_i of each weak classifier $h_i \in \mathcal{H}^{weak}$ such that $m = \arg\min_i e_i$.

Similar to the off-line case, the weak classifiers $\mathcal{H}^{weak}$ correspond to features, *i.e.*, the hypotheses generated by the weak classifier is based on the response of the feature. One selector can therefore select from a subset of M features $\mathcal{F}_{sub} = \{f_1, ..., f_M \mid f_i \in$

$\mathcal{F}\}$ of its feature pool. Note, the selector can be interpreted as a classifier (it switches between the weak classifiers). Training a selector means that each weak classifier is trained (updated) and the best one (with the lowest estimated error) is selected.

> In summary: *The main idea is to apply on-line boosting not directly to the weak classifiers but to the selectors.*

In the off-line case this is not required, because at each boosting round one new weak classifier is added to improve the strong classifier. However, it is the same as the original algorithm [123, 130] as can be easily observed from Figure 3.1 (a), where at each time a new selector is added.

In contrast, the overall principle of our novel on-line approach is depicted in Figure 3.1 (b) and shown in Algorithm 4. In particular, the training of the new *On-line Boosting for Feature Selection* works as follows: First, a fixed set of N selectors $h_1^{sel}, .., h_N^{sel}$ is initialized randomly, each with its own feature pool $\mathcal{F}_n$. When a new training sample $(\mathbf{x}, y)$ arrives the selectors are updated. This update is done with respect to the importance weight λ of the current sample. The weak classifier with the smallest error is selected by the selector

$$\arg\min_m(\hat{e}_{n,m}), \qquad \text{where} \qquad \hat{e}_{n,m} = \frac{\lambda_{n,m}^{wrong}}{\lambda_{n,m}^{corr} + \lambda_{n,m}^{wrong}} \tag{3.6}$$

is the error of the m-th weak classifier $h_{n,m}$ in the in the n-th selector, estimated from the weights of correctly $\lambda_{n,m}^{corr}$ and wrongly $\lambda_{n,m}^{wrong}$ classified examples seen so far. Finally, the corresponding voting weight α_n and the importance weight λ_n of the sample are updated and passed to the next selector h_{n+1}^{sel}. This procedure is repeated for all selectors. The number of selectors is constant similar to the number of weak classifiers in Oza's on-line algorithm. A strong classifier is obtained by linear combination of selectors:

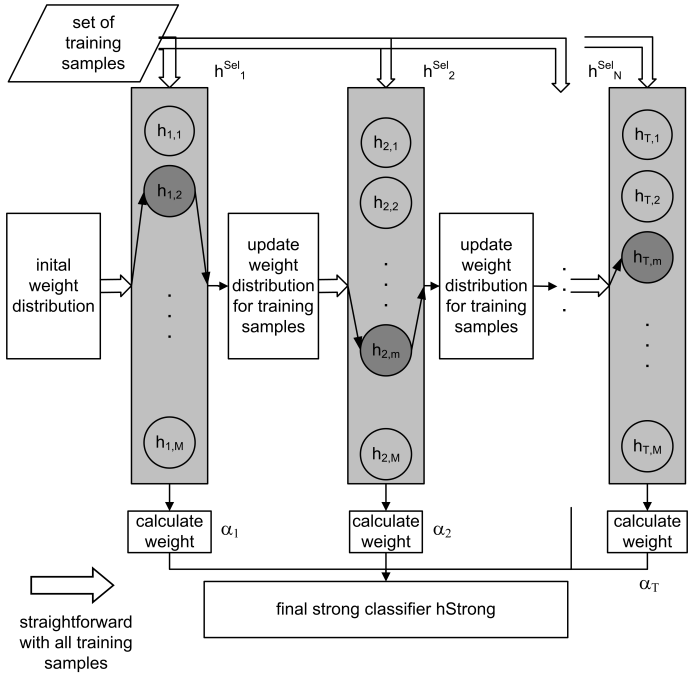
$$H(\mathbf{x}) = \text{sign}\Big(\sum_{n=1}^{N} \alpha_n h_n^{sel}(\mathbf{x})\Big). \tag{3.7}$$

In contrast to the off-line version a strong classifier is available at any time.

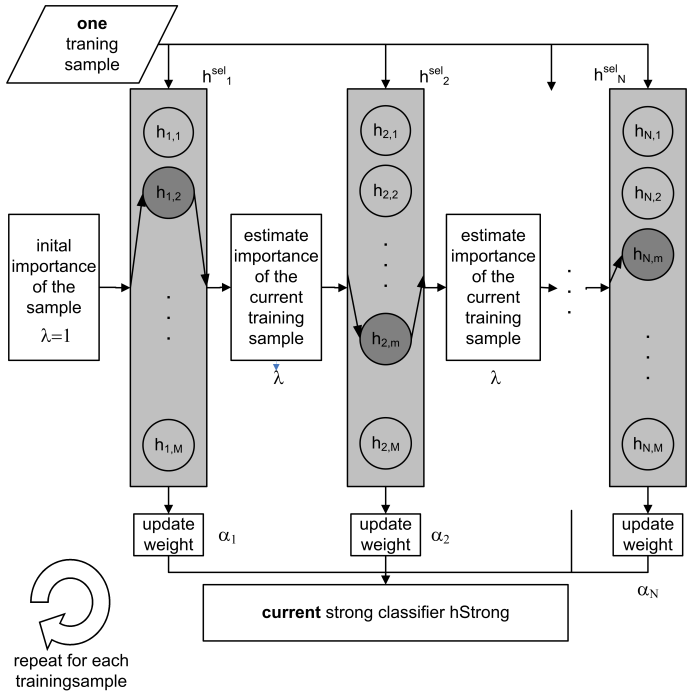
3.2.1 Discussion of the Switching

The information exchange within the selectors is done by the importance λ of the labeled example $\mathbf{x}$. As can be easily seen (Equation B.10), λ_n can be written as

$$\lambda_n = \exp(-yH_{n-1}(\mathbf{x})) = \exp\Big(-y\sum_{i=1}^{n-1} \alpha_i h_i^{sel}(\mathbf{x})\Big). \tag{3.8}$$



(a) Off-line boosting for Feature Selection



(b) On-line boosting for Feature Selection

Figure 3.1: Flow charts for the standard *Off-line Boosting for Feature Selection* using the new formalism of *selectors* (a) and the novel *On-line Boosting for Feature Selection* (b).

Algorithm 4 On-line *AdaBoost* for Feature Selection

Require: training example $(\mathbf{x}, y\rangle)$, $y \in \{-1, +1\}$
Require: strong classifier H (initialized randomly)
Require: weights $\lambda_{n,m}^{corr}$, $\lambda_{n,m}^{wrong}$ (initialized with 1)

initialize the importance weight $\lambda_0 = 1$
// for all selectors
for $n = 1, 2, .., N$ **do**

// update the selector h_n^{sel}
for $m = 1, 2, .., M$ **do**

// update each weak classifier
$h_{n,m} = \text{update}(h_{n,m}, \langle \mathbf{x}, y\rangle, \lambda_{n-1})$

// estimate errors
if $h_{n,m}(\mathbf{x}) = y$ **then**
$\lambda_{n,m}^{corr} = \lambda_{n,m}^{corr} + \lambda_{n-1}$
else
$\lambda_{n,m}^{wrong} = \lambda_{n,m}^{wrong} + \lambda_{n-1}$
end if
$\hat{e}_{n,m} = \frac{\lambda_{n,m}^{wrong}}{\lambda_{n,m}^{corr} + \lambda_{n,m}^{wrong}}$
end for

// choose weak classifier with the lowest error
$m^+ = \arg\min_m(e_{n,m})$
$\hat{e}_n = \hat{e}_{n,m^+}$; $h_n^{sel} = h_{n,m^+}$
if $\hat{e}_n = 0$ or $\hat{e}_n > \frac{1}{2}$ **then**
exit
end if

// calculate voting weight
$\alpha_n = \frac{1}{2} \cdot \ln\left(\frac{1-\hat{e}_n}{\hat{e}_n}\right)$

// update importance weight
if $h_n^{sel}(\mathbf{x}) = y$ **then**
$\lambda_n = \lambda_{n-1} \cdot \frac{1}{2 \cdot (1-\hat{e}_n)}$
else
$\lambda_n = \lambda_{n-1} \cdot \frac{1}{2 \cdot \hat{e}_n}$
end if

end for

Thus, it depends on the $n-1$ previous weak classifiers and their decisions. In the on-line setting all the classifiers are depending on time, hence, the statistics of the samples can only be correctly estimated when the classifier H_n^t draws the same decision for all former time indexes $t = 1, \ldots, T$. Otherwise the true statistic is disturbed. Thus, switching of the weak classifiers within one selector introduces noise to the whole learning process. However, switches of features is quite common and even necessary if the learning problem changes over time, *i.e.*, features have to be exchanged in order to be adaptive. The following strategies have been considered during the work on this thesis:

Reseting: After the n-th selector has switched to another weak classifier the corresponding learned parameters are reseted for all following selectors $n' > n$. Due to the fact, that the selectors converges sequentially, this approach is appropriable. No switching noise is introduced, however, many drawbacks exists: By reseting, all former seen examples are implicitly discarded for the following selectors and the complexity of the strong classifier is certainly reduced. In particular, this is a big disadvantage wen coping with changing environments, where early selectors are switching. Further on, the selectors may start oscillating if the errors are similar, which is usually the case for later onces with high errors. Nevertheless, reseting *some* statistics in combination with the *WaldBoost* algorithm (see Section 4.2) is useful.

Ignoring: Due to our empirical knowledge (from many experiments), we can also benefit by propagating the example through all weak classifiers even when they are not converged. This can be explained by the fact, that they share some common statistics. We observed, that on average 80% to 90% equal decisions are drawn when a switch happens.

Fading Memory: Using fading memory for all parameter, is a compromise of the former two mentioned approaches. This is discussed in more detail in Section 4.3.

Gradient Feature Selection: Liu and Tu [77] propose a gradient feature selection mechanism for on-line boosting. Hence, they present an unified objective for feature selection and weak classifier updating. See Section 4.5 for more details.

3.3 Illustrative Experiment

In order to demonstrate the proposed algorithm, we apply it on a simple toy example. For that purpose we consider the two-dimensional XOR problem. The goal is to train a boosted classifier, which distinguishes between the positive and negative labeled samples $\mathbf{x} = (x_1, x_2)^T \in \mathbb{R}^2$.

In order to perform feature selection, a selector holds a set of features which are possible projections of the data on a line. We encode it by the normalized row-vector $\mathbf{f} = (f_{x1}, f_{x2})$, which projects the point $\mathbf{x}$ to the feature value by the inner-product $\mathbf{f} \cdot \mathbf{x}$. A weak classifier is trained by estimating the median of the positive m^+ and negative m^- class, respectively. We achieve a hypothesis as simple decision stump and hence we get for a feature j

$$h_j(\mathbf{x}) = p_j \text{sign}(\mathbf{f}_j \cdot \mathbf{x} - \theta_j), \tag{3.9}$$

where $\theta_j := 0.5|m^+ + m^-|$ is the threshold and $p_j := \text{sign}(m^+ - m^-)$ the parity of the decision. It can be shown easily, that it is not possible the separate the two classes with any hypothesis from this hypothesis class. Thus, we apply *Off-line Boosting for Feature Selection* as well as our proposed on-line counterpart.

Figure 3.2 depicts the result for training four selectors each with 8 weak classifiers. The weak classifiers are equally distributed to cover the angle range from 0 to π and thus can describe all possible lines at that angles. For on-line training 250 randomly

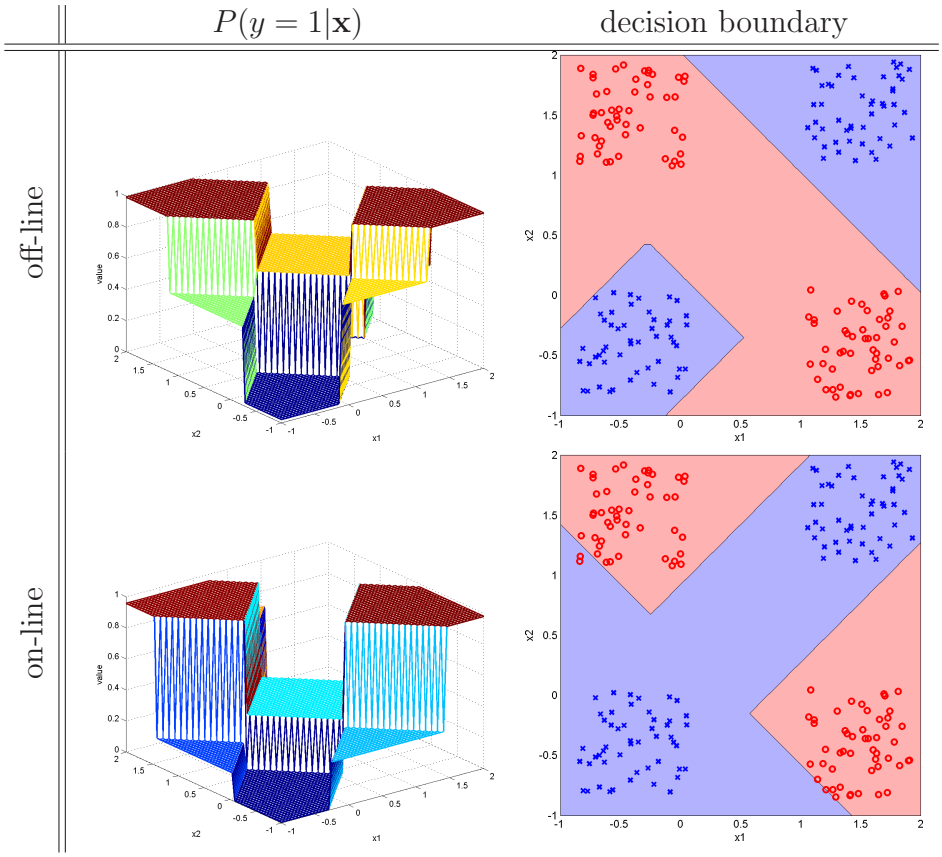


Figure 3.2: Final strong classifier achieved by using off-line boosting (first row) and by the on-line version after 250 updates (second row).

selected points are drawn from the dataset. Qualitatively the results are equivalent. In the following we take a closer look.

The first two rows of Figure 3.3 shows the weak classifiers decision boundaries for the off-line and the on-line approach. Details are depicted in the following two rows, whereas the third row shows the chosen features (the angle of the projection) within each of the four selectors. The index on the horizontal axes shows the feature and the height of the bars the value of the voting weight α. As can be seen the red (on-line) and blue (off-line) bars are similar. The learned threshold of the weak classifiers (position of the decision boundary on the projection) are shown in the last row. Dotted lines are those obtained by the off-line algorithm (blue: median m^- for the negative class; red: median m^+ for the positive class; black for the threshold θ).

The convergence of the selection process is shown in Figure 3.4 and verifies the the-

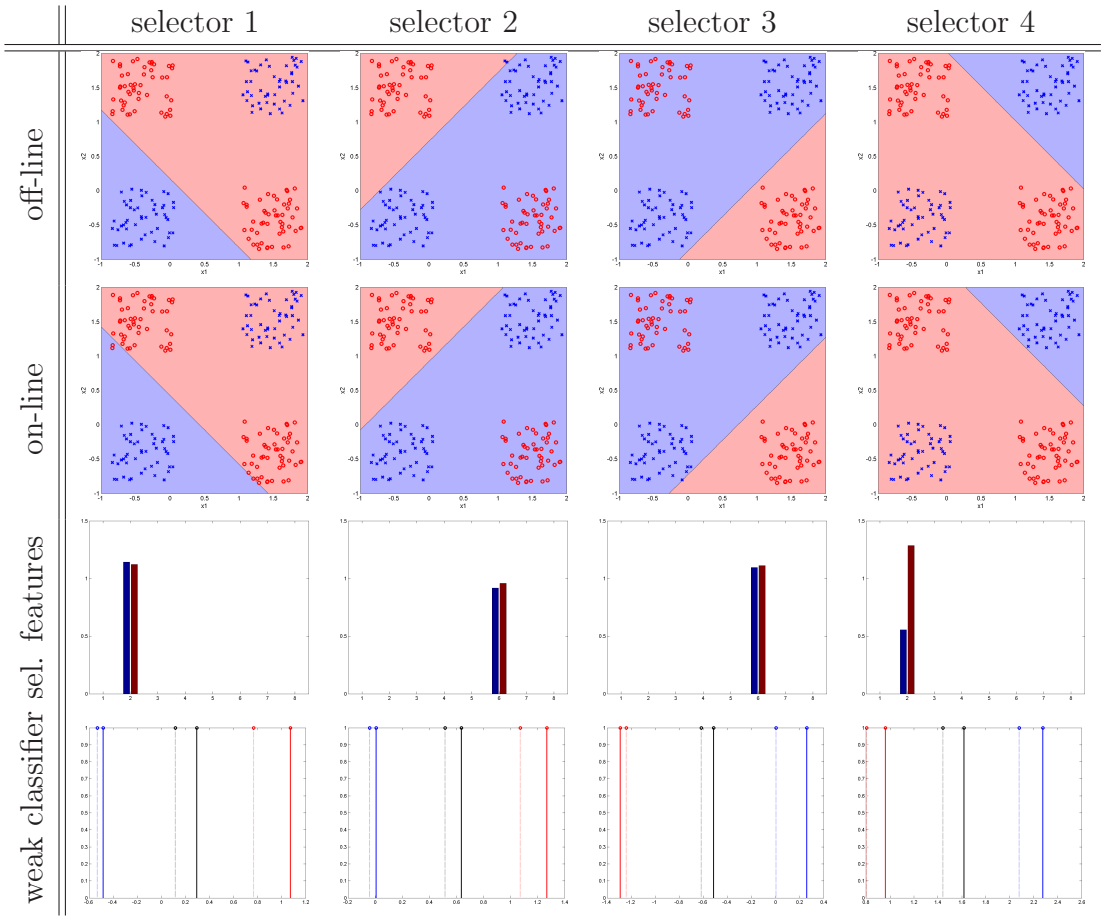
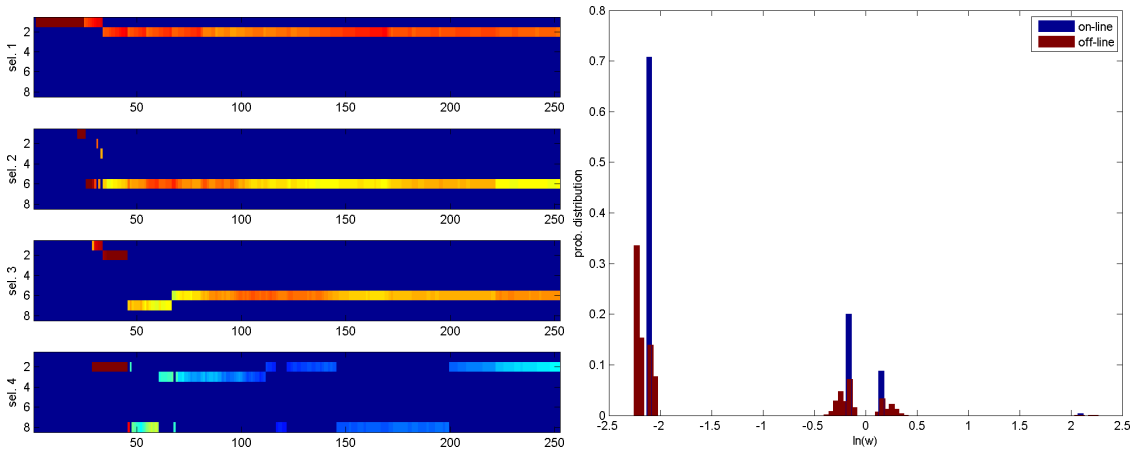


Figure 3.3: Comparison of the chosen features by the selectors using the *Off-line* and *On-line Boosting for Feature selection*. The decision boundary of the weak classifiers (first two rows) are similar, which is shown in detail by the selected features (third row; the height of the bar corresponds to α), and by the learned thresholds (last row).



(a) feature switches (b) switching noise for the 4-th selector

Figure 3.4: Feature switches within the four selectors (a). As shown by the theory, if the data is sampled according to one fixed distribution the convergence of the selected weak classifier (rows) as well as its assigned voting weight (encoded by the color) converges sequentially, beginning with the first selector. Before the selectors has converged (*i.e.*, stable selection of one feature and its corresponding voting-weight) noise is introduced in the weight distribution. In comparison to the off-line version the distribution is spread out (b).

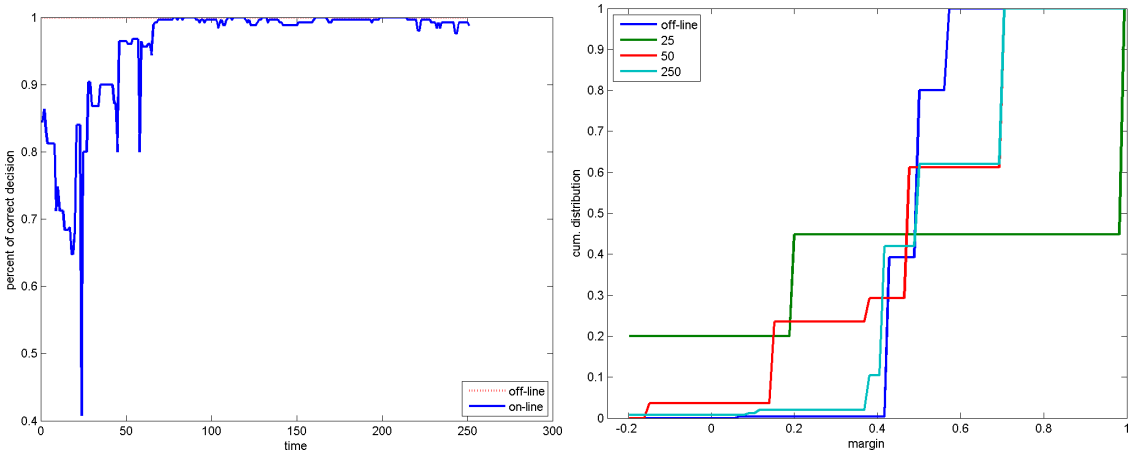
oretical results. First, the first weak classifier has to converge. Thus, the estimated error will converge and hence the selector selects the same feature as the off-line algorithm. Afterwards this is going to happen for the next selectors and so on. Colors in the plots show the value of the voting weight α.

Figure 3.5 (a) finally shows the training error on the dataset plotted over time. In addition, the cumulative margin distribution is depicted in Figure 3.5 (b) for four different times. The margin [112] is defined as

$$\text{margin}(\mathbf{x}) := \frac{y \sum_{n=1}^{N} \alpha_n h_n(\mathbf{x})}{\sum_{n=1}^{N} \alpha_n} = \frac{yH(\mathbf{x})}{\sum_{n=1}^{N} \alpha_n} \tag{3.10}$$

and thus is positive if the correction is correct and negative otherwise. Boosting maximizes the minimal margin [102]. Both, the training error and the margin distribution converges to the off-line counterpart[2]. To sum up, the proposed algorithm scans the whole feature set and provides at any time a subset that is as good as possible. In addition, we also get a weight for each selected feature, which allows to combine them into a strong classifier.

[2]Sometimes it happens that the on-line version beets the off-line version. This can be explained, since the off-line boosting is a greedy approach and thus it do not achieve the optimal solution.



(a) performance (b) cumulative margin

Figure 3.5: Training error of the on-line approach potted over time, *i.e.*, number of updates (a). Further, the cumulative margin distribution (b), shown at four times, converges to that one achieved by the off-line algorithm.

3.4 Image Features

In this section we describe how the proposed *On-line Boosting for Feature Selection* algorithm can be applied to compute vision applications. Similar as in the off-line case (see Section 2.4), we define (simple) image features, which corresponds to weak classifiers. In addition, to the classical HAAR-like features [130] we use the following two types:

Orientation histograms: First, image filtering is done to obtain a gradient image. We use the SOBEL-filtering, with kernels for horizontal and vertical edges. A magnitude weighted histogram over the gradient angles (directions) is built to represent the underlying rectangular patch. In particular, we use a 8 bin orientation histogram with equidistant bin size. The basic idea is to describe the appearance of an object part by the local gradient information. This is used in many other approaches, *e.g.*, [15, 68, 78] and also shows relations to the human visual system [20].

Local Binary Patterns (LBP): We use a simple version of LBP [85], especially a four-neighborhood (*i.e.*, $2^4 = 16$ patterns) as a 16 bin histogram feature (similar to [143]). This feature is a texture descriptor which captures the statistic of normalized pixel values in a local neighborhood. The LBP-value

of a 3×3 image patch $\mathbf{x}$ is calculated as follows

$$LBP(\mathbf{x}) = \sum_{i=0}^{3} s(x_i - x_{center}) \cdot 2^i \qquad \text{with} \qquad s(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}, \quad (3.11)$$

where x_i are the positions around the center pixel (x_{center}) using fourth neighborhood. The final representation is a histogram of the LBP values obtained by shifting the 3×3 patch in a specified rectangular region.

Note, that the computation of all feature types can be done very efficiently using integral images [130]. This idea easily can be adapted to represent histograms: for each bin one integral image is built separately [40, 100]. These features allow for describing a large variety of objects. However, one can think of combining more of such (local) features (*e.g.*, color feature or covariance features) and also include global features, like in [144].

Similar to the off-line case each weak classifier corresponds to a feature. The only difference is that we have to implement an on-line learning algorithm for the weak classifiers. In general, any on-line learning algorithm can be used which builds a weak classifier h_j for a feature j, where $f_j(\mathbf{x})$ evaluates this feature on the image $\mathbf{x}$. In fact, depended on the feature type we used the following methods.

Haar-like Features: As hypotheses for the classical HAAR-like wavelets we use either a simple threshold

$$h_j(\mathbf{x}) = p_j \text{sign}(f_j(\mathbf{x}) - \theta_j), \quad (3.12)$$

$$\text{where} \qquad \theta_j = 0.5|\mu^+ + \mu^-| \qquad \text{and} \qquad p_j = \text{sign}(\mu^+ - \mu^-) \quad (3.13)$$

or a Bayesian decision criterion, based on the estimated GAUSSIAN probability density function $g(\mathbf{x}|\mu, \sigma)$:

$$\begin{aligned} h_j(\mathbf{x}) &= \text{sign}(P(1|f_j(\mathbf{x})) - P(-1|f_j(\mathbf{x}))) \approx \\ &\approx \text{sign}(g(f_j(\mathbf{x}|\mu^+, \sigma^+) - g(f_j(\mathbf{x})|\mu^-, \sigma^-)). \end{aligned} \quad (3.14)$$

The principle is depicted in Figure 3.6.

Histogram Features: For the histogram based features (orientation histogram and LBP), we use nearest neighbor learning with a distance function D (in fact, we use the EUCLIDEAN distance)

$$h_j(\mathbf{x}) = \text{sign}(D(f_j(\mathbf{x}), \mathbf{p_j}) - D(f_j(\mathbf{x}), \mathbf{n_j})), \quad (3.15)$$

where the cluster centers for positive $\mathbf{p_j}$ and negative $\mathbf{n_j}$ samples are learned. In fact, we estimate two GAUSSIAN-distributions for each bin, one

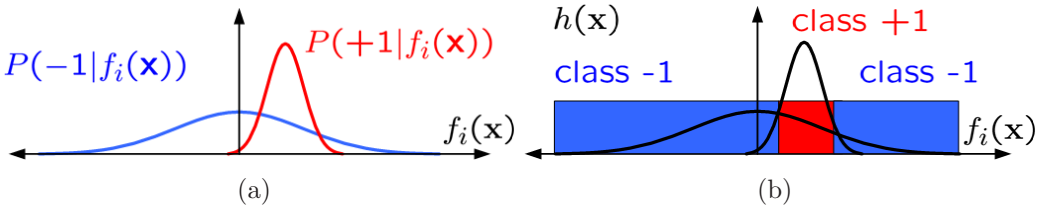


Figure 3.6: Estimated distributions for positive and negative samples for a feature f_j (a), which are used to create a weak classifier (b).

for the positive and one for the negative class. Hence, we end up with $\mathbf{p_j} = ((\mu_1^+, \sigma_1^+), \ldots, (\mu_8^+, \sigma_8^+))$ and $\mathbf{n_j} = ((\mu_1^-, \sigma_1^-), \ldots, (\mu_8^-, \sigma_8^-))$.

The remaining task is to estimate the means and variances of the GAUSSIAN-distributions, which are then used to build the hypothesis. In other words, the values μ^+, μ^-, σ^+, and σ^- have to be determined in an on-line manner. We present two very simple approaches, based on recursive techniques. In practice both are efficient and achieve similar performance.

Kalman-Filtering: The KALMAN-filter is a widely used recursive technique for tracking linear dynamical systems under GAUSSIAN noise [135]. The properties of the state space model

$$\mathbf{z_t} = \mathbf{A} \cdot \mathbf{z_{t-1}} + \mathbf{B} \cdot \mathbf{u_{t-1}} + \mathbf{w_{t-1}} \tag{3.16}$$
$$\mathbf{y_t} = \mathbf{C} \cdot \mathbf{z_t} + \mathbf{v_t}; \tag{3.17}$$

are determined by the matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. $\mathbf{u_t}$ is the system input, $\mathbf{w_t} \sim \mathcal{N}(0, \mathbf{Q})$ and $\mathbf{v_t} \sim \mathcal{N}(0, \mathbf{R})$ are uncorrelated random noise processes with covariance $\mathbf{Q}$ and $\mathbf{R}$, respectively. $\mathbf{z_t}$ are the state variables and $\mathbf{y_t}$ the observed system output.

In order to build a hypothesis h_n as above, we incrementally estimate the means (μ^+, μ^-) and variances (σ^+, σ^-) by the KALMAN-filtering approach. Thus, we build a simple state space model for estimation the (constant) mean and achieve $\mu_t = \mu_{t-1} + v_t$ and $\sigma_t^2 = \sigma_{t-1}^2 + v_t$ for the variance. Therefore, we set $\mathbf{A} = 1$, $\mathbf{B} = 0$, $\mathbf{C} = 1$, $\mathbf{Q} = 0$, $\mathbf{R} = 0.01$ (which indeed are now scalar values) and furthermore the initial state $P_0 = 1000$, $\mu_0 = 0$ and $\sigma_0^2 = 0$. The following update equations for the adaptive estimation can be derived:

$$K_t = P_{t-1}/(P_{t-1} + R) \tag{3.18}$$
$$\mu_t = K_t \cdot f_j(\mathbf{x}) + (1 - K_t) \cdot \mu_{t-1} \tag{3.19}$$
$$\sigma_t^2 = K_t \cdot (f_j(\mathbf{x}) - \mu_t)^2 + (1 - K_t) \cdot \sigma_{t-1}^2 \tag{3.20}$$
$$P_t = (1 - K_t) \cdot P_{t-1}. \tag{3.21}$$

Approximated Median: Following the approximated median rule [81], which increments a running estimate of the median m_t by one if the input is larger than the estimate, and decrements by one if the opposite is true. More formally,

$$m_t = \begin{cases} m_{t-1} & f_j(\mathbf{x}) = m_{t-1} \\ m_{t-1} - K_t & f_j(\mathbf{x}) < m_{t-1} \\ m_{t-1} + K_t & f_j(\mathbf{x}) > m_{t-1} \end{cases} \quad \text{where} \quad K_t = \frac{1}{2\sqrt{t}} \tag{3.22}$$

is the learning rate, which is motivated by a statistical view. If the number of samples n is large enough (*i.e.*, $n \geq 30$) the central limit theorem can be applied to obtain a 95% confidence intervall. The true mean is then within the bound $\frac{2\sigma}{\sqrt{n}}$ [62], where σ^2 is the variance of the data distribution. In fact we set afterwards $\mu^+ = m^+$ and $\mu^- = m^-$. The variance is not estimated und hence only the threshold hypotheses for the HAAR-like wavlets are used.

Chapter 4

Extensions and Discussions

Several modifications and approximations have been made in order to make the *On-line Boosting for Feature Selection*, proposed in the last section, more applicable for practical applications[1]. In the following sections we do this with respect to (i) speeding up the training process and allowing to explore a larger feature pool; (ii) speeding up the evaluation process and how to set the complexity of the classifier (*i.e.*, the number of weak classifiers); (iii) adaptivity; and (iv) including prior knowledge. Furthermore, we show extensions, which have been developed by other researchers.

[1] The reader is pointed to the applications in Part II of this thesis for experiments.

4.1 Speeding up the Training Process

Assuming a strong classifier with N selectors, which determines the hypothesis class. In order to evaluate a strong classifier, the N selected features (weak classifiers) have to be evaluated. In contrast, for updating the strong classifier each weak classifier within the selectors have to be updated. Both speed and memory are $\mathcal{O}(MN)$ assuming that all N selectors have the same number of M weak classifiers. The main computational effort is spent for updating the weak classifiers, which depends on the learning algorithm and the time to calculate the feature value. The time consumed by our approach is negligible.

On the one hand, for practical applications (*e.g.*, tracking), which have to run in real time we are limited by the feature pool. But on the other hand, the algorithm has to provide a "sufficient enough" feature pool in order to achieve a good performing classifier. In the following we propose two extension to overcome these problems.

4.1.1 Exploring a Large Feature Pool

The whole feature pool $\mathcal{F}$ contains an enormous amount of possible features because of the highly over complete representation (each feature prototype can appear at different position and scale). Given a 24×24 image patch $45,396$ classical HAAR-like features [130] are possible. When increasing the feature pool, *e.g.*, [74] uses rotated features as well, this value increases to over $100,000$. Thus, for practical, real time applications, it is nearly impossible to check all features.

In the off-line case at each boosting iteration a new weak classifier is created and thus a feature from the feature pool $\mathcal{F}$ is selected and added it to the ensemble. In the on-line case each selector has its local feature set $\mathcal{F}_n$. Since we know how good the individual weak classifiers (features) perform, we propose to adapt the local feature pool $\mathcal{F}_n \subseteq \mathcal{F}$ of each selector by replacing the worst feature (*i.e.*, that with the highest error) with a randomly chosen new one from the feature pool $\mathcal{F}$. Note, in order to establish a sufficient meaningful statistic the feature has to be given a certain number of positive and negative updates.

If the process is running for a long time, a lot of features are processed and evaluated but still only a small number of features is sufficient for updating the selector. Since in the on-line case learning continues, the model will continuously improve by exploring more features and training data. One can further think of performing an off-line pre-computation in order to select "good" (task specific) features $\mathcal{F}_{task} \subseteq \mathcal{F}$. The on-line algorithm uses then $\mathcal{F}_{task}$ to find a solution for the learning problem.

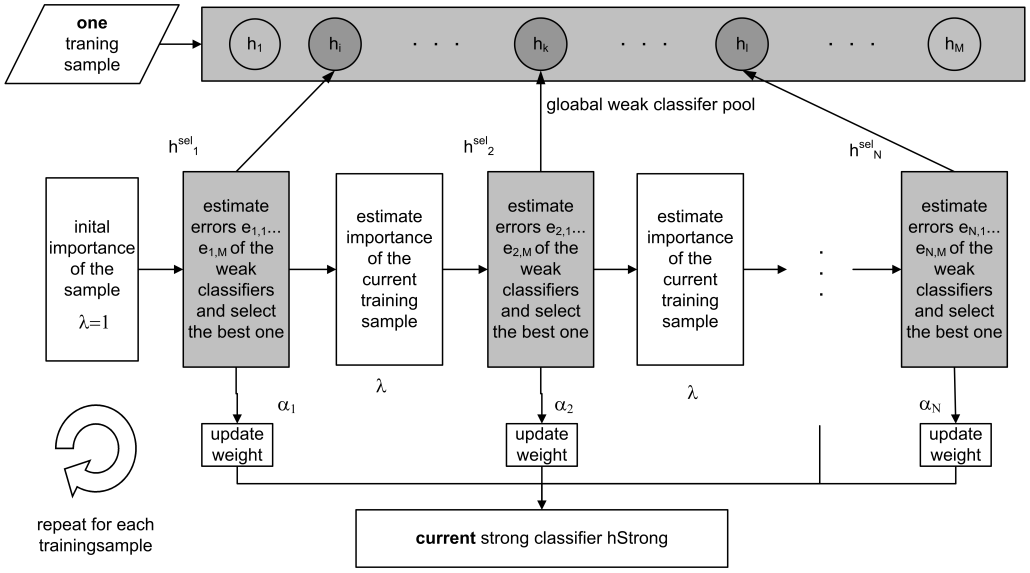


Figure 4.1: Principle of *On-line Boosting for Feature Selection* using a shared feature pool.

4.1.2 Direct Feature Selection[2]

In order to speed up the training process, we propose to use a single "global weak classifier" pool (similar to [140]). This pool is shared by all selectors instead of single pools for each of them as depicted in Figure 4.1. The advantage of this modification is that for each sample, all weak classifiers need to be updated only once. Then the selectors sequentially switch to the best weak classifier with respect to the current estimated λ and the importance weight is passed on to the next selector. This procedure is repeated until all selectors are updated.

Another advantage is that more features can be considered for one selector. The disadvantage of this approach is, that the importance of the sample λ is not taken into account in order to train the weak classifiers. However, our experience in various experiments (at least by using the image features defined in Section 3.4) shows that the feature selection is more dominant and thus it is a good approximation.

In order to increase the diversity of the weak classifiers and to allow changes in the environment, the worst weak classifier of the shared feature pool is replaced with a new randomly chosen one (same as discussed in the previous section).

[2]Adapted from H. Grabner, M. Grabner, and H. Bischof. Real-Time Tracking via On-line Boosting. *In Proceedings British Machine Vision Conference (BMVC)*, pages 47-56, 2006. [31]

4.2 Sequential On-line Boosting Classifier[3]

Yet, there remain two important unsolved problems:

(i) optimization of the classifier evaluation speed, and

(ii) automatic determination of the classifier complexity.

We show, how to solve both of these problems. To overcome the first point, following the idea of Viola and Jones [130], who proposed a cascaded *AdaBoost* classifier, other authors tried to improve the evaluation speed of the classifier by *e.g.*, introducing *FloatBoost* (weak classifiers can be also removed from the strong classifier) [71], a vector boosting [52], or by using sequential decision making theory [120]. All these methods work in an off-line manner, meaning all the training samples are given in advance and the classifier is kept fixed after being trained. Recently, Wu and Nevatia [139] investigated to use the cascade approach in the on-line boosting framework. 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 classifiers to be given in advance. 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.

We introduce Wald sequential decision theory in the on-line framework inspired by the *WaldBoost* algorithm [120]. Our approach overcomes both problems of classifier speed and complexity optimization in the on-line setting.

4.2.1 *WaldBoost*

The *WaldBoost* algorithm [120] is an off-line training algorithm, which combines the *AdaBoost* training and Wald's sequential decision theory [134]. The *WaldBoost* training goal is to minimize the training error as in *AdaBoost* but at the same time to minimize the evaluation time of the classifier. More formally, *WaldBoost* finds a quasi-optimal sequential decision strategy S^* such that

$$S^* = \arg\min_S \bar{T}_S \qquad \text{s.t.} \quad \beta_S \leq \beta, \quad \alpha_S \leq \alpha \tag{4.1}$$

[3] Adapted from `H. Grabner, J. Sochman, H. Bischof, and J. Matas. Training Sequential On-line Boosting Classifier for Visual Tracking.` *`In Proceedings International Conference on Pattern Recognition (ICPR)`*, `2008.` [38]

for specified α and β, where $\bar{T}_S$ is the average time-to-decision, α_S is the false negative and β_S the false positive rate of the sequential strategy S. For a two class problem S is a sequence of decision functions $S = S_1, S_2, \ldots$ where $S_n \colon (x_1, \ldots, x_n) \to \{-1, +1, \sharp\}$. The strategy S takes one measurement x_i at a time and in step n makes a decision S_n based on measurements $x_1, \ldots, x_n$. The '$\sharp$' sign stands for a "continue" (do not decide yet) decision. If a decision is '$\sharp$', x_{n+1} is measured and S_{n+1} is evaluated. Otherwise, the output of S is the class returned by S_n.

The *WaldBoost* algorithm uses outputs of weak classifiers found by *AdaBoost* as measurements (*i.e.*, it uses *AdaBoost* as a measurement selector). A *WaldBoost* classifier then becomes

$$H_n(\mathbf{x}) = \begin{cases} +1 & f_n(\mathbf{x}) \geq \theta_B^{(n)} \\ -1 & f_n(\mathbf{x}) \leq \theta_A^{(n)} \\ \text{continue} & \theta_A^{(n)} < f_n(x) < \theta_B^{(n)} \end{cases}, \tag{4.2}$$

where $f_n(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i h_i(\mathbf{x})$. The goal of training is to find the proper weak classifiers h_n and the thresholds $\theta_A^{(n)}$ and $\theta_B^{(n)}$. The thresholds can be computed given the classifier response function $f_n(\mathbf{x})$. From the Wald theory, we are looking for two thresholds on the likelihood ratio R_n. Unfortunately, R_n is difficult to estimate due to the high dimensionality of both probability densities. Instead, a projection to an one dimensional space is used [120]

$$R_n(x) \cong \hat{R}_n(\mathbf{x}) = \frac{p(f_n(\mathbf{x})|y = -1)}{p(f_n(\mathbf{x})|y = +1)} \,. \tag{4.3}$$

However, the training process rebuilds repeatedly the training and the validation set using bootstrapping (*i.e.*, already decidable training samples are replaced by those which could not be decided yet). As the validation set used for estimating the thresholds changes, direct density estimation gives $p(f_n(\mathbf{x})|y = C, \to n)$ where $C \in \{-1, +1\}$ and $\to n$ stands for the condition that the sample has not been decided up to training step n, instead of desired $p(f_n(\mathbf{x})|y = C)$. Using Bayes formula we get

$$\hat{R}_n(x) = \frac{p(f_n(\mathbf{x})|y = -1, \to n)p(\to n| + 1)}{p(f_n(\mathbf{x})|y = +1, \to n)p(\to n| - 1)} \tag{4.4}$$

which leads to estimation of the likelihood ratio taking into account the bootstrapping.

From Wald's theory the thresholds $\theta_A^{(n)}$ and $\theta_B^{(n)}$ are estimated using $\hat{R}_n$ by finding thresholds for which $\hat{R}_n \geq A$ or $\hat{R}_n \leq B$ respectively, where $A = (1 - \beta)/\alpha$ and $B = \beta/(1 - \alpha)$. A practical way of estimating the thresholds is to look for such values of $f_n(x)$ for which the ratio of negative and positive samples multiplied by

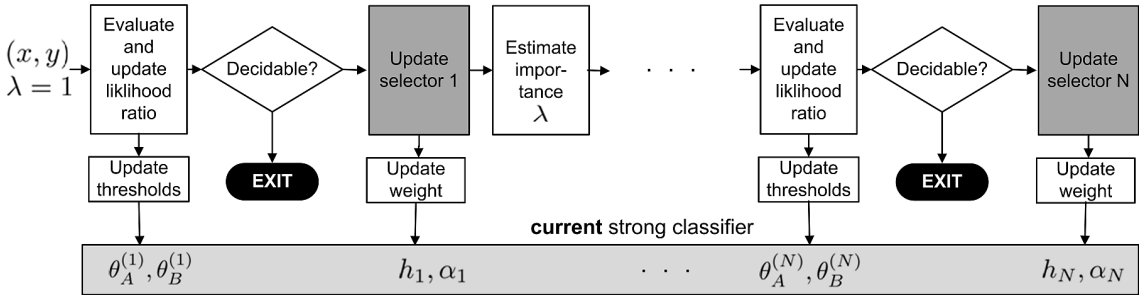


Figure 4.2: On-line *WaldBoost* classifier.

the correction factor which takes the discarded samples into account, fulfills the conditions.

4.2.2 On-line *WaldBoost*

In order to find the Wald thresholds $\theta_A^{(n)}$ and $\theta_B^{(n)}$ the likelihood ratio $\hat{R}(x)$ from Equation 4.4 has to be estimated. In the off-line training the statistics are computed on an independent validation dataset. 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 Wald statistics before it is used for training the strong classifier. The probabilities $p(\rightarrow n|C)$ can be estimated by computing the portion of samples seen so far and not decided until n-th selector. The densities $p(f_n(\mathbf{x})|y = C, \rightarrow n)$ are estimated from the samples which are not decided until the n-th selector only. In our implementation they are approximated by Gaussian-distributions. Given these probabilities and α and β parameters the thresholds $\theta_A^{(n)}$ and $\theta_B^{(n)}$ are estimated as in die previous section. The overall principle is depicted in Figure 4.2.

However, since a feature-switch in selector k causes a wrong estimate of the statistics of subsequent selectors, the statistics are reset where selectors $n \geq k$, *i.e.*, $p(f_n(x)|y = C, \rightarrow n)$ is set to the uniform distribution and $p(\rightarrow n|C) = 0.5$ 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. Experiments on a visual object tracking task (see Section **??**) show that the method is able to automatically adapt the classifier complexity to changing problem difficulty.

4.3 Time Dependent On-line Boosting[4]

A main limitation in the on-line boosting approach up to now is that it assumes that the samples are drawn from one fixed distribution. In this extension, we want to focus on the adaptivity, where we assume a smart changing of the distributions [127].

Concerning robust adaptiveness, the original approach has several limitations: First, the sample weights contribute forever to the entire model statistics and are only unlearned (*i.e.*, getting less important) by updating the system with new ones. Hence, there is yet no control how fast information "fades" away and new one is gained. Second, samples with high errors get a much higher weight assigned than low-error samples. When sampling always from the same (static) distribution this is perfect, since boosting focuses on the difficult examples. Nevertheless, this makes the system extremely sensitive to label noise and, especially, when dealing with an adaptive learning problem, this assumption makes no sense anymore.

To overcome these limitations we propose the following modifications: The first change to on-line boosting considers the basic assumption that the examples are all drawn from a fixed distribution. This has to be taken into account in all dynamic elements of the system, especially:

Weak Classifier: The update rule for estimating the probability density functions (replacing the KALMAN-filtering like updates of mean and variance in Equation 3.12) of the weak classifier.

Errors: To estimate the errors, *i.e.*, both λ^{corr} and λ^{wrong} in Equation 3.6. In fact, at each update step we first multiply these values by a factor $K < 1$ before updating with the new importance λ.

As we aspire fading memory we propose to use exponential forgetting of the examples over time. Therefore, we define the following update rule for the estimation of the value $\hat{z}_t$ using the previous value $\hat{z}_{t-1}$ and a new measurement z

$$\hat{z}_t = K\hat{z}_{t-1} + (1-K)z \qquad \text{where} \qquad K = \sqrt[\Delta t]{0.5} \tag{4.5}$$

gives the factor of how much previous information should be kept. In particular, it determines that half the information is kept in the time interval Δt.

The second change allows that just the time is important and not the sample itself. Furthermore, it limits the effect of label noise. Each new incoming sample $(\mathbf{x}_t, y_t)$ for the first selector is initialized with the importance $\lambda_0 = 1$. This means if it is

[4] Adapted from H. Grabner, C. Leistner, and H. Bischof. Time Dependent On-line Boosting for Robust Backgroundmodeling. *In Proceedings Proceedings International Conference on Computer Vision Theory and Applications*, 2007. [32]

well predictable its importance decreases by propagating through all the selectors otherwise increases. Assuming label noise a single noisy example can get very high importance and can change the entire model statistics rapidly. Therefore, we propose to keep the importance of the samples at the end of the ensemble constant. We first propagate the example through the set of selectors and obtain the value of λ_N without doing an update. The actual update is done with the initial value set to $\lambda_0 = \frac{1}{\lambda_N}$ which clearly results in keeping it at one at the last selector. This simple modification ensures trusting the model more than the examples which means that the system, inherently, has some kind of outlier detection implemented. Please note, that this modification does not change the overall boosting process, it rather gives the example a prior importance.

Furthermore, the fading memory limits the switching problem (mentioned in Section 3.2.1), since old information fades aways. Additionally, we introduce a soft-selector, in order to limit the hard switching effect within the selectors of the cost of computational complexity. In contrast to taking the best weak classifier (*i.e.*, the one with the smallest error) for each selector it uses the information of all weak classifiers combined. Although every arbitrary classifier fusion rule might be applied, we chose to use the simple sum-rule which in practice yields good results [59]. As a result, all weak classifiers, the errors and therefore the importance weight λ as well as the voting α_i are changing continuously. Finally, the prediction of a selector is now

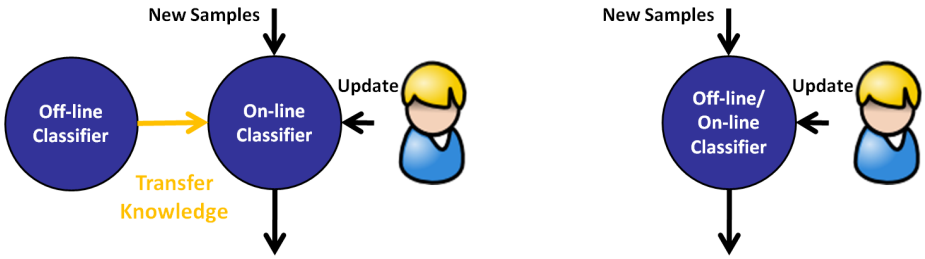
$$h^{sel}(\mathbf{x}) = \text{sign}\Big(\sum_{m=1}^{M} \alpha_i h_i\Big). \tag{4.6}$$

4.4 Including Prior Knowledge[5]

The question we focus in this section, is how prior knowledge can be included into the on-line learning process. This has the advantage that the on-line learning does not have to start from scratch and hence, fewer examples are required to achieve good performance. In other words, the on-line learning process adapts the knowledge to the particular task. Assuming that the prior knowledge is given as an off-line classifier, the goal is to transfer this knowledge to an on-line classifier, which can then be improved.

We assume that a prior (off-line) classifier, which was built on a distribution $P' : \mathcal{X}' \times \mathcal{Y}'$ different from the novel target distribution $P : \mathcal{X} \times \mathcal{Y}$. Nevertheless, we assume that both distributions are quite similar $P' \sim P$. Thus, we can benefit from

[5] Adapted from P. Roth, H. Grabner, C. Leistner, M. Winter, and H. Bischof. Interactive Learning a Person Detector: Fewer Clicks - Less Frustration. *In Proceedings Workshop of the Austrian Association for Pattern Recognition*, 2008. [105]



(a) knowledge transfer (b) direct retraining

Figure 4.3: Knowledge transfer: (a) a new on-line classifier is built using the information, that is already captured by an off-line classifier; (b) proposed approach – an off-line trained classifier is directly re-trained.

the prior classification (also denoted as "transfer" learning). However, the main question is, which part of prior information can be re-used and how this should be done.

4.4.1 Classifier Transfer

The simplest way to make use of prior knowledge is to transfer pre-learned knowledge (prior) via labels to an on-line classifier (see Figure 4.3 (a)). Precisely, the on-line classifier is updated with samples $(\mathbf{x}, \hat{y})$ of the novel scene, where the labels $\hat{y} = H_{off}(\mathbf{x})$ are provided by the off-line classifier.

More sophisticated ways of knowledge transfer may be applied. For instance, similar to [113] we can transfer the information from an off-line classifier H_{off} to an on-line classifier H_{on} by applying H_{off} as the first weak classifier. In fact, the succeeding weak classifiers in the on-line ensemble compensate the errors of the prior off-line classifier. More formally, the thus obtained combined classifier is a weighted sum of the off-line and the on-line classifier:

$$H(\mathbf{x}) = \alpha_{off}(H_{off}(\mathbf{x}) - \theta_{off}) + H_{on}(\mathbf{x}). \tag{4.7}$$

Since H_{off} is interpreted as first weak classifier a threshold θ_{off} is set according to the response of the new distribution. Additionally, the weight α_{off} is calculated using the standard boosting formulation with error $\hat{e}_{off}$, which is the estimated error for the off-line classifier on the new scene. Thus, α_{off} automatically determines the importance of the off-line classifier for the novel distribution as part of the on-line classifier. As a drawback, the complexity of the on-line classifier has to be large if the off-line classifier has an error near 50% (*i.e.*, the two distributions are very "different"). This, however, yields to the common problem that many training samples have to be provided again.

4.4.2 Direct Re-training

To avoid the drawbacks of the previously discussed methods for knowledge transfer, we propose to directly update the off-line trained classifier in an on-line manner (see Figure 4.3 (b)). For that purpose, we have to ensure that all statistics, that are necessary for on-line updating, are stored during the off-line training phase. This can be done straightforward for all components:

Weak classifier: In order to build a weak hypothesis $h_n : \mathcal{X} \rightarrow \{-1, 1\}$ corresponding to an image feature f_n we apply a learning algorithm. Precisely, we estimate the distributions $P(y = 1|f_n(\mathbf{x}))$ and $P(y = -1|f_n(\mathbf{x}))$ for positive and negative samples, respectively. Assuming that positive and negative feature values follow GAUSSIAN-distributions, we can calculate the mean and the variance from all off-line samples. These parameters can then easily be adjusted during the on-line learning stage [32] to re-calculate $h_n(\mathbf{x})$ as described in the previous section.

Errors: The error of the weak classifier is used to select the best weak classifier within a selector to calculate the voting weight α and to update the importance λ. In the off-line case the error depends on the weights w_i of the training samples, that were classified correctly and incorrectly. These values have to be saved as λ^{corr} and λ^{wrong}, which can then be updated by the importance λ in the on-line case. Thus, by using Equation 3.6 the estimated error can be re-calculated.

These modifications of the off-line learning process allow us to on-line re-train an off-line trained classifier later on. Thus, we can retain the information captured during the off-line training and we can still adapt an existing classifier to a new specific scene. The drawback of these approach is that the off-line classifiers have to be trained in that way, *i.e.*, the training data has to be given. By using a classifier transfer strategy (as briefly discussed in the last section) one can make use of any a-priory given classifier.

4.5 Other Extensions and Applications

The following extensions were not developed by myself or our team. They are listed here for completeness. Note, only those papers are mentioned, that extend or directly make use of the approach and not only cite it[6] as related work.

[6]The original paper, `H. Grabner, and H. Bischof. On-line Boosting and Vision. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 260-268, 2006. [30]`, has 32 cites on Google Scholar up to now (August 18, 2008).

Real On-line Boosting: Xj *et al.* [141] replace the discrete *AdaBoost* with the real valued version [114] and claim a faster convergence. They demonstrate their algorithm on the task of visual tracking. Furthermore, a particle filter and an extended set of features are used in the application.

Asymmetric On-line Boosting: Pham *et al.* [97] extend the approach in order to cope better with highly asymmetric problems like it was done by Viola and Jones [131] in the off-line case. Asymmetric learning is important, especially for learning object detectors. They show an increase in accuracy (0-10%) as well as in speed compared to our original method.

Gradient Feature Selection: Liu and Tu [77] propose a gradient feature selection mechanism for on-line boosting. The approach iteratively updates the features (orientation histograms in their implementation, but not limited to it) in a gradient decent manner. Hence, they present an unified objective function for feature selection and weak classifier updating. It seems that this method naturally avoids the switching of the features within one selector. Experiments on person detection and tracking applications demonstrate the effectiveness of their approach.

Boosting Adaptive Linear Weak Classifier: Parag *et al.* [94] propose to modify the form of the weak classifiers. The classifier adapt themselves to conform with the changes over time. Their proposed method modifies the internal parameters of the base learners for the final classifier to blend with the change. They apply their algorithm for visual tracking. This is related to our work using time dependent on-line boosting (see Section 4.3).

Cascade Structure and Extended Application: Wu and Nevatia [139] build on our ideas and use their extended approach to improve part-based object detectors. In fact, they make use of other features and improve noise robustness. Furthermore, they propose to run a cascade like structure to speed up the evaluation process. This is strongly related to our work using Wald's sequential decision theory (see Section 4.2).

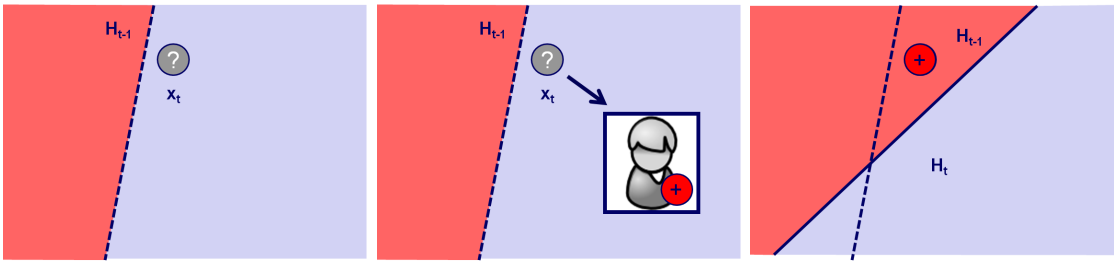
Tracking: Woddy *et al.* [137] extend our tracking approach by a local generative appearance model. The main contribution is the use of a generative model to guide the on-line feature selection progress to regions of an image which maintain a valid appearance and thus, they improved robustness to occlusions.

Tracking: Grabner *et al.* [43] integrate ensemble classifiers in optical-flow based tracking. *On-line Boosting for Feature Selection* is used in order to adapt the voting-weights of the weak classifiers ensemble, which corresponds to simple gray pixel differences. For the tracking step an affine motion model is proposed.

Part II

The Role of Supervision

In the first part of this thesis we proposed the *On-line Boosting for Feature Selection* algorithm. The algorithm works on labeled samples, *i.e.*, in a supervised manner. However, for many tasks and applications, especially in computer vision, the labels are not known. Since our learning algorithm cannot directly deal with these unlabeled examples, we have to first provide a label. This is done by a "teacher", *i.e.*, the supervision which is the main focus of this second part. Since the sample is now labeled, we can perform an update of the classifier (see Figure 5.1).



(a) unlabeled examples (b) labeling via a "teacher" (c) on-line update

Figure 5.1: In order to update the current classifier H_{t-1} with an unlabeled example $\mathbf{x}_t$(a), a "teacher" is used to label it (b). Afterwards a supervised on-line learning algorithm can be used to update the classifier (c).

Contents of Part II

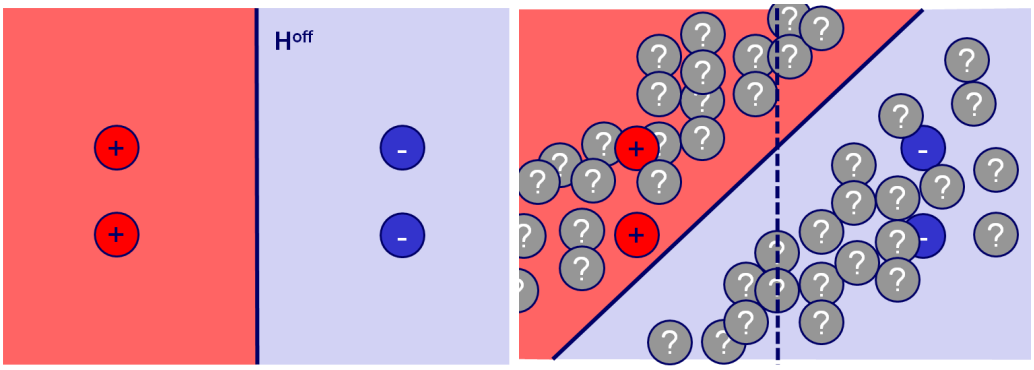
Chapter 5

The Supervision

In this chapter we briefly review how unlabeled data can be included in off-line learning and that it indeed can increase performance. Afterwards, assuming a supervised learning algorithm is given, we show different ways of doing this for the on-line setting. This sets the frame for the other chapters in this part, where the different approaches are shown on particular applications.

5.1 Semi-Supervised Learning

In general, machine learning methods can divided into the following groups, dependent on how the data or the feedback to the learner is provided. In *Supervised Learning* a training sample $(\mathbf{x}, y)$ consists of the feature $\mathbf{x}$ and the corresponding (given) label y. In *Unsupervised learning* or *Clustering* no label is provided and thus the learning algorithm is looking for interesting structures in the data, *i.e.*, group similar samples together. In *Reinforcement Learning* a reward (not the label itself) is given to the learning algorithm. The rewards depends on how good the learner has selected its action. *Semi-supervised learning* is supervised learning, where additionally unlabeled data is given. In the semi-supervised learning framework, the main question is:



(a) supervised (b) semi-supervised

Figure 5.2: In supervised learning for all samples its label is provided (a), whereas in semi-supervised learning only for a few samples the target value is given (b). The unlabeled data can help to improve classification results, *e.g.*, by enforcing that the decision border is in a low density region.

How should the unlabeled data be taken into account?

A simple example, illustrated in Figure 5.2, shows that one can indeed benefit from unlabeled data. In off-line learning quite a variety of different methods exists, where each of them make certain assumptions (see [145] for a good overview of existing methods).

Co-training [10] assumes that multiple classifiers are trained over multiple feature views of the same labeled examples. Graph-based, semi-supervised methods assume that labeled and unlabeled data is connected by a graph. The edges encode the similarity (distance) between the examples. Similar examples should have the same label. In kernel-based methods (*e.g.*, transductive support vector machines) also the margin of unlabeled examples is taken into account, *i.e.*, they serve as a smoothness or regularization term in order to emphasize decision boundaries which lie in a low density region. Generative semi-supervised methods assumes a model for the data distribution. By using the *Expectation Maximization* algorithm the unlabeled data gets assigned (pseudo) labels and afterwards the model is updated. It is important to mention that unlabeled data not always improves the performance. Especially, this is the case when the underlying assumptions are invalid, *e.g.*, as shown by the following example [145]:

> *Quite a few semi-supervised learning methods assume that the decision boundary should avoid regions with high density. If the data is generated from two heavily overlapping* Gaussian*-distributions, the decision boundary would go right through the densest region, and these methods would perform bad. The art is to design methods which at least do not decrease their performance by unlabeled data.*

5.2 On-line Learning via a Teacher

Including unlabeled data in on-line learning means to unsupervised improve a classifier, which is randomly initialized, or pre-trained with some labeled data. Since our learning algorithm (*On-line Boosting for Feature Selection*) is a supervised learning algorithm, which cannot directly use unlabeled data the main questions are:

(i) When should an update be done?

(ii) With which label should the update be done?

(iii) What happens in the case of wrong updates?

In general, such adaptive approaches usually suffers from the drifting problem. A small error can be accumulated more and more and the whole system can end in a catastrophic state (*i.e.*, regarding object detection, a classifier that was trained using many incorrect updates would yield many false positives and/or the detection rate would decrease). Grossberg [44] addresses this by the *Stability versus Plasticity Dilemma*. That is, how can one design a learning system that is plastic enough to learn new information and, at the same time, is stable enough to not forget old important information, that it has already learned.

A closely related topic is concept drift [122, 127]. A Concept drift takes place when the underlying structure of the data space or the underlying concept changes in reality. The effectiveness of any incremental learning algorithm, is its ability to capture the concept drift as the training proceeds. The following properties should considered:

Stability: The prediction accuracy of the test set should not vary widely at every incremental training step.

Improvement: There should be an improvement of the accuracy as the training goes on and the learning algorithm sees more training examples. Issues like convergence and dependency on the initial conditions have to be considered.

Recoverability: The learning method should be able to recover from its errors, *i.e.*, even if the performance drops at a certain learning step, the algorithm should be capable of recovering back to the previous best performance.

5.2.1 Approaches

Summarizing, new unlabeled data has to be robustly included into an already built model. More formally, at time t given a binary classifier H_{t-1} and an unlabeled

example $\mathbf{x}_t$. The classifier predicts a label $y_t \in \{+1, -1\}$ for $\mathbf{x}_t$ which can further be used by an "analyzer" to generate the label $\hat{y}_t$, which indeed is then used to update the classifier.

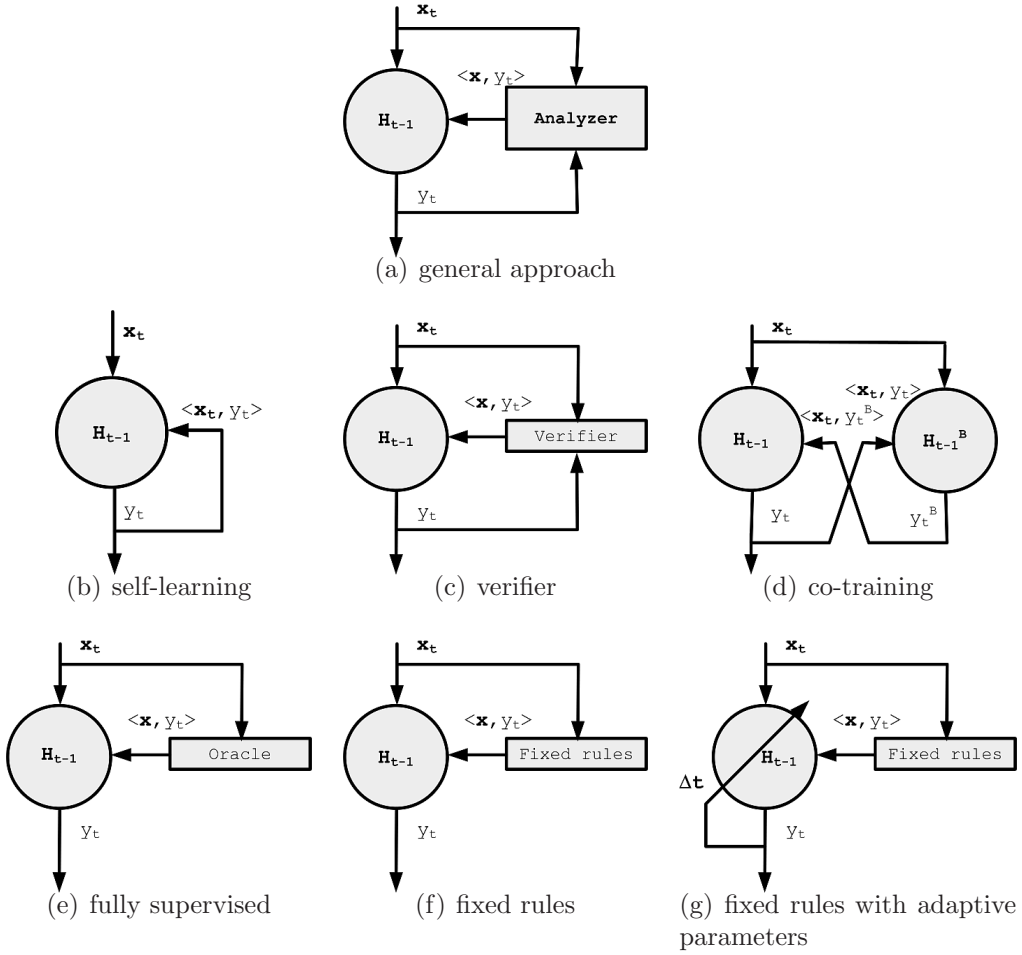


Figure 5.3: General approach for on-line updating a classifier H (a). Depending on the design of the analyzer a wide range of methods can be obtained; from supervised learning (e) (the analyzer is an oracle) to self-learning (b) where the analyzer is hot-wired. In order to limit drifting, verifiers are used (c). A special case is co-learning (d) where the labels are provided by another classifier. However, this approach has strong assumptions. By using fixed update rules (f) the classifier response $y_t = H_{t-1}(\mathbf{x}_t)$ is not taken into account at all for delivering the update. Thus, we have neither direct nor indirect feedback. This is extended in (g) where only parameters are adjusted with direct feedback.

Figure 5.3 depicts a few approaches. If no verification is given, as depicted in (b) this is named self-learning. In a self-learning framework the current classifier evaluates an input sample and predicts a label which is then directly used to update the classifier.

Hence, the classifier teaches itself by its own predictions. In general, such methods suffer from the drifting problem since there is no re-active process included. In (c) the results obtained by the classifier are verified by a "sophisticated" analyzer and if the obtained labels are confident the samples are used for updating the classifier. Nair and Clark [83] proposed to use motion cues for this purpose. Roth *et al.* [108] extended this idea and additionally applied a generative model for verification. In contrast, Wu and Nevatia [139] used local parts of the object in order verify the detections and to improve the detection results. When using another classifier as verifier this is known as co-training [10, 69]. Two classifiers are trained in parallel using different views of the data. The confident predicted labels are used to update the other classifier, respectively (d). The main drawback is the assumption that the two classifiers are statistically independent.

All of these approaches mentioned so far have a feedback from the classifier somehow back to generate the label. Thus, if the label $\hat{y}$ is wrong an update decreases the classification performance and the classifier drifts away. There are two issues involved:

(i) how many label noise is generated by the supervision, and

(ii) how robust is the classifier to tackle with that label noise.

Some approaches are more immune than others but the principle problem stays the same. Hence, in the last row of Figure 5.3 no direct feedback of the classifier is done. The knowledge of an expert is encoded into rules (*e.g.*, an other classifier teaches the on-line classifier). Thus, the labels are stable and "predictable". Maybe some internal parameters of the model will be adapted (g). Note, that this is in the common understanding not an expert system. The expert knowledge is used to train a classifier and thus, the classifier can generalize more and even outperform the teacher on particular situations.

So far only label noise (wrong updates) is discussed. However, for computer vision application another important issue has to be taken into account: label jitter, meaning a positive sample, which is not well aligned. The two possible errors are shown in Figure 5.4.

5.3 Outline

The applications for using *Off-line Boosting for Feature Selection* is more or less object detection, since all the training data has to be given in advance. By using the proposed *On-line Boosting for Feature Selection* algorithm other applications can be explored as well. We demonstrate the multifariousness of the method on such diverse

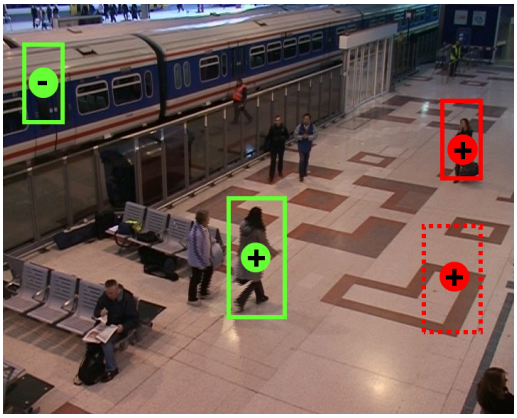


Figure 5.4: Green marked image patches corresponds to correct labeled positive and negative samples. Whereas the red marked patches show the two possible errors: *label noise*, *i.e.*, a wrong label (dotted red) and *label jitter*, *i.e.*, a misaligned positive patch (solid red).

tasks as learning complex background models, visual tracking, and improving object detectors. All approaches benefit significantly by the on-line training. However, the proposed algorithm is a fully supervised two class learning algorithm. Therefore, two issues have to be considered:

(i) each task has to be formulated as a binary classification problem, and

(ii) labels for the unlabeled samples have to be generated.

Both points are discussed for all applications. In particular, the following chapters are organized with respect to the different methods of supervision. Beginning from fully supervised learning over learning via verification to self-learning, and finally, learning using fixed update rules. Advantages and disadvantages of the approaches are discussed by means of computer vision applications.

Note: For motivation, related work, details and detailed experiments we kindly refer the reader to the corresponding papers, which are listed at the beginning of each section. *Particularly, it should be mentioned, that the approaches are at least comparable or even superior to state-of-the-art results although this will not be illustrated in full detail.*

Chapter 6

Fully Supervised

In this chapter, we consider fully supervised learning, *i.e.*, we train a classifier in an on-line manner but all labels are provided by an oracle (*e.g.*, a human supervisor). Thus, we can assume to have zero (or as low as possible) label noise. We show this updated scheme by training an object detector in an interactive manner as depicted in Figure 6.1. During learning the classifier should continuously improve its performance, *i.e.*, reducing the false positive rate and simultaneously increase the detection rate.

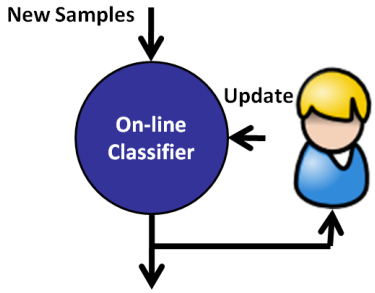


Figure 6.1: In a fully supervised learning system, the updates are assumed to be correct. *Active Learning* uses a feedback loop, *i.e.*, the classifier "asks" for informative examples, which are then labeled by an oracle.

6.1 Learning an Object Detector[1]

We have developed an efficient framework for automatic car detection from large aerial images. Cars appear as small objects, which vary in intensity and many details, which are not visible. Moreover, the urban scene contains a complicated background with variety of objects that look like cars such as windows, roofs of buildings, and corners of streets. All these properties make it difficult to characterize the features of a car and imposes challenges in recognizing cars from aerial images. The detection of cars is useful for many reasons. One particular application of, our group was further working on, is to improve ortho-photos [64]. The idea is to detect the cars, remove them by an in-painting approach and then to use this to build the ortho-photos. Even the 3D-reconstruction process can be improved.

We use *On-line Boosting for Feature Selection* together with an interactive training framework to efficiently train and improve a car detector. After training, detection is performed by applying the trained classifier exhaustively on the images. This process delivers many overlapping detections, which are the probabilities of the appearance of an object at a certain location. Therefore, a post processing stage is needed to refine and combine these outputs. A car is considered to be detected if the output confidence value of the classifier is above a threshold (*i.e.*, zero). On the one hand, the lower the threshold the more likely an object is detected as a car, on the other hand the more likely a false positive occurs. For a higher threshold the false positive rate decreases at the expense of the detections.

6.1.1 The Supervision

The training process is performed by iteratively labeling samples from the images and updating parameters for the model. The labeled samples can be positive or negative. In order to minimize the hand labeling effort we apply an active learning strategy. The key idea is that the user has to label only examples, which are not classified correctly by the current classifier. In fact, it has been shown in the active learning community [95], that it is more effective to sample at the current estimate of the decision boundary than the unknown true boundary. This is exactly achieved by our approach. We first evaluate the current classifier on an image. The human supervisor labels additionally "informative" samples, *e.g.*, mark the wrongly labeled examples (*i.e.*, either a false detection or a missing one) and performs an update of

[1] Adapted from T. Nguyen, H. Grabner, B. Gruber, and H. Bischof. On-line boosting for car detection from aerial images. *In Proceedings IEEE International Conference on Research, Innovation and Vision for the Future, 2007. [84]* and H. Grabner, T. Nguyen, B. Gruber, and H. Bischof. On-line boosting-based car detection from arial images. *ISPRS Journal of Photogrammetry & Remote Sensing, 63/3, 382-396, 2008. [34]*

the classifier. The new classifier is applied again on a new (or the same) image and the process continues.

6.1.2 Selected Experiments

We start with a random classifier. The classifier is improved on-line after labeling training samples by the user. During the training process we have labeled 1420 training samples. There are 410 positive samples, each sample contains a car, and 1010 negative samples, each contains diverse background image patches (for a few examples see Figure 6.2).

(a) positive (b) negative

Figure 6.2: Examples of positive (a) and negative (b) labeled training samples during the on-line training process.

This whole interactive training process takes approximately four hours[2]. The more informative the samples are the faster the system can learn. Compared to other object (car) detection systems, our system needs quite small number of image samples for training. The number of positive samples is much less than the number of negative samples we need to train the system, which comes from the fact that the variability of the background is much larger than the one of cars.

As we can see in Figure 6.3, after several iterations almost all cars, which have rather clear appearance and fit to the (angle of the) detector, are detected. For a quantitative evaluation, we use the common measurement for object detection problem named recall-precision curve (RPC) [2]. The precision rate shows how accurate we are at predicting the positive class. The recall rate shows how many of the total positive we are able to identify. The F-measure is the harmonic mean that can be considered as trade-off between recall and precision. Figure 6.3 (d) shows the continuous improvement of the training classifier over time (*i.e.*, number of labeled training samples).

[2]Since we are training on-line, a classifier is available at each time. Thus, with fewer training examples an acceptable result can be obtained after about 2 hours. The longer the training (*i.e.*, the more samples are labeled) the better the performance will be.

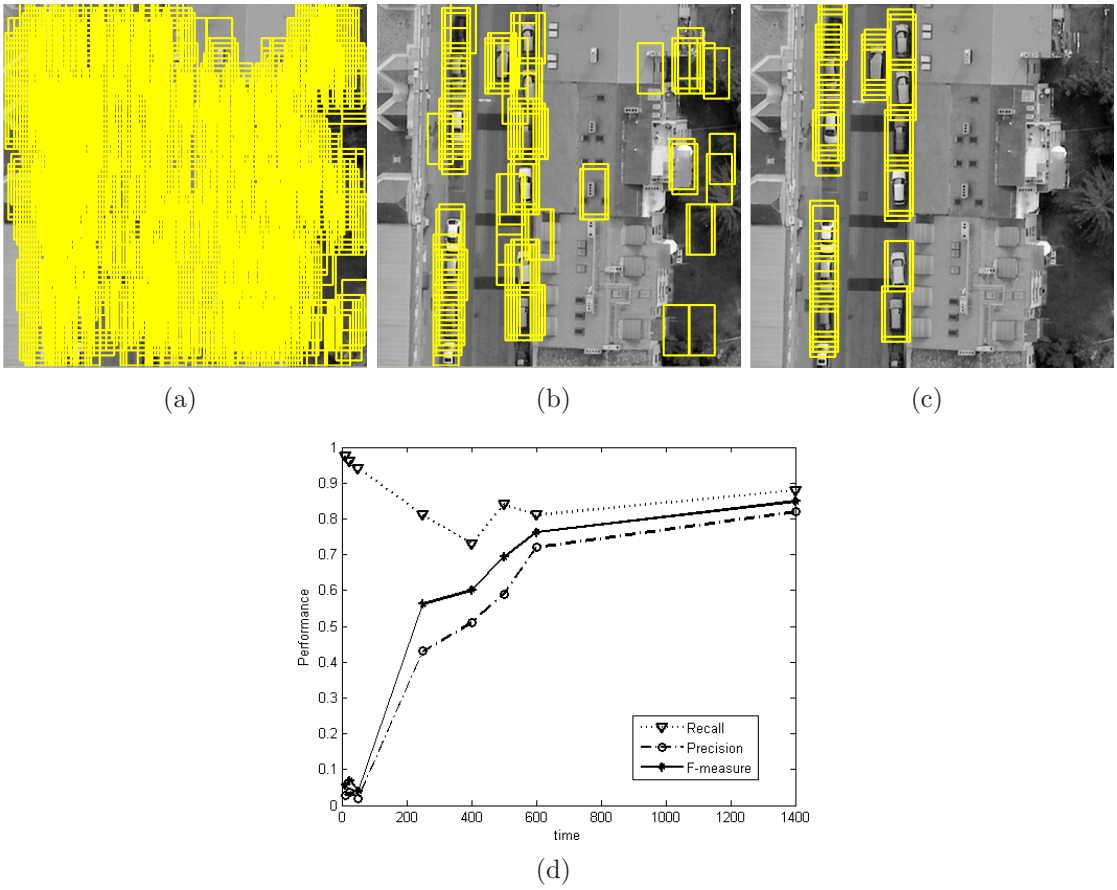


Figure 6.3: Learning process: Improvement of classifier performance - (a) original subimage, (b) result after training the classifier with only one positive sample, (c) after training with a few samples and (d) the final result without post-processing.

6.1.3 Discussion

Since labeling of samples in the training phase is an interactive process with visualization, we can intuitively choose (label) the most informative and discriminative sample at each update. This allows the parameters of the model to be updated in a greedy manner with respect to minimizing the detection error, meaning that the parameters of the model can be learned very fast. This process avoids labeling redundant samples that do not contribute to the current decision boundary. Thus, we employ a selective sampling technique, based on boosting, which dramatically reduces the amount of human labor required for this task. A similar method has been used by Abramson and Freund [1] motivated by the sentence: *One of the most labor intensive aspects of developing accurate visual object detectors using machine learning is to gather sufficient amount of labeled examples.*

6.2 Summary

Since we scope with a fully supervised learning problem with no (or almost no) label noise no drifting happens. Hence, on-line learning continuously increases the detection rate and reducing the false positive rate over time. However, the ordering of the examples have an significant effect of the performance and the learning time, as the theory suggests, *e.g.*, [95].

Chapter 7

Verification

In this chapter, we use a verifier to provide labels for updating the classifier. This is motivated twofolds. First, in order to minimize the hand labeling effort, *e.g.*, we want to limit the number of interactions of a human supervisor. Second, there exist applications, where no interaction is possible at all, but a verification can be done via other cues. The first three applications, shown in this chapter, are improving an object detector over time, the fourth application is visual object tracking.

7.1 Learning an Object Detector using 3D Information[1]

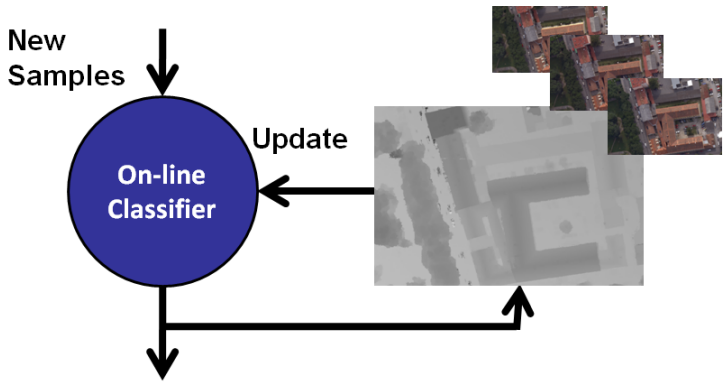


Figure 7.1: The labels for updating the on-line classifier are verified by a 3D height model. The robustness is achieved by using redundancy. In fact, overlapping images are used to build such range image and a conservative update strategy.

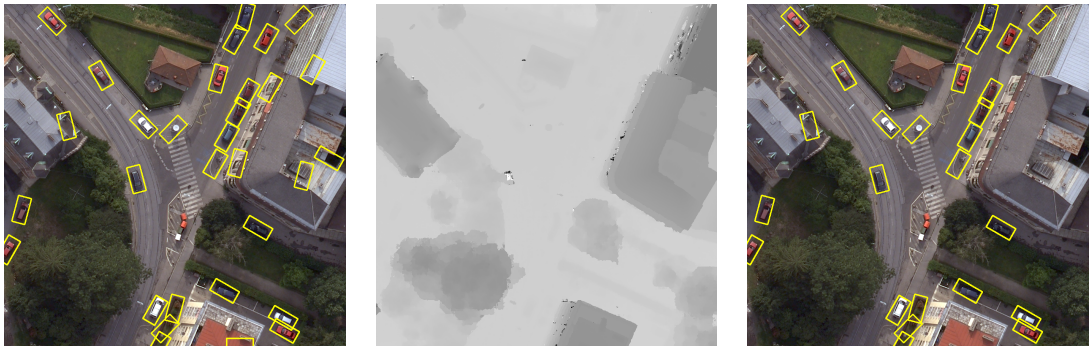
For the task of object detection, we demonstrate how to reduce the hand labeling effort considerably by 3D information. In particular, we show the training of an efficient car detector for aerial images. *On-line Boosting for Feature Selection* is used to incrementally improve the detection results. Initially, we train the classifier with a single positive (car) example, randomly drawn from a fixed number of given samples. When applying this detector to an image we obtain many false positive detections. We use information from a stereo matcher to detect some of these false positives (*e.g.*, detected cars on a facade) and feed back this information to the classifier as negative updates. This improves the detector considerably and thus reduces the number of false positives. Similar results to hand labeling by iteratively applying this strategy can be obtained.

7.1.1 The Supervision

The goal is to facilitate the problem of obtaining a large number of labeled samples by using 3D information as a teacher (*i.e.*, to provide labels for the on-line boosting algorithm), depicted in Figure 7.1.

After initialization with a single positive sample the iterative training process is started. The current classifier is evaluated on a randomly chosen training image.

[1]Adapted from S. Kluckner, G. Pacher, H. Grabner, H. Bischof, and J. Bauer. A 3D teacher for car detection in aerial images. *In Proceedings ICCV Workshop on 3D Representation for Recognition (3dRR-07), 2007.* [60]



(a) (b) (c)

Figure 7.2: A small part of an areal image with all detections superimposed (a). After verification using an range image from 3D reconstruction with a radiometric resolution of 16 bit (b), only the overlayed detections (c) are verified.

This classifier provides a set of locations, where it detects cars in the image. Only a subset of them are correct detections; the others are false positives. We aim to identify these false positives robustly and use them as negative updates for the classifier. The verification is performed on the according 3D range data of the aerial image. We use the simple but powerful and robust assumption that a car has to be located on similar height values in the currently visited detection window. Therefore, we analyze the height data in this enlarged patch window at the location of a detection and fit a plane to these values. From the robust estimate of the plane for the range image patch, we verify the detection on the basis of its slope. If the slope of this plane above a certain threshold, we consider this patch as negative update.

After that, the classifier is evaluated on the positive samples and updated with the worst performing one, if it is not detected as car. This autonomous process is repeated until a stopping criterion is fulfilled. In contrast to manual training we can train a classifier without any human interaction apart from the construction of the set of positive samples. Comparing our learning curve to the results given in [34, 84] (Section 6.1), we obtain a similar performance in a fractional amount of time.

7.1.2 Discussion

The approach is quite similar to just bootstrapping continuously from the range image negative samples. However, we benefiting from active learning and decrease training time. Note, if we consider a false positive as a correct detection, this does not perturb the on-line learning process. We only update false detections, where we are confident that they do not correspond to cars. In Figure 7.3 subsets of the

positive and negative samples that are used for updating are shown.

(a) positive (b) negative

Figure 7.3: Examples of positive (a) and negative (b) samples used for updating the classifier. The positive samples are hand labeled and given in advance, while the negative samples are generated autonomously.

As can be seen the negative ones correspond to those, which are on facades (*i.e.*, have a large different in the height values). Nevertheless, sometimes cars appear in the set of negative updates. Note, this label noise only occurs as false negatives since we use a fixed set of positive images. Unfortunately, the number (variance) of positive images is constant. At the moment we cannot robustly generate new positive updates from the range image. Thus, we evaluate the classifier on all hand labeled positive examples in each iteration of the learning process. To avoid drifting, we perform a positive update of the classifier, if a sample is not correctly classified. This ensures that we train a car-detector and allows to recover from wrong updates.

The approach can be further extended by including more cues, *e.g.*, Pacher *et al.* [93] make use of an extracted street layer in order to reduce the false positive rate.

7.2 Learning an Object Detector using a Multi-camera System[2]

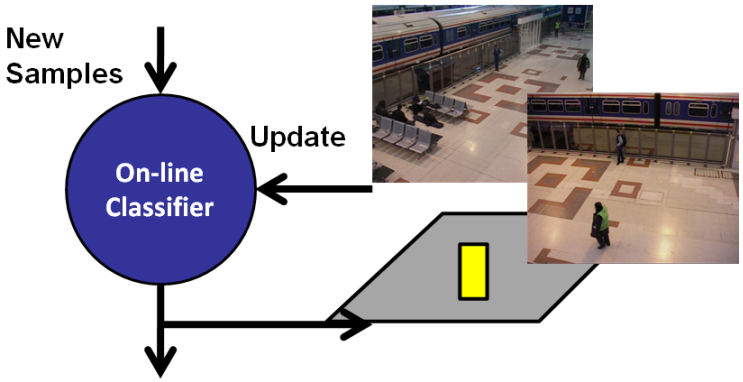


Figure 7.4: The detections of the on-line classifier are verified by all other classifiers, corresponding to cameras looking at the same scene. Since the system is running for a long period of time, only very confident decisions are used in order to update the model.

We make use of both, verification using redundancy and verification using geometry, which brings us to a multi camera system. The overall camera network is depicted in Figure 7.5 (a). We have a setup with n partly overlapping cameras, each of them observing the same 3D scene. In general, the objects-of-interest can move in the world coordinate system (x_w, y_w, z_w). We can assume that the objects-of-interest (*i.e.*, the persons) are moving in a common ground-plane. However, having overlapping camera views the local image coordinate system (x_i, y_i) can be mapped onto each other by using a homography based on an identified point in the ground-plane. In addition, for each camera an estimation of the ground-plane is required.

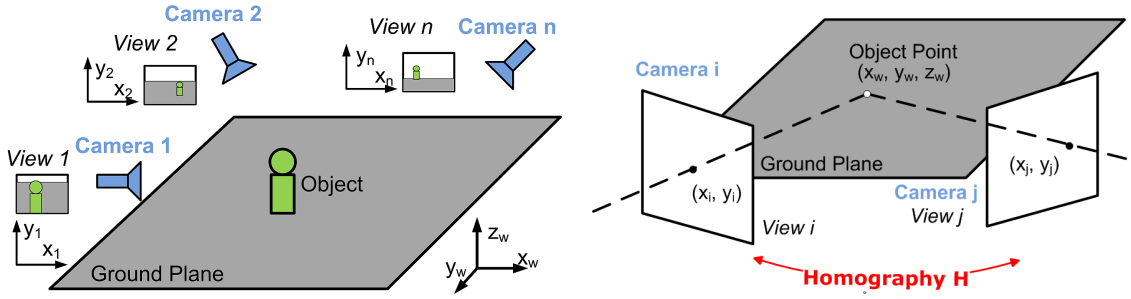
Similarly to Khan and Shah [58], we use homography information to map one view onto an other. It is well known (see, *e.g.*, [47]) that points on a plane from two different views are related by a planar homography. The principle is depicted in Figure 7.5 (b). Hence, the plane induces a homography $\mathbf{H}$ between the two views, where the homography $\mathbf{H}$ maps points $\mathbf{x}_i$ from the first view to points $\mathbf{x}_j$ in the second view:

$$\mathbf{x}_j = \mathbf{H}\mathbf{x}_i \ . \tag{7.1}$$

In particular, we estimate the homography by selecting corresponding points manually. In fact, for the current setups we estimated the ground-plane manually by selecting at least four corresponding points in each image, but to have an autonomous

[2]Adapted from C. Leistner, P. Roth, H. Grabner, H. Bischof, A. Starzacher, and B. Rinner. Visual On-line Learning in Distributed Camera Networks. *In Proceedings International Conference on Distributed Smart Cameras*, 2008. [66]

system an unsupervised autonomous approach (*e.g.*, [96]) might be applied as well.



(a) overview (b) homography induced by a plane

Figure 7.5: System overview of our proposed approach (a). Multiple cameras observe a partly overlapping scene and collaborate during update phase. The cameras can exchange information because the scene is calibrated which can be done calculating a homography between each cameras (b).

Once we have calibrated the scene we can start co-training [10, 69]. In fact, in our approach the different views on the data is realized by different camera views, *i.e.*, we can verify if a detection in one view was also reported in a different one.

7.2.1 The Supervision

In order to start the training process, we first train a general prior $H^P(\mathbf{x})$ by off-line boosting. Thus, a classifier is trained using a set of positive $\mathcal{X}^+$ and negative labeled samples $\mathcal{X}^-$ [130]. Such a classifier is trained emphasizing on a high recall rate rather than on a high precision and can therefore be applied on all different views. Then, the thus obtained classifier H^P is cloned and used as an initial classifier for all camera views. Please note, even exactly the same classifier is applied for that purpose due to the different camera positions we get the independent observations required for co-training. Since it has been shown [106] (see Section 4.4) that off-line classifiers can be easily re-trained using on-line methods (*i.e.*, all acquired statistics are interpreted as if they have been estimated on-line) these cloned classifiers can be re-trained on-line and adapted to a specific camera later on.

In particular, we propose the following re-training approach for updating n cameras, where we verify or falsify the obtained detections, in order to improve the corresponding classifiers:

Verification If all responses of the other (corresponding) classifiers $j = 1, \ldots, n$ are also positive, the example is verified and added to the pool of positive examples: $\mathcal{X}^+ \cup \mathbf{x}$.

Falsification If all responses of the other classifiers are negative, the example is classified as a false positive. Thus, the classifier H_i is updated immediately with $\mathbf{x}$ as a negative example. After each negative update the pool of positive samples $\mathcal{X}^+$ is checked if it is still consistent; otherwise a positive update with the corresponding sample is performed.

In all other cases we do not perform an update. With this very conservative and simple update strategy the arising label noise can be minimized and thus the detections keep stable over time. Since we are continuously learning over time these few updates are sufficient to adapt to the specific scene. Note, if $H_i(\mathbf{x})$ has a negative response nothing happens. However, the positive samples are collected during the verification step and the local pool of scene-specific samples $\mathcal{X}$ increases.

7.2.2 Selected Experiments

In the experimental setup we assume static cameras with partly overlapping views. For all cameras sharing a view-point area, we estimate the ground-plane homography. We evaluated our approach on the *PETS 2006*[3] data set. It can be seen, that although only a small number of frames were processed the precision was significantly improved while the recall rate stays at the same level. The improving classifier performance over time is illustrated in Figure 7.6.

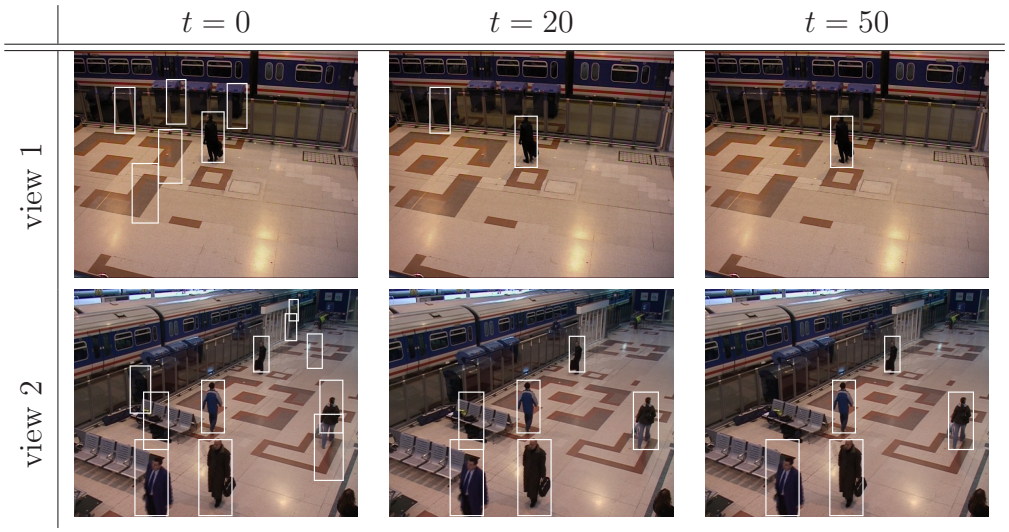


Figure 7.6: Improvement over the co-training iterations on the PETS 2006 indoor scene. After $t = 50$ iterations, the results stayed at a constant performance level.

[3] `http://www.pets2006.net`, (February 13, 2008)

7.2.3 Discussion

Due to homography projection and label jitter, alignment errors are introduced even when the ground plane is perfectly calibrated. Hence, misaligned samples may be used as updates, which may ends in corrupted classifiers. This shows, the importance of well aligned samples.

An further problem occurs, if he object is partially occluded by other objects or persons. Hence, the other camera(s), which are used to verify/falsify the detection may correctly detects the object this might yield to wrong updates. More sophisticated methods, like a global occurrence-map [23] would limit these problems. Also other cues, like motion or shape information can be easily included in order to achieve more robust updates, *i.e.*, decrease label noise and label jitter by doing less (only very secure and well aligned) updates. Some of this approaches will be mentioned in the in the next section, within the *Conservative Learning* framework.

7.3 Learning an Object Detector using a Reconstructive Model[4]

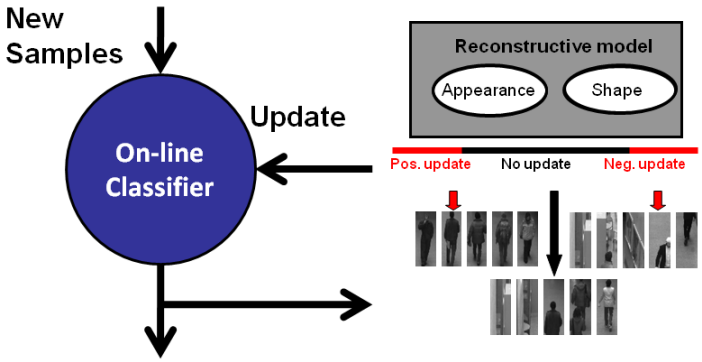


Figure 7.7: The detections of the discriminative on-line classifier are verified by another classifier, which, in fact corresponds to a reconstructive model. Since the system is running over a long period of time, on-line very confident decisions are used in order to update the model.

On-line learning in order to improve detection results has been investigated in the past, *e.g.*, [69, 83]. The basic idea is to start with a very simple object detection system and to exploit a huge amount of unlabeled video data by being very conservative in selecting the training examples. In fact, we start with a simple moving object classifier and proceed with incremental PCA (on shape and appearance) as a reconstructive classifier. The key idea is to use reconstructive and discriminative classifiers in an iterative co-training fashion to arrive at increasingly better object detectors. We demonstrate the framework on a surveillance task, where we learn person detectors with minimal or even without hand labeling.

Discriminative representations (as it is achieved by the on-line boosting) are compact, task dependent, efficient, and effective, but usually not very robust. On the other hand, reconstructive representations are usually less efficient and less effective, but more general and robust. The ultimate goal is to combine these two representations to achieve the best of both worlds, which would lead to efficient and effective, while still general and robust continuous learning techniques.

[4]Adapted from P. Roth, H. Grabner, D. Skocaj, H. Bischof, and A. Leonardis. Conservative visual learning for object detection with minimal hand labeling effort. *In Proceedings German Association for Pattern Recognition (DAGM), pages 293300, 2005. [107]* and P. Roth, H. Grabner, D. Skocaj, H. Bischof, and A. Leonardis. On-line conservative learning for person detection. *In Proceeding IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. [108].* Additionally, the interested reader is pointed to the PhD-Thesis [105] from P. Roth about more details.

7.3.1 The Supervision

We use a reconstructive model, which assures robustness and serves for verification. In fact, we use a PCA-based subspace representation as a reconstructive model. This low-dimensional representation captures the essential reconstructive characteristics by exploiting the redundancy in the visual data. As such, it enables "hallucinations" and comparison of the visual input with the stored model [67]. Having this model, each image can be checked whether it is consistent with the current model or not. When a false detection occurs, the reconstruction error is significantly larger (*i.e.*, the original image and its reconstruction differ significantly), thus the image gets discarded. We thus assure that the discriminative learner gets most of the time the clean data. If the reconstruction error for both, appearance and shape, is very low there is a positive update of the classifier; if the reconstruction error is big and some motion restrictions are fulfilled there is a negative update. Both, positive and negative updates are required.

In this way the inconsistent data can be rejected and the discriminative model can be trained from "clear" data only. The whole process run in an incremental manner, feeding the learner continuously, as new data arrives.

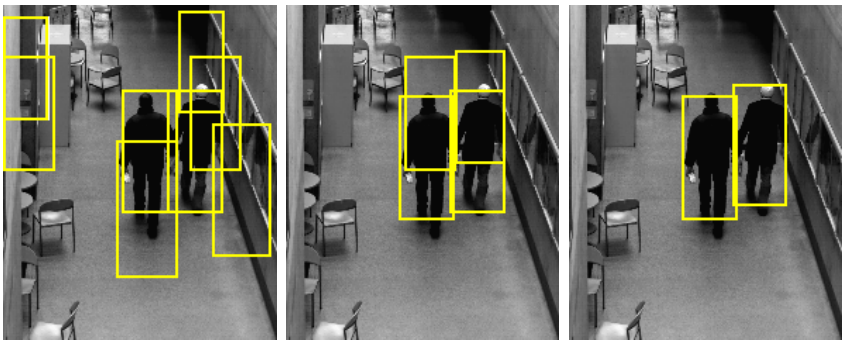
7.3.2 Selected Experiments

For evaluation purposes we have generated a challenging test set of 300 frames (containing groups of persons, persons partially occluding each other and persons walking in different directions) and a corresponding ground truth. Figure 7.8 shows the detections by three different on-line classifiers (initial, after 300 and after 1200 training frames) on the test sequence. Since this initial classifier is worse there is a greater number of false positives in the beginning (a) that can be completely removed by on-line training, as can be seen in (b) and (c). Detailed performance curves are depicted in subfigure (d). Some more examples of persons correctly detected by the final classifier are depicted in Figure 7.9 (first row).

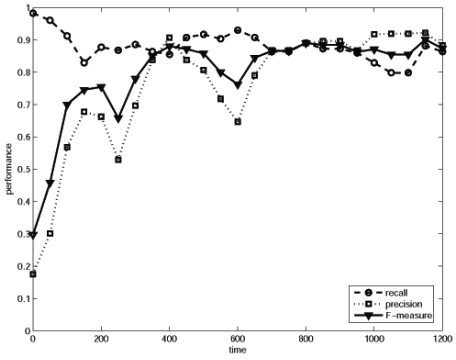
The proposed framework is quite general (*i.e.*, it can be used to learn completely different objects and can be extended in several ways, *e.g.*, Figure 7.9 (second row) demonstrates the same algorithm applied to cars.

7.3.3 Discussion

The classifier is only updated if we are very confident, in particular all patches are verified by motion and PCA (appearance and shape). Furthermore, the updates are well aligned (no label jitter) mainly due to the reconstructive shape model. For examples of positive and negative updates see Figure 7.10.



(a) 0 (b) 300 (c) 1200



(d)

Figure 7.8: Improving detection performance over time (updates).

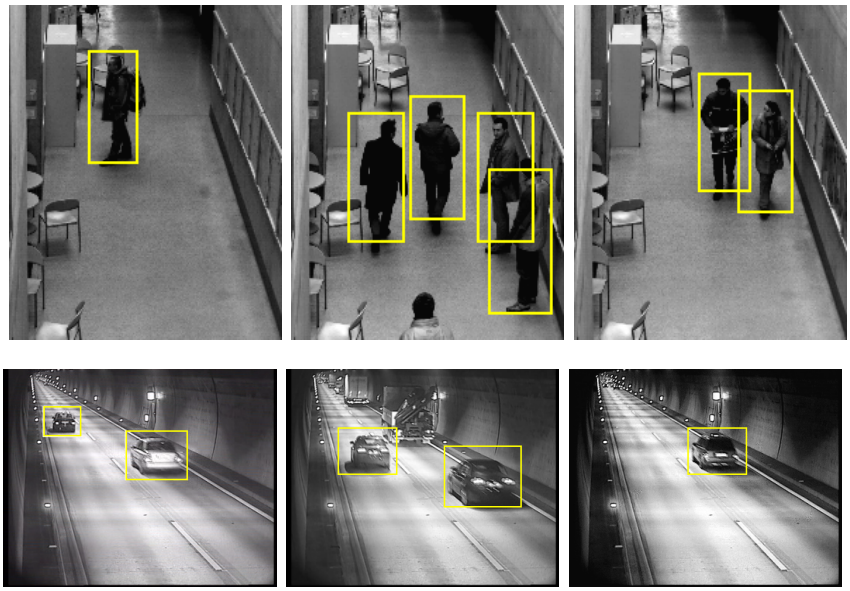


Figure 7.9: Detections by the final classifiers.

(a) positive (b) negative (c) uncertain

Figure 7.10: Examples of positive (a) and negative (b) samples used for updating the classifier. Due to the conservative approach no updates are done with uncertain samples (c).

The conservative approach assures, that non-relevant (corrupted or inexact) data is included into the model. However, if once a wrong update is done the approach drifts. Most of the time, this error is immediately corrected by the next updates. Thus, the classifier has the ability to recover from wrong updates, as can be seen in Figure 7.11. More modules, operating on different modalities (*e.g.*, tracking, color, *etc.*) will further increase the robustness and generality of the system.

(a) $t-2$ (b) $t-1$ (c) t

Figure 7.11: After achieving stable results marked by the bounding boxes (a) a wrong update causes failures, *i.e.*, many false positives (b). However, the model has the ability to recover (c) by the next few updates (red: negative updates; green: positive updates; white: uncertain detection).

7.4 Tracking by Detection[5]

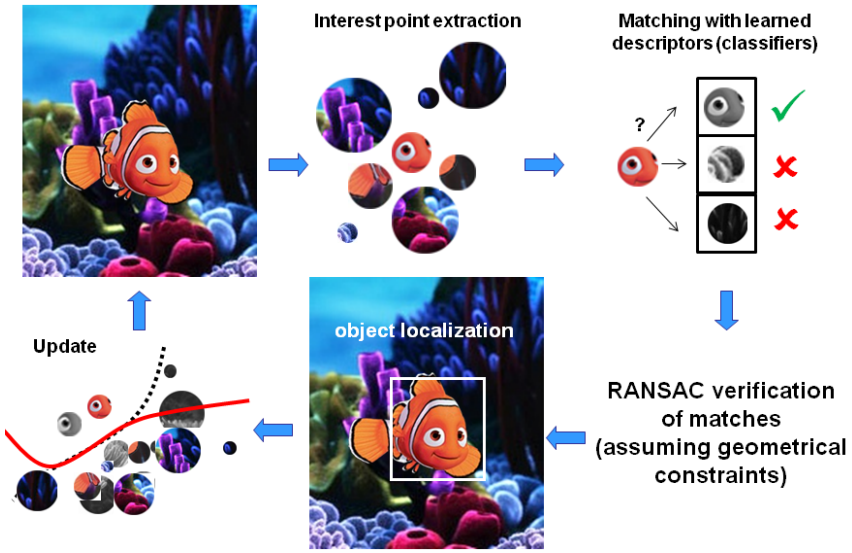


Figure 7.12: Tacking loop: Learning a classifier for feature representation allows to simplify the problem of matching keypoints since the corresponding point has just to be distinguished from to currently detected ones. The usage of an on-line classifier allows to build this representation during tracking by collecting samples over time.

We treat tracking as a matching problem of detected keypoints between successive frames. The description of the keypoints are learned using classifiers. Contrary to existing approaches, we are able to start tracking of the object from scratch requiring no off-line training phase before tracking. The tracker is initialized by a region of interest in the first frame. Keypoints lying with this region are considered as object keypoints, all others as background keypoints. The task is to robustly re-detect the object by its object keypoints. Therefore, the descriptions of the object keypoints are learned. In fact, we propose to learn distance functions in the space of keypoints $d : K \times K \rightarrow [0, 1]$. Similar to [49], boosting can be used to learn a classifier which is then equivalent to learning a distance function. In other words we implicitly define a multiclass classification problem in the original vector space K by generating a binary partition of the data in the product space $K \times K$. We choose the simple one vs. all partition, *i.e.*, each keypoint can be distinguished from all the other points. In contrast to methods using a fixed metric for keypoint description (*e.g.*, SIFT [78]), discriminative learning of keypoint descriptions allows to incorporate scene information and therefore simplifies the tracking problem to a

[5]Adapted from `M. Grabner, H. Grabner, and H. Bischof. Learning features for tracking.` *`In Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR),2007 [41].`* Additionally, the interested reader is pointed to the PhD-Thesis [39] from M. Grabner about more details.

classification problem among the currently detected scene keypoints, *i.e.*, we only have to discriminate the current object keypoints from the ones detected in the background.

During tracking we want to use the most reliable features of the object. Therefore, we introduce a mechanism which on-line determines a ranking of the currently used features by estimating for each classifier the probability for a match in the next frame. Since we have independent classifiers for local features, this measure can be used for discarding bad features and replacing them with new ones which probably might be better suited for matching.

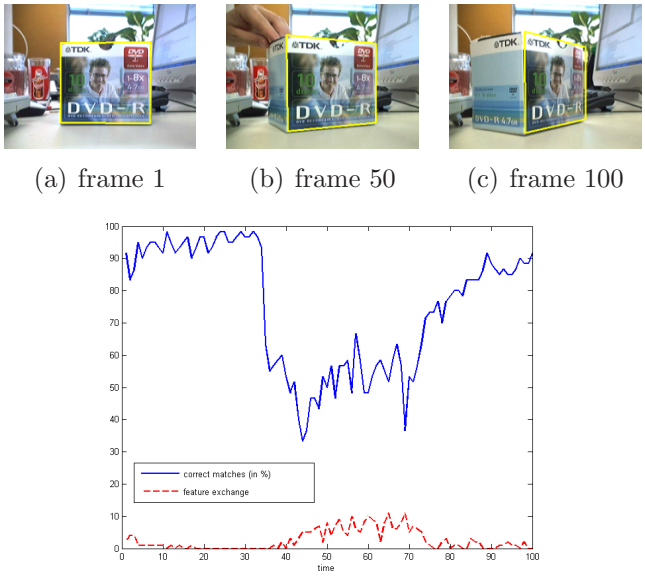
7.4.1 The Supervision

The tracking loop is depicted in Figure 7.12. Matches are obtained by applying the classifiers on all detected keypoints. For verification of them, which is important for obtaining correct labels for updating, the homography between successive frames is estimated using RANSAC [47]. Thus, in fact, the geometry serves as verifier. Establishing a homography assumes that the tracked object is planar or the depth discontinuities are small compared to the distance to the camera. Thus, we achieve a subset of correct matches $M^c \subseteq M$ verified by the homography. In case that the number of inliers exceeds a threshold, we assume to have correctly determined the homography and successfully tracked the object. Therefore, for each classifier we can compute its corresponding patch in the actual frame, which is then used for making a positive update of the classifier H_i. For negative updates we again choose patches extracted from any other keypoint $k \in K_t$. If we cannot establish the homography of the object between two consecutive frames we perform no updates to the classifiers and no detection is achieved.

7.4.2 Selected Experiments

On-line learning of local features allows us to simplify the classification problem in an elegant way. On the one hand an on-line classifier allows us to exploit information obtained over time by applying correct matches as positive updates to the corresponding classifier. In addition, discriminative classifiers allow to incorporate scene information by considering background keypoints as negative samples. An example is depicted in Figure 7.13. After changing the pose of the object, the number of matches decreases. Classifiers are exchanged as well as updated and the tracker stabilizes again.

Different objects have been chosen in order to illustrate the tracking performance, shown in Figure 7.14. The first row shows the property of local approaches of being invariant to occlusions. In addition, illumination changes and affine transformations



(a) frame 1 (b) frame 50 (c) frame 100

Figure 7.13: A pose change of the target object causes a decrease of the number of matches. If the appearance change of the object is not too large such that the object does not get lost, the mechanism for exchanging local features allows to find reliable features for matching again.

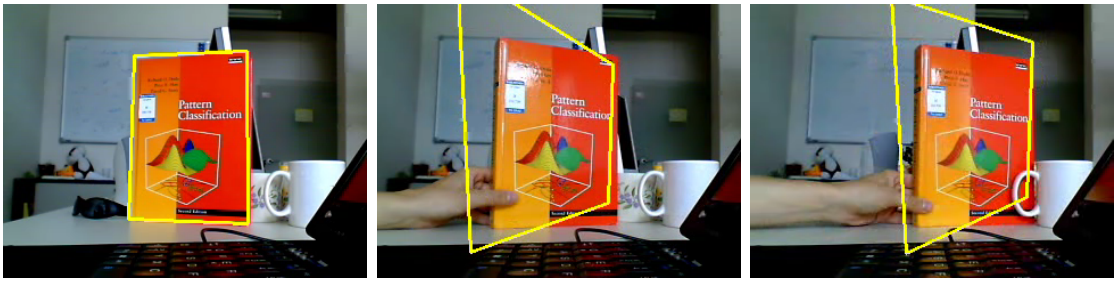
of the target object do not confuse the tracker. It can even handle large scale and pose changes of the target object as illustrated in row 2 due to the feature exchange property. Fast movement of the target object can cause a loss of the target object, as shown in the next sequences. However, once the appearance is similar to the one before it has been lost, the tracker can re-detect the object and continue tracking. Row 5 illustrates that even targets with a certain amount of changing content can be successfully tracked. In this case we have simulated a changing texture by tracking a monitor that is playing a video. The last sequence shows the benefit of discriminative feature learning. A textured target is tracked even though the same texture is present in the background.

7.4.3 Discussion

The supervision is provided through a robust geometrical verifier, *i.e.*, the keypoints are updated after the robust RANSAC verification. However, if the object is occluded for a certain time (and due to the feature exchange), keypoints found on the occluding object can start dominating the tracking. Hence, the tracker focuses on the wrong object (similar as it is shown in Section 8.1.4). In the non occluding case, drifting mainly occurs if the homography can not estimated properly. This is the case if too few keypoints are found on the object or if they are not well dis-

Figure 7.14: Sample sequences.

tributed among the object. Even a slightly incorrect estimated homography can yield to confusions, *i.e.*, background keypoints become object keypoints or vice a verse. Furthermore, new keypoints at the boundary can be wrongly assigned to either the set of object or background keypoints. This is shown in Figure 7.15. By using some simple heuristics the effect can be limited. However, fast appearance changes can only be handled if the number of inliers is allowed to be quite small (*i.e.*, the parameters of RANSAC are set soft). In general, there is a trade-off of losing the object or continuously tracking it with the possibility to drift. Since in



(a) (b) (c)

Figure 7.15: Drifting of the tracker since object keypoints and background keypoints where swapped during the RANSAC verification step (b). Once the homography is wrongly estimated the updates and thus the tracking results stay wrong (c).

the first case no updates are done, tracking may continue if the object has returned to its former appearance.

7.5 Summary

In this chapter, we present a method for dealing with unlabeled data using a supervised learning algorithm. In fact, this is done with the help of other accessible information. This informations is used to verify or falsify the decision of the classifier *before* the sample may is used for an update. A couple of possible verifiers are shown on different computer vision applications. In fact, we used verifications via redundancy, verifications via other learning algorithms, and verification via geometrical constrains. Of course, one can think of many more possibilities, *e.g.*, motion, symmetry, *etc.*, or combinations of them. Further, even non visual verification may be used, *e.g.*, via a touch-sensor on a robot. The underlying principle is to reduce the label noise by independent information. Since, in many applications a big (unlabeled) dataset (*e.g.*, a video stream) is available, one can be very conservative, just allowing very "secure" updates. As a side remark, geometric constrains are used since a long time in computer vision, *e.g.*, in order to establish correspondences in images by RANSAC [47].

Chapter 8

Self-learning

In this chapter, we consider self-learning, *i.e.*, the classifier predicts the labels and directly performs an update with its own predictions. Due to this direct feedback the classifier is highly susceptible to drift, *e.g.*, achieve self-fulfilling prophecies. However, on controlled, well defined tasks, or for a short period of time, impressive results can be achieved. We show this approach on two examples, the first on object tracking and the second on background modeling.

8.1 Classifier-based Template Tracking[1]

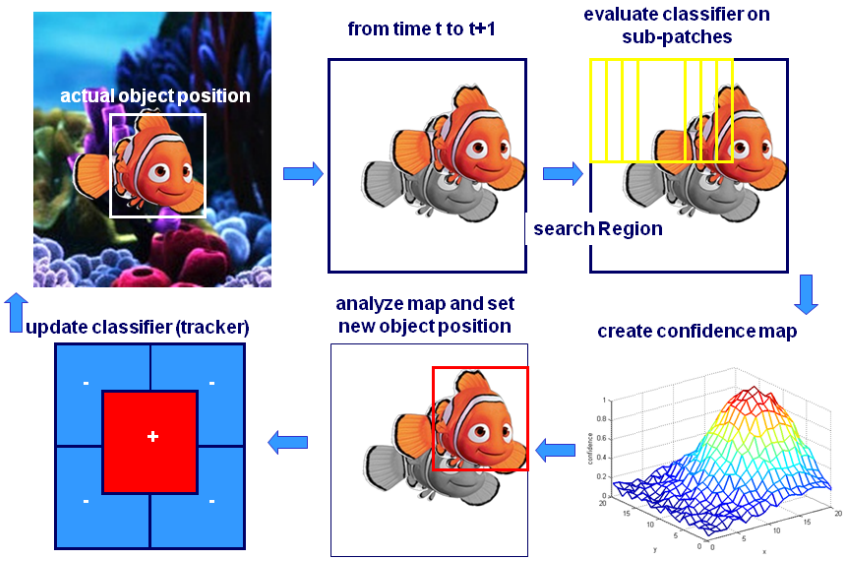


Figure 8.1: Tacking loop: Given an initial position of the object in time t, the classifier is evaluated at many possible positions in a surrounding search region in frame $t+1$. The achieved confidence map is analyzed in order to estimate the most probable position and finally the tracker (classifier) is updated. The usage of an on-line classifier allows to build this representation during tracking by collecting samples over time and so the classifier focus on the local problem of distinguish the object from its surrounding background.

Very recently tracking was approached using classification techniques such as support vector machines [4, 5, 136]. Similarly, we consider the tracking problem as a binary classification problem between object and background. Most existing approaches construct a representation of the target object before the tracking task starts and therefore utilize a fixed representation to handle appearance changes during tracking. In addition, to the on-line adaption problem, recently many techniques have addressed the idea of using information about the background in order to increase the robustness of tracking. We are dealing with a simpler problem. We want to train a classifier, which discriminates the current appearance of the object from its surrounding background. Training (updating) the classifier is done in an on-line manner. This allows to adapt the classifier while tracking the object. Therefore, appearance changes of the object (*e.g.*, out of plane rotations, illumination

[1] Adapted from `H. Grabner, M. Grabner, and H. Bischof. Real-Time Tracking via On-line Boosting.` *In Proceedings British Maschine Vision Conference, 2007* [31]. Additionally, the interested reader is pointed to the PhD-Thesis [39] from M. Grabner about more details.

changes) are handled quite naturally. Moreover, depending on the background the algorithm selects the most discriminative features for tracking resulting in stable tracking results. By using fast computable features the algorithm runs in real-time.

8.1.1 The Supervision

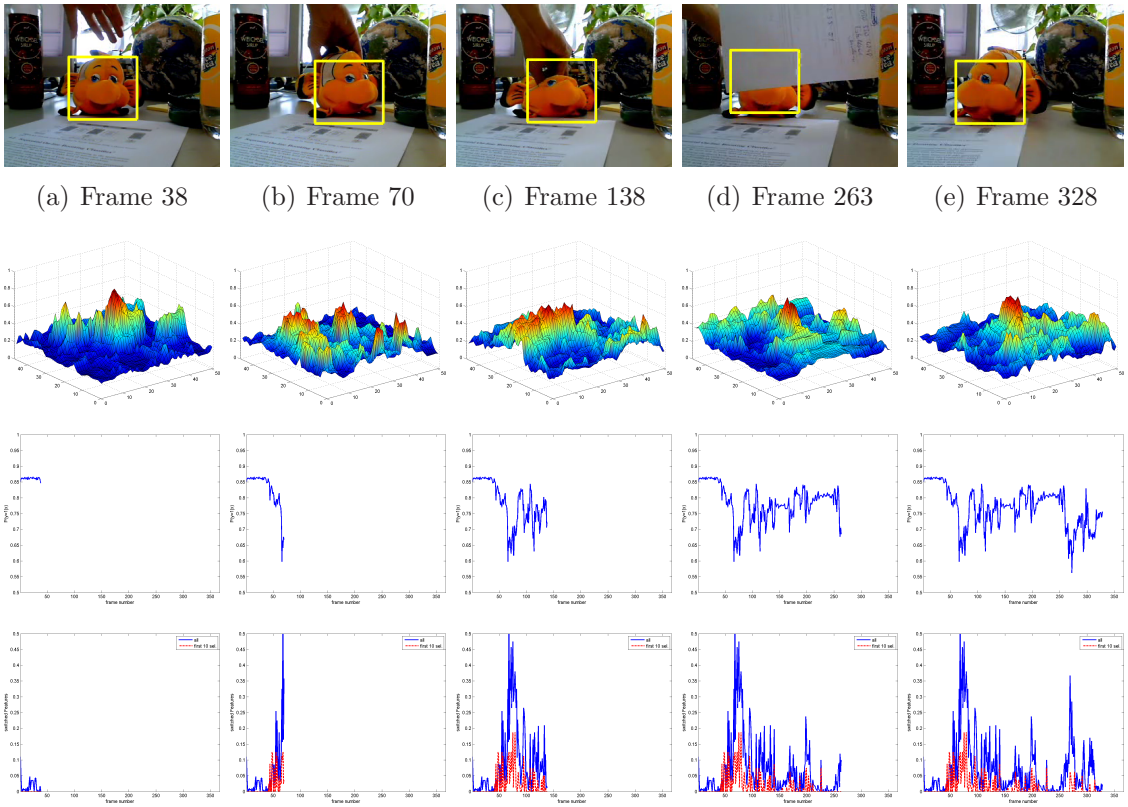
The principle of the tracking approach is depicted in Figure 8.1. Since we are interested in tracking, we assume that the target object has already been detected. This image region is assumed to be a positive image sample for the tracker. At the same time negative examples are extracted by taking regions of the same size as the target window from the surrounding background. These samples are used to make several iterations of the on-line boosting algorithm in order to obtain a first model, which is already stable. Note that these iterations are only necessary for initialization of the tracker. The tracking step is based on the classical approach of template tracking [46]. We evaluate the current classifier at a region of interest and obtain for each position a confidence value. Note, as shown in the theoretical section, boosting estimates the log-likelihood (Equation 2.3). We analyze this confidence map and shift the target window to the new location of the maximum.

Once the object has been detected the classifier has to be updated in order to adjust to possible changes in appearance of the target object and to become discriminative to a different background. The current target region is used as a positive update of the classifier while again the surrounding regions represent the negative samples. As new frames arrive, the whole procedure is repeated.

8.1.2 Selected Experiments

Figure 8.2 illustrates the behavior of our proposed method. If the target object changes its appearance or the surrounding background of the target becomes different, the tracker needs to update his features, which is reflected in oscillations of the confidence maximum (see row 3) and a flattened confidence map (see row 2). Movement of the tracking target is represented by a shifted peak in the confidence map. In the last row of the figure the percentage of switches within the selectors are plotted. The blue line corresponds to all selectors and the red line takes only the first 10 selectors into account. As the object changes its appearance the representation has to adapt. Therefor, the features switch. As expected, the first selectors are more stable and converge faster.

Results for different tracking object are shown in Figure 8.3. The approach can cope very well with occlusions and background clutter. Moreover, even large pose variations of the head (row 2) do not confuse the tracker showing that the tracker adapts to novel appearances of the target object. Row 4 illustrates that only little



(a) Frame 38 (b) Frame 70 (c) Frame 138 (d) Frame 263 (e) Frame 328

Figure 8.2: Tracking results on a sequence (row 1) containing a combination of appearance changes (*i.e.*, illumination, occlusion, movement, out of plane rotation). The behavior of the proposed tracker is analyzed considering the confidence map (row 2), the maximum confidence value over time (row 3) and the percentage of exchanged features (row 4).

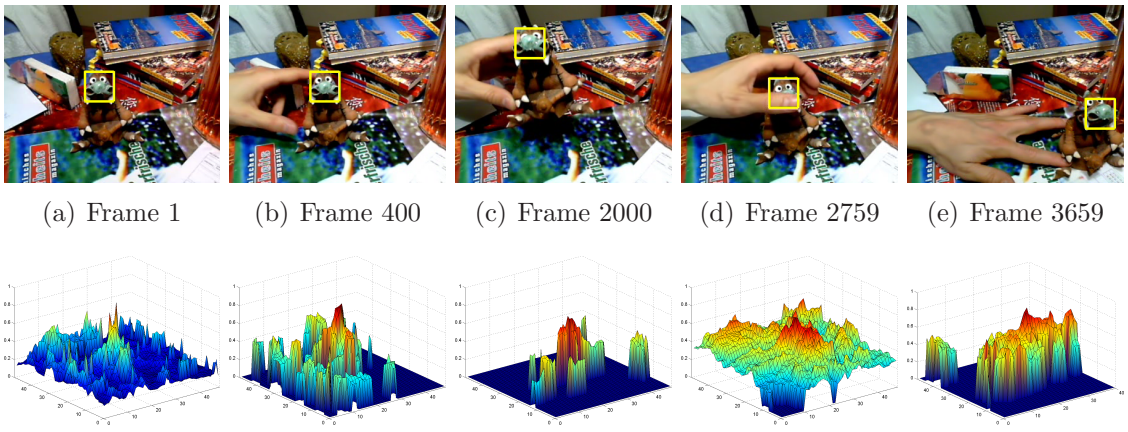
texture of the object is sufficient for tracking. A glass, having almost no texture, is tracked and again shows the reliability and adaptivity of the proposed tracker. Row 5 demonstrates the behavior in case of multiple very similar target objects. As can be seen, even though the objects significantly overlap the trackers get not confused demonstrating that the classifiers have really learned to distinguish the specific object from its background. The same argumentation can be used for the last row, where a piece of texture is tracked first in front of a homogeneous background. Afterwards, if we change the background to the same texture, features are chosen, which still allows for successful tracking of the object. To summarize, the tracker is able to handle all kinds of appearance variations of the target object and always aims at finding the best discriminative features for discriminating the target object and the background from the surrounding background.

Figure 8.3: To illustrate the generality of the proposed method, sequences of four different objects have been captured. The tracking results show that even objects with almost no texture (see row 4) can be successfully tracked. Moreover the tracking algorithm can cope with multiple initialized objects even if they have similar appearance (see row 5). Due to the on-line selection of discriminative features, the tracker can track the object (texture patch) within nearly the same background (see row 6).

8.1.3 Speedup using On-line *WaldBoost*[2]

In order to speed up the tracking process we are using the on-line *WaldBoost* algorithm (see Section 4.2) where the parameters were set to $\alpha = 0.02$ and $\beta = 0$.

Figure 8.4 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 (*e.g.*, as used by [141]), a confidence map is computed by evaluating the classifier at all positions within a local search region. The position of the object 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 speedup shown in Figure 8.5. The speedup is calculated as $N/\tilde{N}$, where $\tilde{N}$ is the number of weak classifiers used before the decision is reached and averaged 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 speedup is continuously increasing, since background patches can be discarded early. On average, we achieved a speedup 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 and thus results are exactly the same as in [31]. In general, the achieved speedup depends on dynamically changing problem difficulty and how often the Wald statistics have to be reset (*e.g.*, at frame 2706). Further, higher values of α lead to more speedup but having the risk of losing the object if it changes its appearance too fast. The achieved speedup (≥ 1) can be



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

Figure 8.4: Tracking of an object (1st row) and the classifier response map within the search window (2nd row). Values equal to zero mean early rejection, *i.e.*, saving of the computation time.

[2]Adapted from `H. Grabner, J. Sochman, H. Bischof, and J. Matas. Training Sequential On-line Boosting Classifier for Visual Tracking.` *`In Proceedings International Conference on Pattern Recognition (ICPR)`*, `2008. [38]`

used for instance for extending the search region to handle faster movements or to include other degrees of freedom like scale.

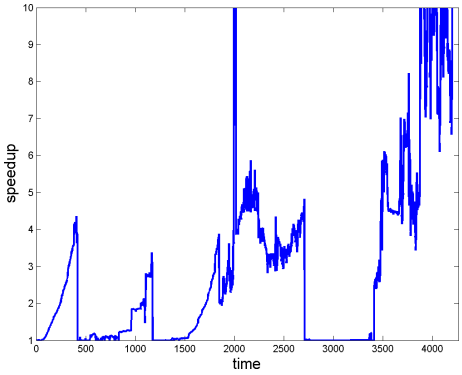
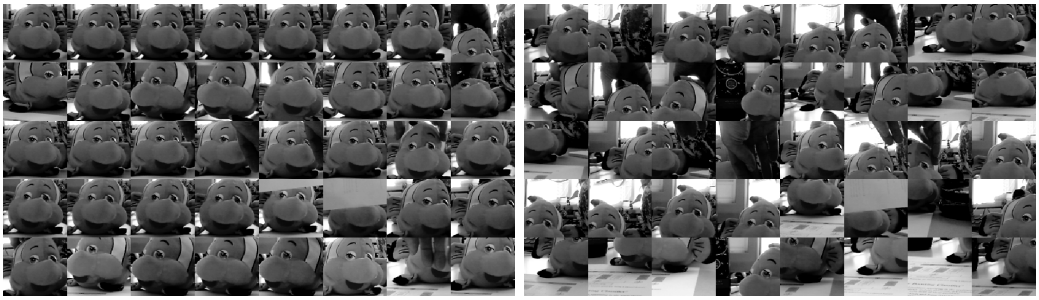


Figure 8.5: Speed-up compared to the non-sequential on-line boosting approach [31].

8.1.4 Discussion

Due to the self-learning policy occlusions cannot be distinguish from appearance changes. If occlusions are just short the model is not disturbed. The updates for the illustrative tracking sequence from Figure 8.2 are depicted in Figure 8.6. Which allows successfully tracking of the object under appearance changes.

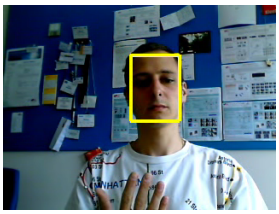


(a) positive (b) negative

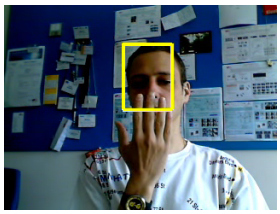
Figure 8.6: Examples of positive (a) and negative (b) samples used for updating the classifier.

Of course, an overlap for a long duration can cause adaption to the foreground object and finally leading to failures, as can be seen in Figure 8.7.

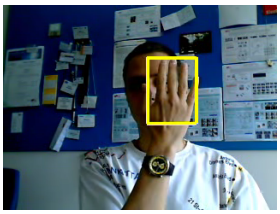
In general, drifting occurs in long term (noise) as well as by short term (fast movements) tracking sequences. For further discussion the reader is pointed to Section 10.2, where we briefly present an approach to limit these issues.



(a) Frame 37



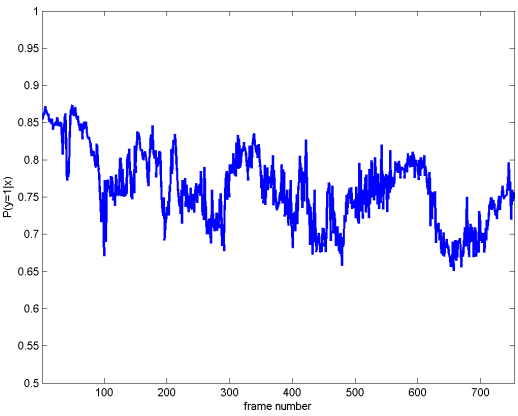
(b) Frame 118



(c) Frame 433

(d) Frame 642

(e) positive updates



(f) confidences

Figure 8.7: Drifting of the tracker, since it can not distinguish between (allowed) appearance changes and occlusions. Due to the wrong updates, the face tracker (a) gets a hand tracker (b)-(d) and (successfully) tracks the hand. This behavior is reflected by the performed positive updates (e). Unfortunately, the tracker does not notice that, as shown by the confidences (f), which does not significantly decrease over the whole sequence.

8.2 Classifier-based Background Model[3]

A basic task in surveillance applications is background subtraction. One needs a robust and flexible background model. Based on the idea of a block based background model [48] (which contains also a good overview of related work), we propose a *classifier based* background model. The basic idea is to partition the image in small (overlapping) blocks, each block contains a classifier, which classifies the region as foreground or background. The overall principle is depicted in Figure 8.8.

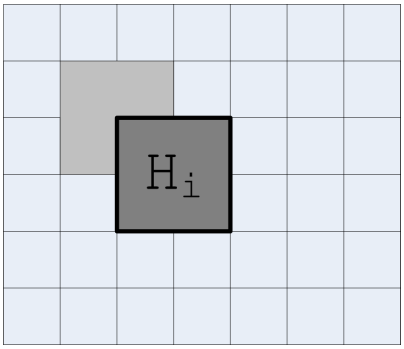


Figure 8.8: The background model is formed by a grid of regular aligned classifiers with an overlap in each direction.

We define as background everything, that is statistical predictable in the image. Therefore everything, that cannot be predicted is foreground. This definition allows us to describe dynamic (multi modal) background (*e.g.*, flashing light, moving leaves in the wind, flag waves, etc.). A robust background model has to adapt to dynamic backgrounds and must be sensitive to corresponding foreground objects. Therefore on-line algorithms are required. In the other case, when the patch can be modeled an update policy is used to update the model in order to take care of *natural* (allowed) changes in the background (e.g. lightning changes).

The main idea is to learn if the underlying image patch is predictable using the classifier. If so, this is considered as "allowed" background and otherwise it is considered as unknown (therefore foreground), *i.e.*, a region is labeled as foreground if it cannot be modeled by the classifier. Thus, the obtained confidence of the classifier is below a certain threshold $f(\mathbf{x}) < \theta^{eval}$, where $f(\mathbf{x})$ is the real valued confidence response of the strong classifier $H(\mathbf{x})$.

Since, boosting is used to train the classifier, which is a discriminative learning method, normally both positive and negative labeled samples are required in order

[3]Adapted from H. Grabner, P. Roth, M. Grabner, and H. Bischof. Autonomous Learning a Robust Background Model for Change Detection. *In Proceedings IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, 2006. [37]*

to learn a decision boundary. Contrary, the task of background modeling is a *one class* classification problem, meaning only positive samples (the observed images) are given. Therefore, for each feature the negative distribution is estimated directly without learning (*e.g.*, assuming each pixel is a random variable which is normal distributed or by using statistics of natural images [53]). In fact, we model the gray value of each pixel as uniformly distributed with mean 128 and variance $\frac{256^2}{12}$ (for an 8 bit image). Applying common statistics, the parameters of the negative distribution μ^- and σ^- of HAAR-like features can be easily computed.

8.2.1 The Supervision

For updating the classifiers we adopt the following very simple policy. We update a classifier if its confidence response is within a certain interval:

$$\theta_{lower}^{update} < f(\mathbf{x}) \leq \theta_{upper}^{update}. \tag{8.1}$$

Usually $\theta_{lower}^{update} = \theta^{eval}$ and the upper threshold is set to avoid over-fitting. In addition, in the post-processing steps several regions (known objects) are excluded from updating for a certain time.

First, in the initial period a separate classifier is built for each patch by considering all frames as positive examples. Later on, the algorithm analyzes the image and does positive updates according to a given policy. Depending on the design of the policy a wide range of behaviors is possible (*i.e.*, should an object be feed into the background model and if, how fast). Also the confidence values around the actual patch can influence the policy to build a stable spatial-temporal model.

8.2.2 Selected Experiments

An experimental example is shown in Figure 8.9 for a cluttered desk scenario. The first row shows the input sequence overlapped with blue rectangles indicating classifiers, which are currently updated. Those classifiers that give a negative response, and by definition these are foreground objects, are depicted in the second row. Added, removed or shifted objects in the scene are detected very well. Note, that during learning of this sequence the screen saver was active. Therefore, this dynamic background has been correctly modeled as background by the classifiers. But, when we wake up the computer the change is detected.

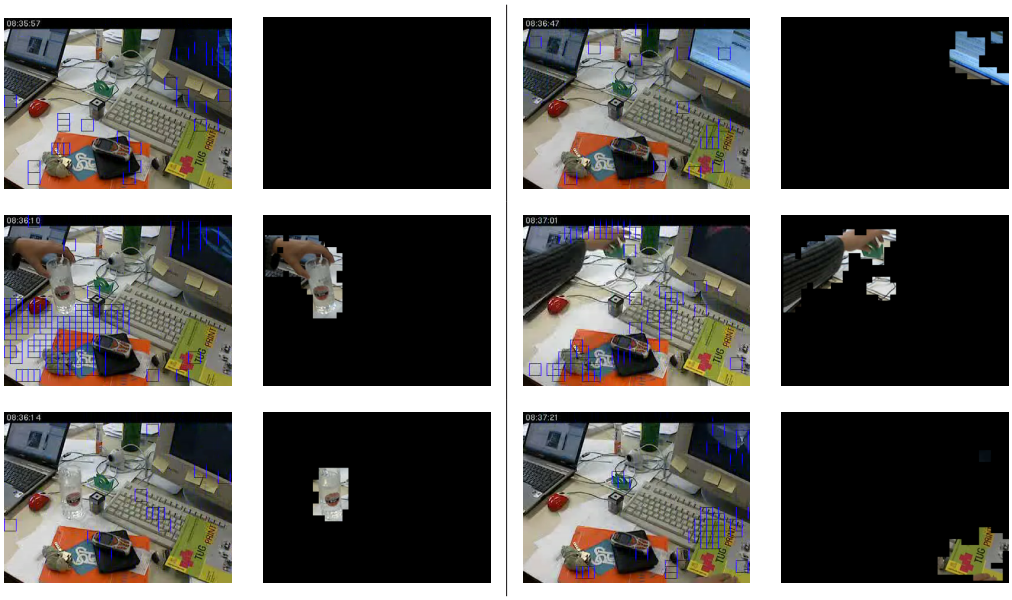
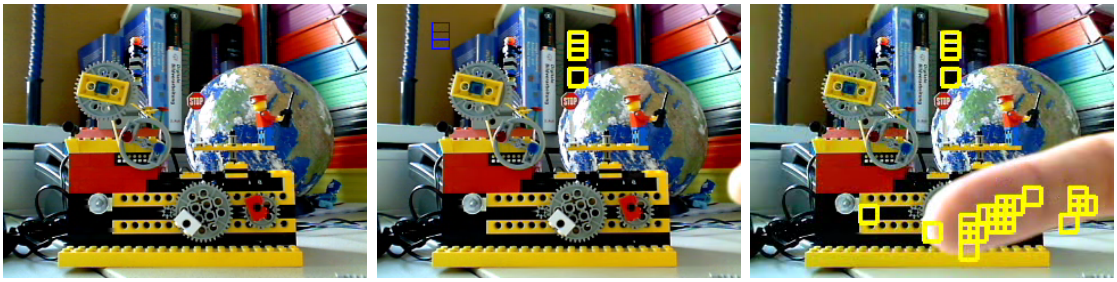


Figure 8.9: Evaluation of the background model. Each thin blue rectangle in the input image (first row) corresponds to classifiers, which are updated in order to learn the "allowed" changes. The regions, which cannot be modeled by the learned classifiers are related to the foreground (second row).

8.2.3 Discussion

The system has two phases: First, an initial learning stage where a separate classifier is built for all image patches assuming that all input images are positive examples (*i.e.*, correspond to allowed background variations). Later on, in order to be adaptive to the scene, new input images are analyzed and the background model is updated according to a given, yet not totally traceable, policy. Then, it ends up with three different thresholds, which have to be hand-tuned as well as some higher update policy, for instance, that neighboring patches are inhibited to be updated for a certain time (yet another irreproducible variable) when the current patch is considered as foreground. Further on, due to its analogy to self-learning which relies on a *direct* feedback of its own predictions, the approach tends to drift and ends up in not predictable states when running for a long time (*e.g.*, 24 hours a day, 7 days a week). This is shown in Figure 8.10. To sum up, the method shows high performance potentials but is only tediously, if not impossible, to be applied in practice. This work was therefore continued, see Section 9.1.

However, removing the update problem and applying the idea in an off-line learning task the proposed classifier-based background model achieves very good results. In fact, we applied it for defect detection, where a fixed set of aligned training images is



(a) (b) (c)

Figure 8.10: When running for a relative short period of time excellent results (a) can be obtained. However, when the system runs for a long time, many updates (marked by blue rectangles) are performed. Thus, the classifier starts to model everything as "allowed" foreground (parts of the finger in (c)). Unfortunately, the contrary happens as well, *i.e.*, the classifier might be too sensitive to changes and thus do not update (is looked) (b, c).

given, assuming, that most of them contains no defects.[4]. Some results are depicted in Figure 8.11.

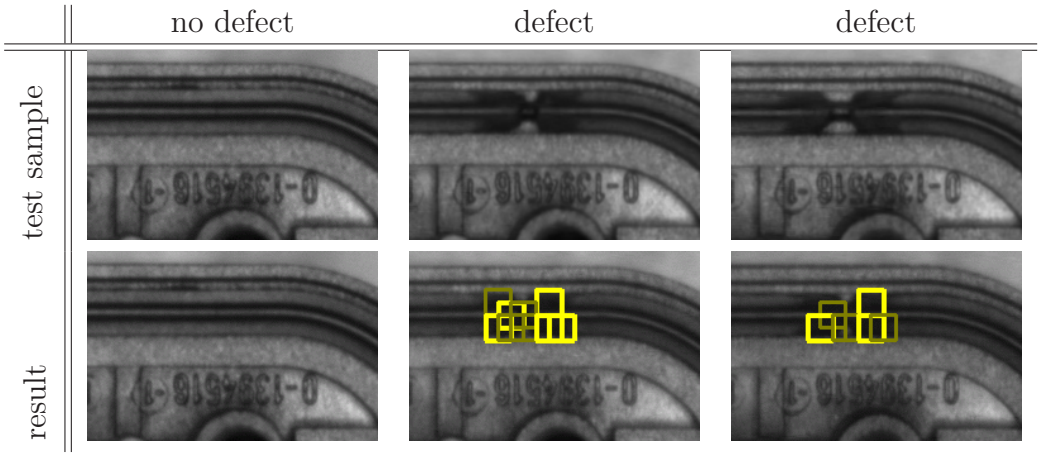


Figure 8.11: Defect detection using a classifier-grid. Each grid element corresponds to a boosted classifier trained off-line only from positive samples.

8.3 Summary

Self-learning approaches, shown in this chapter, directly depends on its own predictions. The robustness is only achieve over a certain time, as long as the labels are

[4]This was studied with an master thesis by Thomas Kenner [57].

correctly predicted. Due to the direct feedback, the effect is in general not manageable and further on self-reinforcing. The only possibility to make it robust is that the classifier itself is robust.

Chapter 9

Fixed Updates

In this chapter, we consider updates which are delivered by fixed rules, *i.e.*, the unlabeled sample is first given to a "teacher" which labels it. this is motivated in order to overcome or at least limit drifting, which usually happens when using dynamic update strategies. If the fixed rules can capture the problem well, the system will not drift. We show this approach on two applications, both with fixed, jet simple, hand designed update rules. First, for learning a robust background model, and second, we applied it for learning an pedestrian detector.

9.1 Robust Classifier-based Background Model[1]

The classifier-based background model, described in Section 8.2, is effective for describing highly dynamic scenes. The main drawback is its update strategy. Due to this dependency on its own predictions, the model performs quite well for a relatively short period of time but finally tends to learn foreground objects very quickly without offering any control on its temporal behavior. Furthermore, the cumbersome update strategy is highly scene dependent and, therefore, has to be hand-tuned every now and then. Therefore, we extend it by using an alternative version of the on-line learning algorithm which is controlled via temporal aspects and is resistant to outliers. In fact, we apply the time depended version of the *On-line Boosting for Feature Selection* (see Section 4.3), where one can specify a time constant Δt which controls fading memory of the internal parameters.

9.1.1 The Supervision

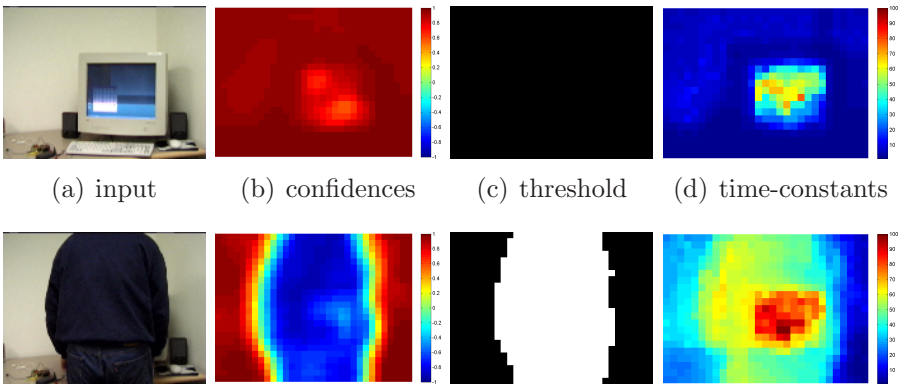
In order to get rid of the self-learning update strategy, we propose a fixed yet simple update strategy. Each classifier H_i with the corresponding patch $\mathbf{x}_{i,t}$ incorporates every new upcoming frame t as a positive example $(\mathbf{x}_{i,t}, +1)$. For automatically updating[2] the parameter Δt we choose a simple dynamic control system, taking the following observation into account: The time constant should be large enough in order to model dynamic behavior but still as small as possible in order to be highly sensitive to small background changes. If observing a static scene then every movement or change should be considered as foreground. On the other hand, observing a dynamic behavior, *e.g.*, leafs in the wind, this should be modeled, and therefore we have to increase the time constant up to a limit where it is still possible to model these dynamic backgrounds. Furthermore, we assume that the time constant should move smoothly. For each individual on-line classifier having its own Δt we use the following estimator

$$\Delta t_t = K_i \Delta t_{t-1} + K_p \hat{\Delta t} \quad \text{where} \quad \hat{\Delta t} = 1 - 0.5\,(1 + f(\mathbf{x}))\,, \tag{9.1}$$

where $K_i \in [0 \quad 1]$ assumes the smoothness constraint and $K_p > 0$ is multiplied by the current estimate $\hat{\Delta t}$, which is considered to be proportional to the confidence of the patch $\mathbf{x}$. As soon as the confidence changes dramatically, *i.e.*, a totally unknown intruder enters the scene, our control system tremendously increases Δt, which yields the implicit result that the harder the underlying scenario changes, the

[1] Adapted from H. Grabner, C. Leistner, and H. Bischof. Time Dependent On-line Boosting for Robust Backgroundmodeling. *In Proceedings International Conference on Computer Vision Theory and Applications, 2007 [32]*

[2] Please note, that of course one can also specify the time constant by hand in order to get a predefined background model for a specific application.



(a) input (b) confidences (c) threshold (d) time-constants

Figure 9.1: The first row depicts the test scene after $t = 100$ and the second row at $t = 251$, respectively. The second column shows the obtain confidences. As we assume the flickering monitor to be background, the confidences are quite high. In order to achieve such results our control system autonomously sets the Δt in the flickering area higher than in the non-dynamic rest of the scene.

higher the controller sets Δt and, thus, the longer it takes for a new object to "fade" into the background. Yet, smooth changes result in only small changes of Δt which let the system adapt to small background changes, *e.g.*, slightly changing lighting conditions. This allows us to autonomously model different dynamic movements and periods for each classifier patch without drifting into unpredictable states since only Δt and λ_0 as model parameters are changed but not the label of the updates.

The confidence of the classifier corresponds to the likelihood that the example corresponds to the background. In fact, we robustly detect outliers and mark them as unknown foreground objects.

9.1.2 Selected Experiments

For all of our experiments we use a classifier grid with a patch-size of 20×20 with an overlap of 75%. To compute the classifier we use only 15 selectors each using a set of 30 weak classifiers. The thus obtained grid of detectors is evaluated and updated whenever a new frame arises. In order to set the time constant Δt for each classifier we set $K_i = 0.95$ and $K_p = 10$.

Figure 9.1 shows the behavior of our proposed background model. The rows correspond to two different times and the columns show the input image and corresponding internal results, which are the achieved confidence map as well as the segmented foreground object, which is a simple threshold on the confidence map, and, in addition, the estimated time constant for each grid element. The sequence is taken from [126], which shows a sequence of a flickering screen. At the end a person enters

and fully occludes the monitor. In the first row each patch is able to model the background quite well using the on-line learned classifier (high confidence). In the region where the screen is located, the time-constant is automatically set to a higher value. This allows the patches located around the screen to model the flickering while still being able to detect the intruding object.

In [126], a test set for evaluating background subtraction methods was presented. It consists of seven video sequences, each addressing a specific canonical background subtraction problem. In the same paper, 10 different methods were compared using the test set. We tested our method against this test set and achieved the results shown in Figure 9.2. In addition to the segmentation, which is achieved by thresholding the classifier output by zero we depict as well the confidence map. This map profiles include more information which can be included in further analysis (*e.g.*, more sophisticated methods such as a mean shift-based clustering can be applied).

9.1.3 Discussion

Our proposed background model is based on fixed and simple update rules, which yield stable results over a long period of time. Finally, the period of time necessary for modeling regular and periodical scene behaviors does not have to be hand-tuned but autonomously adapts to the underlying problem. Because of the fixed update strategy we avoid (limit) drifting. Using the temporal dependency together with the weighting of λ_0 we achieved significantly improved results over the method proposed in Section 8.2, which uses a self-learning strategy.

Note, since using the fixed update rules and a fading memory, we are back to quite old but common used approaches for background modeling, *e.g.*, [121]. However, we benefit from the discriminative learning power.

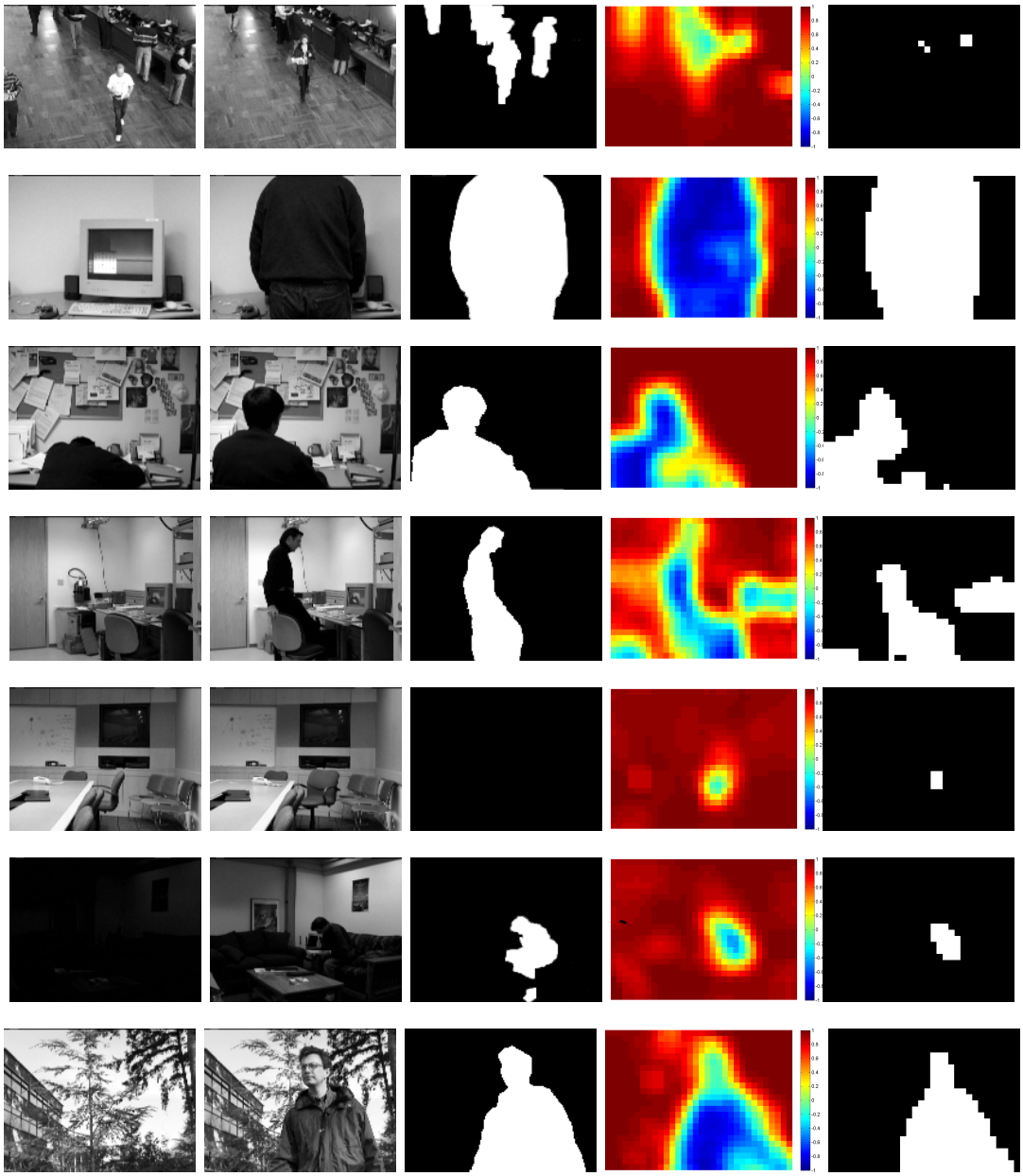


Figure 9.2: Detection results of our method for the test sequences presented in [126]. The first column shows the initial frame of each sequence, second column the test frame and the third column the hand segmented ground truth. In the last two columns our results (the real valued confidence map, a binary segmentation achieved be zero thresholding) are depicted. Note that the performance could be easily increased by analyzing the confidences more closely.

9.2 Grid-based Object Detection[3]

By using a state of the art person detector many false positives are returned and persons are missed. Hence, there is still a long way to go in order to obtain a generic detector with satisfactory performance. But do we really need a generic detector for visual surveillance? Consider, that we have fixed mounted cameras looking always at the same scene, which is a considerable simplification for the task of person detection. Hence, we do not need to solve the generic person detection task, we just need to solve it for that particular scene. Due to a simpler problem this yields also to more compact and, hence, faster as well as less memory consuming classifiers. That is the reason why people have started to look at approaches that can train a person detector for a particular scenario on-line. By limiting the detection task to a specific scene the task becomes easier and less training samples are required. On the other hand on-line unsupervised learning methods tend to wrong updates, which reduces the performance of the detector. The detector might start to drift and would end in an unreliable state.

We proposed to combine background and appearance based models. Therefore, we sub-divide the input images into small overlapping blocks and to train and to maintain a person detector on-line for all of these patches. Since the task of each detector is to detect a person in only one specific patch and at a specific time the complexity of the person detection task is significantly reduced. Hence, we can apply quite simple and fixed update rules for updating the classifiers. This keeps the classifier stable and limits the drifting problem. This is an essential property for practical applications which run for a long time (24 hours a day, 7 days a week).

9.2.1 The Supervision

On the one hand we have a classifier based background model and on the other hand an appearance based object detector, both are formulated as binary classification task. The idea is to combine these two methods to get an improved object detector for the particular scene.

The main goal was to define an update strategy that does not suffer from the drifting problem. The key idea is to reduce the complexity of the pedestrian detection problem such that a very simple and fixed update strategy can be applied. In the following we discuss the chosen update strategy for a single patch (one classifier H_i). Since updates should be generated without a feedback of $H_{i,t-1}$ we make use of the following (very simple) observations:

[3] Adapted from H. Grabner, P. Roth, and H. Bischof. Is Pedestrian Detection Really a Hard Task? *In Proceedings IEEE International Workshop on Performance Evaluation of Tracking and Surveillance,2007.* [36]

Positive updates: Given a set of positive (hand) labeled examples $\mathcal{X}^+$. Then, using $(\mathbf{x}, +1), \quad \mathbf{x} \in \mathcal{X}^+$ to update the classifier is a correct positive update. The set can by quite small; in the extremal case (as we will show in the experiments) it contains only one positive sample. The only assumption is that $\mathcal{X}^+$ is a representative set. Roughly speaking, each possible appearance should be captured by this subset.

Negative updates: The probability that a person is present on patch $\mathbf{x}_i$ is given by $P(\mathbf{x}_i = \text{person}) = \frac{\#p_i}{\Delta t}$, where $\#p_i$ is the number of persons entirely present in a particular patch within the time interval Δt. Thus, the negative update with the current patch $(\mathbf{x}_{i,t}, -1)$ is correct most of the time (wrong with probability $P(\mathbf{x}_i = \text{person})$). The probability of a wrong update for this particular image patch is indeed very low.

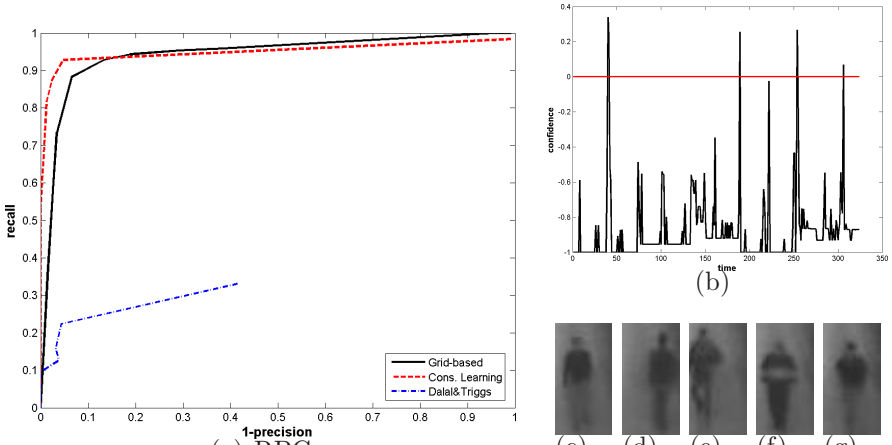
In fact, we maintain a small set (as small as one) of positives samples whereas the negative samples are directly drawn from the image sequence. In the experiments we compared the proposed method to state-of-the-art methods. Since we obtain competitive detection results we showed that for a specific surveillance task even a less sophisticated approach would yield comparable results.

By using these update rules we avoid the dependencies between the updates and the current model. Since the positive updates are per definition always correct the only remaining problem is that occasionally false negative updates may be carried out. Hence, the applied on-line learning method (i) must cope with some (low) label noise, and (ii) must have fading memory (forgetting), which exactly is achieved by the time-depending version of *On-line boosting for Feature Selection* (see Section 4.3).

9.2.2 Selected Experiments

Based on the approximated size of a person and an estimated ground-plane, a grid of detectors using an overlap-rate of 90% is initialized. To compute the grid-based classifier we use only 10 selectors each using a set of 20 weak classifiers, which reflects the simplicity of the task. The thus obtained grid of detectors is evaluated and updated whenever a new frame arises. The set of positive samples was reduced to a single image that was obtained by averaging of approximative 100 images of persons.

We compare the proposed grid-based detector approach to other approaches by analyzing the precision-recall curves (see Figure 9.3 (a)). For the grid-based detectors the evaluation was performed on-line; all detection that are reported during the evaluation/updates procedure are included directly into the statistics. For other methods a pre-trained classifier was evaluated on the test sequence. The Dalal and Triggs detector performs worst among the tested methods. This is not surprising



(a) RPC (b)

(c) 41 (d) 189 (e) 222 (f) 255 (g) 306

Figure 9.3: RPC for the Toy Example test sequence (a) and confidence of a specific grid-based detector over time (b) and the corresponding images (c)-(g).

since a generic detector does not include any scene information. In contrast, the *Conservative Learning* [108] (see also Section 7.3) detector yields high recalls producing only a small number of false positives in order to adapt to the particular scene. But a similar performance can be obtained by applying our simple grid-based approach. To avoid multiple detections non-maxima suppression was applied. Detection results are depicted in Figure 9.4.

Additionally, we show how a single detector for a single block is evaluated. Therefore, the response (confidence) of the classifier is plotted over time which is shown in Figure 9.3 (b). It can be seen that the response is increased whenever a person or a part of a person is present in the patch. A detection is reported if the confidence is above some threshold which is usually set to zero. As can be seen from Figure 9.3 (c)-(g) only persons are detected.

9.2.3 Discussion

The complexity of the detection problem can be reduced such that a single detector has only to distinguish between a single background patch and the appearance of a person, which allows us to make use of fixed update rules. By combining both approaches we finally get a pedestrian detection system that is stable even when running over a long period of time. In fact, we represent the background with high accuracy for each specific grid element and the positive samples provide a suitable "threshold" for the decision. The more samples are used the better the performance can be expected. Further, one can think of of modeling the positive distribution off-line and just on-line estimate the negative one.

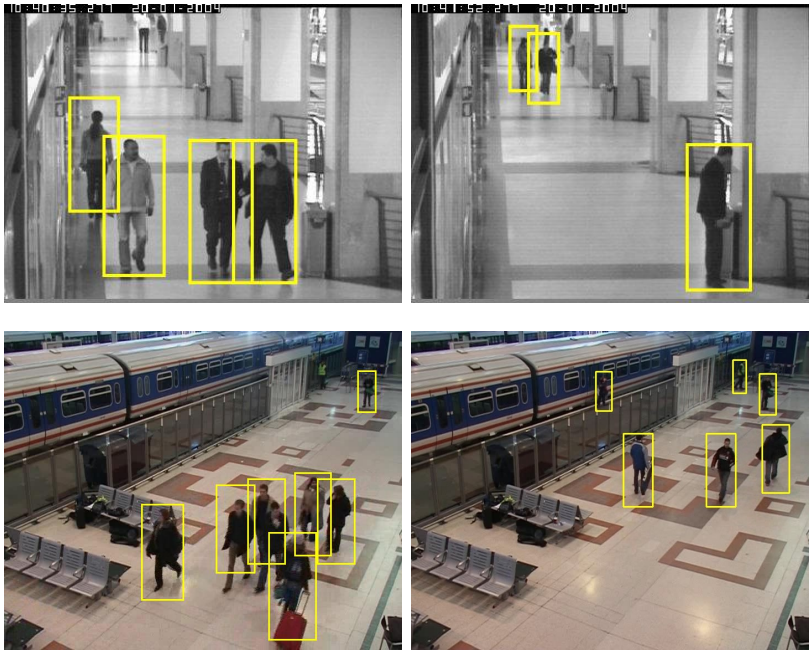


Figure 9.4: Detection results of our proposed grid-based pedestrian detector on the two sequences Caviar (first row) and PETS 2006 (second row).

9.3 Summary

Fixed update rules, shown in this chapter, are immune to drifting per definition, since no direct feedback exists. The classifier is updated according to the decisions of a "teacher", *e.g.*, via hand designed rules taken from an expert. However, the classifier is still adaptive and so it can generalize better by focusing on the current sub-problem. However, the open question is, how do come up with these rules and how "good" they have to be in order to properly solve the problem. This is related to co-learning [10], whereas one classifier is fixed. Thus, the assumption [6], that a classifier (the teacher) should be "never confident but wrong" remains.

Chapter 10

Conclusion and Future Work

In this thesis, we proposed the novel *On-line Boosting for Feature Selection* algorithm. The algorithm performs on-line feature selection using binary labeled samples. Its generality, suitability, and efficiency for computer vision was demonstrated on various applications, *e.g.*, tracking, continuously improving detectors as well as background modeling. This was made possible since all applications can be (i) formulated as binary classification problem and (ii) unlabeled data can be included into the model by some sort of supervision. Extensive evaluations have been done, whereas excellent results can be obtained. Even combinations are possible, like detection, tracking and recognition [42] or detection, tracking and background modeling [37]. The combined approaches benefit essentially from the fact that all tasks can be formulated with the same algorithm (off-line and on-line boosting) and the same features (which additionally can be shared and have to be calculated only once).

However, for most application one has to tackle with unlabeled data. Since the algorithm needs labels, these have to be generated somehow in order to make updates. The second part of this thesis shows different strategies, which are nowadays, unfortunately, not clearly understood in detail. Summarizing, most of the approaches focus on verification or self-learning. Verification methods seems powerfully if a lot of data is available and thus it is feasible that only a few conservative updates are made. However, if this is not the case (*e.g.*, in fast changing environments) may no updates will be performed at all. Hence, no adaption is possible. Self-learning can be used, but it suffers extremely from the drifting problem. Drifting is avoided per definition, using fixed update rules. However, these usually hand designed rules have to be sufficient enough to cope with the problem.

In the following sections, we give an outlook and preliminary results of ongoing work to continue the ideas described in this thesis. First, we briefly show how the on-line

boosting algorithm can be made more robust to feature level noise. The second section addresses label noise. We extend the algorithm in order to directly deal with unlabeled data, *i.e.*, we turn the supervision into the updating process of the algorithm.

10.1 Robustness of the Classifier

Boosting is a discriminative method, which focuses on the hard examples to learn. This is the nature of boosting, but it is also the reason why it is in the common formulation very sensitive to noise. It achieves a hard margin [101] decision and therefore overfits to the training samples, even when they are mislabeled. Therefore, we need more robust models: Robust feature selection by combining (embedding) generative information from the patches. If one has a classifier which is very robust (can be learned under random noise) the label generation process can be very weak. We investigated some effort in that direction [35], where we focus on feature level noise. The results look promising, nevertheless, up to now only off-line learning is considered. However, it seems straight forward to to it on-line as well.

10.2 On-line Semi-supervised Boosting[1]

The main idea is to formulate the on-line training in a semi-supervised manner. In fact, labeled data serves as a prior and the unlabeled data is then used to update the classifier. This allows to drift but limit it to a certain amount. In other words, the prior classifier (the confidence of the prior classifier) tells the on-line classifier how it can be adapted. It is essential to have a honest prior, *i.e.*, it can be wrong but never confident but wrong. The essential point is that the *SemiBoosting* theory tells one how the prior should be included. As effect, we can directly include unlabeled data and are beyond the point of 100% positive or negative updates. Only a few lines of code have to be adapted in the algorithm. In the following we show the principle behavior on the task of visual tacking. Comparing to Figure 1.5 from Section 3.4 we now yield the following Figure 10.1.

We illustrate details of our tracker on frontal faces. As prior classifier and for initialization of the tracking process we took the default frontal face detector from *OpenCV Version 1.0*[2], which delivers state-of-the-art results. This demonstrates that we can use any prior in our method. The primary focus of the experiments

[1] Adapted from H. Grabner, C. Leistner, and H. Bischof. Semi-supervised On-line Boosting for Robust Tracking. *In Proceedings European Conference on Computer Vision (ECCV)*, 2008. [33]

[2] http://sourceforge.net/projects/opencvlibrary/, (March 16, 2008)

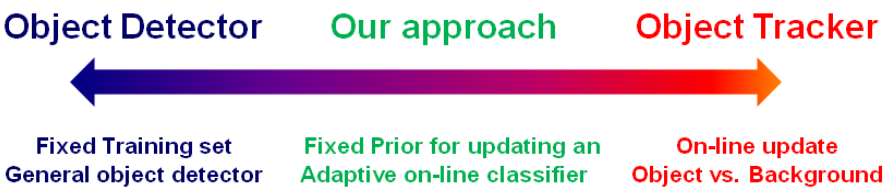


Figure 10.1: Detection and tracking in principle can be viewed as the same problem, depending on how fast the classifier adapts to the current scene. On the one side a general object detector (*e.g.*, [130]) is located and on the other side a highly adaptive tracker (*e.g.*, [31]). Our approach is somewhere in between, benefiting from both approaches: (i) be sufficiently adaptive to new appearance and lightning changes, and the simplification of object vs. background and (ii) limit (avoid large) drifting by keeping prior information from the object.

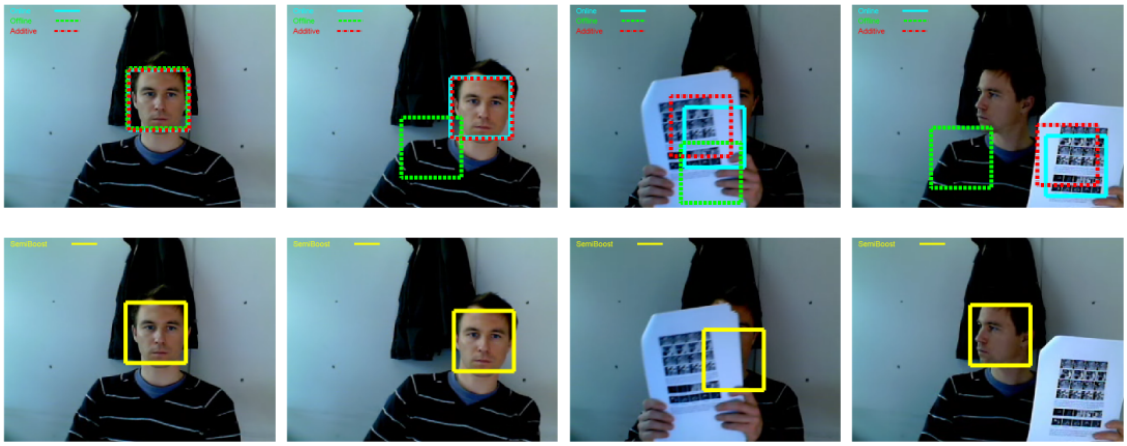


Figure 10.2: Tracking a face in an image sequence under various appearance changes (*i.e.*, occlusion, movement, out of plane rotation, *etc.*). The first row illustrates three different types of update strategies for the tracker, *i.e.*, (i) on-line boosting (cyan), (ii) prior classifier (red), and (iii) a heuristic combination of (i) and (ii) using the sum-rule, *i.e.*, $0.5(H^P(\mathbf{x}) + H(\mathbf{x}))$ (green). The second row shows the *SemiBoost* tracker.

is to compare the *SemiBoost* tracker to other update strategies. As can be seen from Figure 10.2, our approach (second row) significantly outperforms the on-line booster, the prior classifier and an heuristic combination of prior and on-line booster (first row). It shows that we can adapt to appearance changes, whereas the frontal face detector of *OpenCV* fails on side looking faces.

In Figure 10.3, we compare our method to the former on-line boosting approach on various tracking scenarios. First, as can be seen in row 1, we are still able to handle challenging appearance changes of the object. Our approach performs similar to the former on-line tracker up to the third subfigure in row 2, where both get lost. Yet, in contrast to the previous proposed method, as soon as the object becomes visible

again it is re-detected by the *SemiBoost* tracker (using the a priori knowledge) while the on-line boosted tracker meanwhile has adapted itself to a completely different region which it finally tries to track. Row 3 of Figure 10.3 depicts tracking during a fast movement. Due to some incorrect updates and the self-learning strategy of the on-line boosting tracker it may happens that the tracker loses the target and focuses on another part while the semi-supervised tracker is able to re-detect the object. An extremal case is shown in row 4, where we remove the object from the scene. If the object is present again and thanks to the fixed prior our proposed approach has not forgotten the appearance as it is the case for the other tracker and snap to the object again. The next experiment (row 5) focuses on the long term behavior. Thus, we chose to track a non-moving object in a static scene for about 1 hour. In order to emphasize the effect on this relative short timescale we choose low illumination conditions. While our proposed tracker stays at the object the on-line booster starts drifting away. The reason for that is the accumulation of errors. The final experiment shows a special case of drifting as depicted in the last row of Figure 10.3. Two very similar objects are put together in the scene. Since the pure on-line tracker has not the additional prior information it is very likely that it is unstable and may switch to another object.

10.3 Resume

On-line learning is suitable and sometimes even necessary for many applications. – Really interesting tasks were those where unlabeled data has to be incorporated. – However, then one has to deal with the drifting problem.

I personally think, that on should see on-line learning **and** *the label generating process (supervision) as* **ONE** *problem. Maybe the approach mentioned in the last section is a step in this direction.*

This approach can be very beneficial for other tasks as well. For instance, we did preliminary experiments for the grid-based person detector, as described in Section 9.2. The prior information so far was coded by the labeled positive data set. Now, we can train a person classifier, which is as good as possible and then use this as prior in order to build an on-line classifier which just corrects its errors. Nevertheless, a final remark concerning the necessity of on-line learning:

I am quite sure, that for many practical applications one can relax on-line learning to incremental learning, i.e.*, save some "good" old samples for later access. Moreover, may it is even possible to postpone decisions,* i.e.*, use also upcoming examples, which allows to overcome the causality assumption. Robust statistics can then be applied in order to find outliers, correct errors, limit drifting, and thus better results can be expected.*

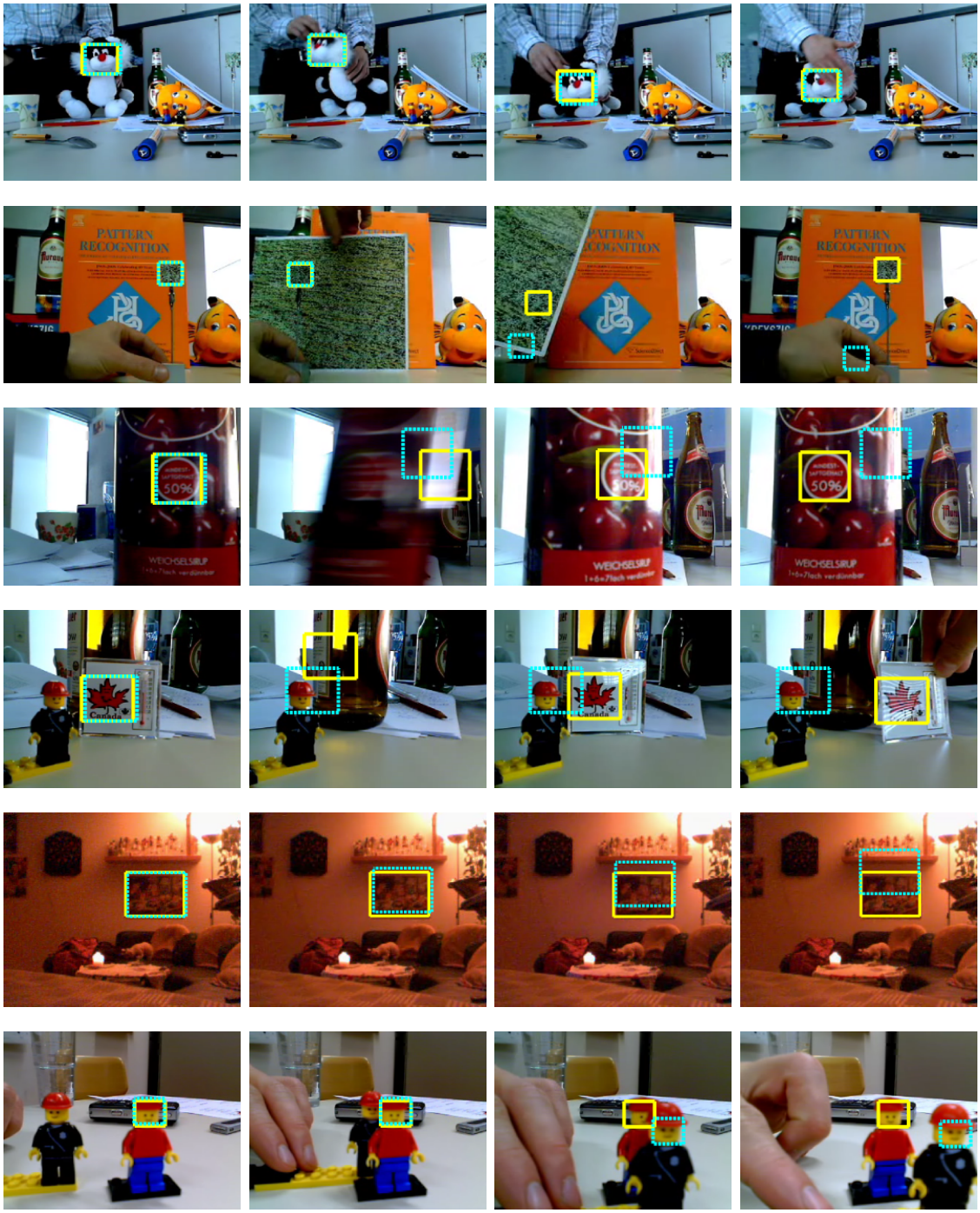


Figure 10.3: Comparisons of our proposed *SemiBoost* tracker (yellow) and the previously proposed on-line tracker (dotted cyan). Our approach is still able to adapt to appearance changes while drifting is limited.

Appendix A

Off-line Boosting

This analysis is based on [25, 26]. In order to derive the *AdaBoost* Algorithm (see Algorithm 1 in Section 2.3.1), we want to minimize the training error

$$\mathbb{E}_{\mathcal{S}} = \frac{1}{L}\sum_{l=1}^{L}\begin{cases} 1 & H(\mathbf{x}_l) \neq y_l \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

on the fixed training set $S = \{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_L, y_L) | y_i \in \{-1, +1\}\}$ with L samples, which can be re-written using the definition of $H(\mathbf{x})$ and concerning $y \in \{-1, +1\}$ as

$$\mathbb{E}_{\mathcal{S}} = \frac{1}{L}\sum_{l=1}^{L}\begin{cases} 1 & \sum_{n=1}^{N}\alpha_n h_n(\mathbf{x}_l) y_l \leq 0 \\ 0 & \text{otherwise} \end{cases} . \tag{A.2}$$

By using the exponential loss function $\text{loss}(\mathbf{x}, y)) = \exp(-yf(\mathbf{x}))$ we get an upper bound (since $\exp(-z) \geq 1,\ z \leq 0$)

$$\mathbb{E}_{\mathcal{S}} \leq \frac{1}{L}\sum_{l=1}^{L}\exp\left(-y_l\sum_{n=1}^{N}\alpha_n h_n(\mathbf{x}_l)\right). \tag{A.3}$$

Instead of doing a global minimization, we suppose that the weak classifiers $h_1(\mathbf{x}), ..., h_{n-1}(\mathbf{x})$ are already calculated and fixed, as are their coefficients $\alpha_1, ..., \alpha_{n-1}$. This means, that we perform a greedy search for the next best classifier, added to the whole ensemble. In other words: *AdaBoost* minimizes the exponential loss criterion (Equation A.3) via a forward-stage wise adaptive approach [26]. Thus, we split of the n-th weak classifier, which is to be added:

$$(\alpha_n, h_n) = \arg\min_{\alpha_n, h_n}\sum_{l=1}^{L}\exp\left(-y_l\sum_{i=1}^{n-1}(\alpha_i h_i(\mathbf{x}_l)) - y_l\alpha_n h_n(\mathbf{x}_l)\right). \tag{A.4}$$

This can be expressed as

$$(\alpha_n, h_n) = \arg\min_{\alpha_n, h_n} \sum_{l=1}^{L} w_{n,l} \exp\left(-y_l \alpha_n h_n(\mathbf{x}_l)\right), \tag{A.5}$$

where the coefficients

$$w_{n,l} := \exp\left(-y_l \sum_{i=1}^{n-1} \alpha_i h_i(\mathbf{x}_l)\right) \tag{A.6}$$

can be viewed as constants because we are optimizing α_n and $h_n(\mathbf{x})$. If we have found $h_n(\mathbf{x})$ and α_n we can easily obtain the update rule for $w_{n+1,l}$ by doing the recursive step (note, the information of the previous learned classifiers is captured (encoded) only by the weights $w_{n-1,l}$):

$$\begin{aligned} w_{n+1,l} &= \exp\left(-y_l \sum_{i=1}^{n} (\alpha_i h_i(\mathbf{x}_l))\right) = \\ &= \exp\left(-y_l \sum_{i=1}^{n-1} (\alpha_i h_i(\mathbf{x}_l))\right) \exp\left(-y_l \alpha_n h_n(\mathbf{x}_l)\right) = \\ &= w_{n,l} \cdot \begin{cases} \exp(-\alpha_n) & h_n(\mathbf{x_l}) = y_l \\ \exp(\alpha_n) & h_n(\mathbf{x_l}) \neq y_l \end{cases}. \end{aligned} \tag{A.7}$$

The solution to Equation A.5 can be obtained in two steps (find the weak classifier $h_n(\mathbf{x})$ and the corresponding weight α_n). Therefore, we re-write Equation A.5 with respect to the correct ($y_l h_n(\mathbf{x}_l) = 1$) and the incorrect ($y_l h_n(\mathbf{x}_l) = -1$) classified samples

$$\exp(\alpha_n) \sum_{l:h_n(\mathbf{x}_l)\neq y_l} w_{n,l} + \exp(-\alpha_n) \sum_{l:h_n(\mathbf{x}_l)=y_l} w_{n,l} = \tag{A.8}$$

$$= (\exp(\alpha_n) - \exp(-\alpha_n)) \sum_{l=1}^{L} w_{n,l} \cdot \begin{cases} 1 & h_n(\mathbf{x}) \neq y_l \\ 0 & \text{otherwise} \end{cases} + \exp(-\alpha_n) \sum_{l=1}^{L} w_{n,l}. \tag{A.9}$$

When minimizing this with respect to $h_n(\mathbf{x})$ we see that the second term is constant and thus this is equivalent to minimizing the weighted error of the training examples for the new classifier, because the overall normalization factor does not affect the location of the minima. Which is the classifier

$$h_n(\mathbf{x}) = \arg\min_{h_n} \sum_{l=1}^{L} w_{n,l} \cdot \begin{cases} 1 & h_n(\mathbf{x}) \neq y_l \\ 0 & \text{otherwise} \end{cases} \tag{A.10}$$

that minimizes the weighted error rate in the prediction y.

Similar, Equation A.5 is minimized with respect to α_n. We define the normalized weighted error e_n of the hypothesis $h_n(\mathbf{x})$ as

$$e_n := \frac{\sum_{l:h_n \neq y_l} w_{n,l}}{\sum_{l=1}^{L} w_{n,l}}. \tag{A.11}$$

Note, this can also be done by normalizing the weights $w_{n,l}$ itself such that $\sum_{l=1}^{L} w_{n,l} = 1$. Therefore, the initial weights are defined by $w_{1,l} = \frac{1}{L},\ l = 1, ...L$. Substituting Equation A.11 into Equation A.8 we get

$$\alpha_n = \arg\min_{\alpha_n} \left(\exp(\alpha_n)e_n + \exp(-\alpha_n)(1 - e_n)\right), \tag{A.12}$$

which is minimized by taking the derivative with respect to α_n and setting it to zero

$$\begin{aligned} \frac{\partial}{\partial \alpha_n} &= \exp(\alpha_n)e_n + \exp(-\alpha_n)(1 - e_n)(-1) = 0 \\ \exp(\alpha_n)e_n &= \exp(-\alpha_n)(1 - e_n) \\ \alpha_n &= \frac{1}{2} \ln\left(\frac{1 - e_n}{e_n}\right). \end{aligned} \tag{A.13}$$

Summarizing, all the formulas presented in the *AdaBoost* algorithm are derived.

A.1 Training Error Theorem

Substituting α_n back into Equation A.12 the result of the minimization in step n is

$$\begin{aligned} &\exp\left(\frac{1}{2}\ln\left(\frac{1-e_n}{e_n}\right)\right)e_n + \exp\left(-\frac{1}{2}\ln\left(\frac{1-e_n}{e_n}\right)\right)(1-e_n) = \\ &= \sqrt{\frac{1-e_n}{e_n}}e_n + \sqrt{\frac{e_n}{1-e_n}}(1-e_n) = 2\sqrt{e_n(1-e_n)}. \end{aligned} \tag{A.14}$$

Let us write the error $e_n = \frac{1}{2} - \gamma_n$. Since a hypothesis that guesses each instances class at random has an error rate 0.5 (on binary problems). Thus, γ_n measures how much better than random are $h_n(\mathbf{x})$s predictions.

$$2\sqrt{e_n(1-e_n)} = \sqrt{1-4\gamma_n} \leq \sqrt{-4\gamma_n^2} \leq \exp\left(\sqrt{-4\gamma_n^2}\right) = \exp(-2\gamma_n^2) \tag{A.15}$$

As can be seen from Equation A.5 the overall approximation (strong classifier) is

then updated by the calculated weak classifier

$$\sum_{i=1}^{n} \alpha_i h_i(\mathbf{x}) = \sum_{i=1}^{n-1} \alpha_i h_i(\mathbf{x}) + \alpha_n h_n(\mathbf{x}). \tag{A.16}$$

After N boosting iterations and combining with Equation A.3 we yield the *AdaBoost* training error theorem:

$$\begin{aligned} \mathbb{E}_{\mathcal{S}} &\leq \frac{1}{L} \sum_{l=1}^{L} w_{1,l} \prod_{n=1}^{N} \exp(-y_l \alpha_n h_n(\mathbf{x}_l)) \leq \\ &\leq \prod_{n=1}^{N} \exp\left(-2\gamma_n^2\right) = \exp\left(-2 \sum_{n=1}^{N} \gamma_n^2\right). \end{aligned} \tag{A.17}$$

Hence, the training error drops exponentially with the number of weak classifiers n, if the error of each weak classifier is better than random guessing ($e_n < 0.5$, or equivalent $\gamma_n > 0$, $n = 1, ..., N$).

A.2 A Statistical view of Boosting

Following Friedman *et al.* [26] we are interested in the expected value over the binary learning problem with is defined of samples from the distribution $P : \mathcal{X} \times \{1, -1\}$.

$$\mathbb{E}(\exp(-yH(\mathbf{x}))) = P(y = 1|\mathbf{x}) \exp(-H(\mathbf{x})) + P(y = -1|\mathbf{x}) \exp(H(\mathbf{x})) \tag{A.18}$$

Since boosting minimizes this exponential loss by calculating $H(\mathbf{x})$ we take the derivative and set it to zero

$$\frac{\partial \mathbb{E}(\exp(-yH(\mathbf{x})))}{\partial H(\mathbf{x})} = -P(y = 1|\mathbf{x}) \exp(-H(\mathbf{x})) + P(y = -1|\mathbf{x}) \exp(H(\mathbf{x})). \tag{A.19}$$

Using the fact $P(y = -1|\mathbf{x}) + P(y = 1|\mathbf{x}) = 1$ we get

$$P(y = 1|\mathbf{x}) = \frac{\exp(H(\mathbf{x}))}{\exp(H(\mathbf{x})) + \exp(-H(\mathbf{x}))} \tag{A.20}$$

If we are interested only in the decision $P(y = 1|\mathbf{x}) > 0.5$ this is equivalent to $H(\mathbf{x}) > 0$. As can be easily seen the boosted classifier minimizes the log-likelihood rate

$$H(\mathbf{x}) = \frac{1}{2} \log\left(\frac{P(y = 1|\mathbf{x})}{P(y = -1|\mathbf{x})}\right). \tag{A.21}$$

Appendix B

On-line Boosting

In this section we want to give another derivation of the on-line *AdaBoost* algorithm as proposed by Oza and Russell [92]. Note, this is not a proof, it is much more an argumentation, which is inspired by the work of Friedman *et al.* [26].

We want to minimize the training as in Equation A.1. Since we do not have access to all the examples at once, we cope with a sequence of samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_t, y_t)$ and are not allowed to re-access them. However we can save some internal parameters. Thus, in on-line learning the cumulative mistakes are minimized and the objective function, Equation A.1, gets

$$\mathbb{E}_{\mathcal{S}} = \sum_{t=1}^{T} \begin{cases} 1 & H_t(\mathbf{x}_t) \neq y_t \\ 0 & \text{otherwise} \end{cases} . \tag{B.1}$$

We get hypothesis $H_1, \ldots, H_t$ which is the best model up to time t and encodes the knowledge from the examples seen so far. In the following, we show how a new sample can be used to update (i) the weak classifiers and (ii) the voting weight α.

Similar as in the off-line case, by using the exponential loss (Equation A.5) gets

$$(\alpha_n, h_n) = \arg\min_{\alpha_n, h_n} \sum_{t=1}^{T} \lambda_{n,t} \exp(-y_t \alpha_n h_n(\mathbf{x}_t)) \tag{B.2}$$

with the coefficient

$$\lambda_{n,t} := \exp\left(-y_t \sum_{i=1}^{n-1} \alpha_{i,t} h_{i,t}(\mathbf{x}),\right) \tag{B.3}$$

which is the importance of the sample $\mathbf{x}_t$ up to the n-th weak classifier. This is closely related to $w_{n,l}$ in the off-line case as defined in Equation A.6. Therefore, the

update step is equivalent and we get

$$\lambda_{n+1,T} = \lambda_{n,T} \cdot \begin{cases} \exp(-\alpha_{T,n}) & h_{T,n}(\mathbf{x}_T) = y_T \\ \exp(-\alpha_{T,n}) & h_{T,n}(\mathbf{x}_T) \neq y_T \end{cases} \tag{B.4}$$

In order to train/update the weak classifier we get according to Equation A.10

$$h_n(\mathbf{x}) = \arg\min_{h_n} \sum_{t=1}^{T-1} \lambda_{n,t} \cdot \begin{cases} 1 & h_n(\mathbf{x}) \neq y \\ 0 & \text{otherwise} \end{cases} + \lambda_{n,T} \cdot \begin{cases} 1 & h_n(\mathbf{x}) \neq y \\ 0 & \text{otherwise} \end{cases} \tag{B.5}$$

The first term is fixed, because we cannot re-access the old training samples. But they are used to train a classifier h_t we have to update this classifier with respect to λ.

Similarly, the update of α can be estimated. According to Equation A.11 we define the estimated error of the examples seen so far and the hypothesis h_n by

$$\hat{e} = \frac{\sum_{t:h_{t,n}(\mathbf{x}_t)\neq y_t} \lambda_{n,t}}{\sum_{t=1}^{T} .\lambda_{n,t}} \tag{B.6}$$

This error can be continuously updated using the sum of correctly classified samples so far λ^{corr} and wrongly classified examples λ^{wrong} (both initialized by 1) by

$$\hat{e_n} = \frac{\lambda^{wrong} + \delta(h_n(\mathbf{x}) \neq y)\lambda}{\lambda^{wrong} + \lambda^{corr} + \lambda}. \tag{B.7}$$

Thus, Equation A.12 can be rewritten

$$\alpha_n = \arg\min_{\alpha} \left(\exp(\alpha_n)\hat{e_n} + \exp(-\alpha_n)(1 - \hat{e_n})\right) \tag{B.8}$$

and the result be solving it (same as in Equation A.13) is

$$\alpha_n = \frac{1}{2} \ln\left(\frac{1 - \hat{e_n}}{\hat{e_n}}\right). \tag{B.9}$$

Summarizing both, the weak classifiers as well as the voting weights can be updated and all the steps from the algorithm are derived.

Note, the only difference in our algorithm is the updates of the weights (Equation B.4) for the next weak classifier. By plugging in the definition of $\alpha_{n,T}$ we get, in contrast to Oza and Russell [92] (see Equation 3.5)

$$\lambda_{n+1,T} = \lambda_{n,T} \cdot \begin{cases} \sqrt{\frac{1-e}{e}} & h_{T,n}(\mathbf{x}_T) = y_T \\ \sqrt{\frac{e}{1-e}} & h_{T,n}(\mathbf{x}_T) \neq y_T \end{cases}. \tag{B.10}$$

Further, by using fading memory (as described in Section 4.3) this can be easily included in our argumentation, by weighting the example in order to update the weak classifier (Equation B.5) as well as weighting the error (Equation B.6).

Appendix C

Publications

During my work at the Institute for Computer Graphics and Vision at Graz University of Technology the following paper were published. For the sake of completeness these publications are reported in the following in an inverse chronological order. This thesis is mainly based on a subset of those papers.

Journals

[1] H. Grabner, T.T. Nguyen, B. Gruber, and H. Bischof. On-line boosting-based car detection from arial images. *ISPRS Journal of Photogrammetry & Remote Sencing*, 63(3):382–396, 2007.

Major Conferences

[1] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *Proceedings European Conference on Computer Vision (ECCV)*, 2008.

[2] A. Saffari, H. Grabner, and H. Bischof. SERBoost: Boosting with expectation regularization. In *Proceedings European Conference on Computer Vision (ECCV)*, 2008.

[3] C. Leistner, H. Grabner, and H. Bischof. Semi-supervised boosting using visual similarity learning. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[4] M. Grabner, H. Grabner, and H. Bischof. Learning features for tracking. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[5] H. Grabner, P.M. Roth, and H. Bischof. Eigenboosting: Combining discriminative and generative information. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[6] H. Grabner and H. Bischof. On-line boosting and vision. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 260–267, 2006.

Other Reviewed Conferences and Workshops

[1] H. Grabner, J. Sochman, H. Bischof, and J. Matas. Training sequential on-line boosting class for visual tracking. In *Proceedings International Conference on Pattern Recognition (ICPR)*, 2008.

[2] C. Leistner, P. Roth, H. Grabner, H. Bischof, A. Stratzer, and B. Rinner. Visual on-line learning in distributed camera networks. In *Proceedings International Conference on Distributed Smart Cameras*, 2008.

[3] G. Schall, H. Grabner, M. Grabner, P. Wohlhart, D. Schmalstieg, and H. Bischof. 3d tracking in unknown environments using on-line keypoint learning for mobile augmented reality. In *In Proceedings Workshop on Visual Localization for Mobile Platforms*, 2008.

[4] P. Roth, H. Grabner, C. Leistner, M. Winter, and H. Bischof. Interatctive learning a person detector: Fewer clicks - less frustration. In *Proceedings Workshop of the Austrian Association for Pattern Recognition (AAPR)*, 2008.

[5] H. Grabner, C. Leistner, and H. Bischof. Time dependent on-line boosting for robust backgroundmodeling. In *Proceedings International Conference on Computer Vision Theory and Applications*, 2007.

[6] H. Grabner, P.M. Roth, and H. Bischof. Is pedestrian detection realy a hard task? In *Proceedings IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2007.

[7] St. Kluckner, G. Pacher, H. Grabner, H. Bischof, and J. Bauer. A 3d teacher for car detection in aerial images. In *Proceedings ICCV Workshop on 3D Representation for Recognition*, 2007.

[8] M. Grabner, H. Grabner, J. Pehserl, P. Korica-Pehserl, and H. Bischof. Flea, do you remember me? In *Proceedings Asian Conference on Computer Vision (ACCV)*, pages 657–666, 2007.

[9] T.T. Nguyen, H. Grabner, B. Gruber, and H. Bischof. On-line boosting for car detection from aerial images. In *Proceedings IEEE International Conference on Research, Innovation and Vision for the Future*, 2007.

[10] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proceedings British Machine Vision Conference (BMVC)*, volume 1, pages 47–56, 2006.

[11] M. Grabner, H. Grabner, and H. Bischof. Real-time tracking with on-line feature selection. In *Video Proceedings in conjunction with IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[12] H. Grabner, P.M. Roth, M. Grabner, and H. Bischof. Autonomous learning a robust background model for change detection. In *Proceedings IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 39–46, 2006.

[13] M. Grabner, H. Grabner, and H. Bischof. Fast approximated SIFT. In *Proceedings Asian Conference on Computer Vision (ACCV)*, pages 918–927, 2006.

[14] M. Kpesi, M. Neffe, T. Van Pham, M. Grabner, H. Grabner, and A. Juffinger. Audio-visual feature extraction for semi-automatic annotation of meetings. In *Proceedings IEEE International Workshop on Multimedia Signal Processing*, 2006.

[15] P.M. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. On-line conservative learning for person detection. In *Proceeding IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[16] P.M. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. Conservative visual learning for object detection with minimal hand labeling effort. In *Proceedings German Association for Pattern Recognition (DAGM)*, pages 293–300, 2005.

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

Books/Editor

[1] M. Grabner and H. Grabner. editors. In *Proceedings Computer Vision Winter Workshop*, 2007.

Technical Reports

[1] M. Grabner, H. Grabner, and H. Bischof. Fast visual object identification and categorization. NIPS Workshop in Interclass Transfer, 2005.

Others

[1] H. Grabner and C. Beleznai. History of computer vision. *OCG Journal*, 3:28–29, 2008.

[2] H. Schwabach, M. Harrer, A. Waltl, H. Bischof, A. Tacke, G. Zoffmann, C. Beleznai, B. Strobl, H. Grabner, and G. Fernndez. Vitus: Video based image analysis for tunnel safety. In *International Conference on Tunnel Safety and Ventilation*, 2006.

[3] H. Grabner. Autodetektion mit AdaBoost. Master's thesis, Graz University of Technology, 2004.

Bibliography

[1] Y. Abramson and Y. Freund. SEmi-automatic VIsual LEarning (SEVILLE): Tutorial on active learning for visual object recognition. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[2] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(11):1475–1490, 2004.

[3] M. Anthony and N. Biggs. *Computational Learning Theory: An Introduction.* Cambridge University Press, 1992.

[4] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26:1064–1072, 2004.

[5] S. Avidan. Ensemble tracking. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 494–501, 2005.

[6] M-F. Balcan, A.Blum, and K. Yang. Co-training and expansion: Towards bridging theory and practice. In *Neural Information Processing Systems.* MIT Press, 2004.

[7] A. L. C. Barczack, M. J. Johnson, and C. H. Messom. Real-time computation of Haar-like features at generic angles for detection algorithms. *Research Letters in the Information and Mathematical Scienece*, 9:98–111, 2006.

[8] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[9] Avrim Blum. On-line algorithms in machine learning. pages 306–325, 1996.

[10] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*, pages 92–100, 1998.

[11] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 236–243, 2005.

[12] K. Bowyer, P. Flynn, and R. Kasturi. The 20th anniversary of the ieee transactions on pattern analysis and machine intelligence. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(1):1–3, 2000.

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

[14] V. Carvalho and W. Cohen. Single-pass online learning: performance, voting schemes and online feature selection. In *Proceedings ACM SIGKDD International Conference on Knowledge Ciscovery and Cata Mining*, pages 548–553, 2006.

[15] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, 2005.

[16] A. Demiriz, K.R. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.

[17] J. DiCarlo and D. Cox. Untangling invariant object recognition. *Trends in Cognitive Science*, 11:333–341, 2007.

[18] T. G. Dietterich. Ensemble methods in machine learning. In *Proceedings International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.

[19] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, 2 edition, 2001.

[20] S. Edelman, N. Intrator, and T. Poggio. Complex cells and object recognition. *Unpublished manuscript*, 1997.

[21] L. Fei-Fei, A. Iyer, C. Koch, and P. Perona. What do we perceive in a glance of a real-world scene? *International Journal of Computer Vision (IJCV)*, 7(10):1–29, 2007.

[22] L. Fei-Fei, R.Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(4):594–611, 2006.

[23] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multi-camera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(2):267–282, 2008.

[24] Y. Freund and R. 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.

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

[26] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[27] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Proceedings International Conference on Computer Vision (ICCV)*, 2007.

[28] C. Giraud-Carrier. A note on the utility of incremental learning. *AI Communications*, 13(4):215–223, 2000.

[29] H. Grabner and C. Beleznai. History of computer vision. *OCG Journal*, 3:28–29, 2008.

[30] H. Grabner and H. Bischof. On-line boosting and vision. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 260–267, 2006.

[31] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proceedings British Machine Vision Conference (BMVC)*, volume 1, pages 47–56, 2006.

[32] H. Grabner, C. Leistner, and H. Bischof. Time dependent on-line boosting for robust backgroundmodeling. In *Proceedings International Conference on Computer Vision Theory and Applications*, 2007.

[33] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *Proceedings European Conference on Computer Vision (ECCV)*, 2008.

[34] H. Grabner, T.T. Nguyen, B. Gruber, and H. Bischof. On-line boosting-based car detection from arial images. *ISPRS Journal of Photogrammetry & Remote Sencing*, 63(3):382–396, 2007.

[35] H. Grabner, P.M. Roth, and H. Bischof. Eigenboosting: Combining discriminative and generative information. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[36] H. Grabner, P.M. Roth, and H. Bischof. Is pedestrian detection realy a hard task? In *Proceedings IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2007.

[37] H. Grabner, P.M. Roth, M. Grabner, and H. Bischof. Autonomous learning a robust background model for change detection. In *Proceedings IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 39–46, 2006.

[38] H. Grabner, J. Sochman, H. Bischof, and J. Matas. Training sequential on-line boosting class for visual tracking. In *Proceedings International Conference on Pattern Recognition (ICPR)*, 2008.

[39] M. Grabner. *Visual Tracking through Online Learning of Discriminative Representations*. PhD thesis, Graz University of Technology, Faculty of Computer Science, 2008.

[40] M. Grabner, H. Grabner, and H. Bischof. Fast approximated SIFT. In *Proceedings Asian Conference on Computer Vision (ACCV)*, pages 918–927, 2006.

[41] M. Grabner, H. Grabner, and H. Bischof. Learning features for tracking. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[42] M. Grabner, H. Grabner, J. Pehserl, P. Korica-Pehserl, and H. Bischof. Flea, do you remember me? In *Proceedings Asian Conference on Computer Vision (ACCV)*, pages 657–666, 2007.

[43] M. Grabner, C. Zach, and H. Bischof. Effcient tracking by linear programming of weak binary classifier ensembles. In *Proceedings German Association for Pattern Recognition (DAGM)*, 2008.

[44] S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Neural networks and natural intelligence*, pages 213–250, 1998.

[45] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. 3:1157–1182, 2003.

[46] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(10):1025–1039, 1998.

[47] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[48] M. Heikkilä, M. Pietikäinen, and J. Heikkilä. A texture-based method for detecting moving objects. In *Proceedings British Machine Vision Conference (BMVC)*, pages 187–196, 2004.

[49] T. Hertz, A. Bar-Hillel, and D. Weinshall. Boosting margin based distance functions for clustering. In *International Conference on Machine Learning*, page 50, 2004.

[50] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2137–2144, June 2006.

[51] D. Hoiem, A.A. Efros, and M. Hebert. Geometric context from a single image. In *Proceedings International Conference on Computer Vision (ICCV)*, 2005.

[52] Ch. Huang, H. Ai, S. Lao, and Y. Li. High-performance rotation invariant multiview face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(4):671–686, 2007.

[53] J. Huang and D. Mumford. Statistics of natural images and models. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 1999.

[54] A. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[55] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 695–700, 2005.

[56] A. Karmaker and St. Kwek. A boosting approach to remove class label noise. In *HIS*, pages 206–211, 2005.

[57] T. Kenner. Fehlererkennung mittels one-class boosting. Master's thesis, Graz, University of Technology, 2007.

[58] S. Khan and M. Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *Proceedings European Conference on Computer Vision (ECCV)*, pages 133–146, 2006.

[59] J. Kittler, M. Hatef, Duin R.P.W, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20:226–239, 1998.

[60] St. Kluckner, G. Pacher, H. Grabner, H. Bischof, and J. Bauer. A 3d teacher for car detection in aerial images. In *Proceedings ICCV Workshop on 3D Representation for Recognition*, 2007.

[61] C. Koch and S. Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology*, 4:219–227, 1985.

[62] E. Kreyszig. *Advanced Engineering Mathematics*. John Wiley (New York), 8th edition, 1999.

[63] L.I. Kuncheva. Classifier ensembles for changing environments. In *Proceedings Workshop on Multiple Classifier Systems*, pages 1–15, 2004.

[64] F. Leberl, S. Kluckner, G. Pacher, H. Grabner, H. Bischof, and M. Gruber. Recognizing cars in arial imagery to improve orthophotos. In *Proceedings American Society for Photogrammetry and Remote Sensing*, 2008.

[65] B. Leibe, A. Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision (IJCV)*, 2007.

[66] C. Leistner, P. Roth, H. Grabner, H. Bischof, A. Stratzer, and B. Rinner. Visual on-line learning in distributed camera networks. In *Proceedings International Conference on Distributed Smart Cameras*, 2008.

[67] A. Leonardis and H. Bischof. Robust recognition using eigenimages. *Computer Vision and Image Understanding (CVIU)*, 78:99–118, 2000.

[68] K. Levi and Y. Weiss. Learning object detection from a small number of examples: The importance of good features. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 53–60, 2004.

[69] A. Levin, P. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. In *Proceedings International Conference on Computer Vision (ICCV)*, volume 2, pages 626–633, 2003.

[70] L-J. Li, G. Wang, and L. Fei-Fei. Optimol: automatic object picture collection via incremental model learning. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[71] S.Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *ECCV*, page IV: 67 ff., 2002.

[72] Y. Li. On incremental and robust subspace learning. *Pattern Recognition*, 37:1509 – 1518, 2004.

[73] Y. Li and W. Ito. Shape parameter optimization for adaboosted active shape model. In *Proceedings International Conference on Computer Vision (ICCV)*, 2005.

[74] R. Lienhart and J. Maydt. An extended set of haar-like features for object detection. In *Proceedings International Conference on Image Processing*, pages 900–903, 2002.

[75] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[76] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

[77] X. Liu and T. Yu. Gradient feature selection for online boosting. In *Proceedings International Conference on Computer Vision (ICCV)*, 2007.

[78] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[79] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157, 1999.

[80] D. Marr. *Vision: a computational investigation into the human representation and processing of visual information.* W. H. Freeman, San Francisco, 1982.

[81] N. J. B. McFarlane and C. P. Schofield. Segmentation and tracking of piglets. *Machine Vision and Applications*, 8(3):187–193, 1995.

[82] T.M. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[83] V. Nair and J.J. Clark. An unsupervised, online learning framework for moving object detection. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[84] T.T. Nguyen, H. Grabner, B. Gruber, and H. Bischof. On-line boosting for car detection from aerial images. In *Proceedings IEEE International Conference on Research, Innovation and Vision for the Future*, 2007.

[85] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(7):971–987, 2002.

[86] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Weak hypotheses and boosting for generic object detection and recognition. In *Proceedings European Conference on Computer Vision (ECCV)*, volume 2, pages 71–84, 2004.

[87] J. O'Sullivan, J. Langford, R. Caruana, and A. Blum. Featureboost: A meta-learning algorithm that improves model robustness. In *Proceedings International Conference on Machine Learning*, pages 703 – 710, 2000.

[88] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 130–136, 1997.

[89] N. Oza. *Online Ensemble Learning*. PhD thesis, University of California, Berkeley, 2001.

[90] N. Oza. Aveboost2: Boosting for noisy data. In *Multiple Classifier Systems*, pages 31–40, 2004.

[91] N. Oza and S. Russell. Experimental comparisons of online and batch versions of bagging and boosting. In *Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

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

[93] G. Pacher, S. Kluckner, and H. Bischof. An improved car detection using street layer extraction. In Janez Pers, editor, *Proceedings Computer Vision Winter Workshop*, pages 1–8, 2008.

[94] T. Parag, F. Porikli, and A. Elgammal. Boosting adaptive linear weak classifiers for online learning and tracking. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[95] J.-H. Park and Y. Choi. On-line learning for active pattern recognition. *IEEE Signal Processing Letters*, 3(11):301–303, 1996.

[96] R. Pflugfelder and H. Bischof. Online auto-calibration in man-made worlds. In *Digital Image Computing: Technqiues and Applications*, pages 519 – 526, 2005.

[97] M-T. Pham and T-J Cham. Online asymetric boosted clasifiers for object detection. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[98] N. Pinto, D. Cox, and J. DiCarlo. Why is real-world visual object recognition hard? *PLoS Computational Biology*, 4(1), 2008.

[99] A. Pinz. Object categorization. In *Foundations and Trends in Computer Graphics and Vision*, volume 1. 2006.

[100] F. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 829–836, 2005.

[101] G. Rätsch, T. Onoda, and K.R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

[102] G. Rätsch, B. Schökopf, S. Mika, and K.R. Müller. Svm and boosting: One class. Technical Report 119, GMD FIRST, Berlin, November 2000.

[103] D.B. Redpath and K. Lebart. Observations on boosting feature selection. In *Proceedings Multiple Classifier Systems*, pages 32–41, 2005.

[104] A. Rosenfeld. From image analysis to computer vision: Motives, methods, and milestones, 1955-1979. Technical report, Center of Automation Research, University of Maryland, 1998.

[105] P. Roth. *On-line Conservative Learning*. PhD thesis, Graz University of Technology, Faculty of Computer Science, 2008.

[106] P. Roth, H. Grabner, C. Leistner, M. Winter, and H. Bischof. Interatctive learning a person detector: Fewer clicks - less frustration. In *Proceedings Workshop of the Austrian Association for Pattern Recognition (AAPR)*, 2008.

[107] P.M. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. Conservative visual learning for object detection with minimal hand labeling effort. In *Proceedings German Association for Pattern Recognition (DAGM)*, pages 293–300, 2005.

[108] P.M. Roth, H. Grabner, D. Skočaj, H. Bischof, and A. Leonardis. On-line conservative learning for person detection. In *Proceeding IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[109] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 203–208, 1996.

[110] C. Rudin, I. Daubechies, and R.E. Schapire. The dynamics of adaboost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research*, 5:1557–1595, 2004.

[111] R. Schapire. The boosting approach to machine learning: An overview. In *Proceedings MSRI Workshop on Nonlinear Estimation and Classification*, 2001.

[112] R. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings International Conference on Machine Learning*, pages 322–330, 1997.

[113] R. Schapire, M.Rochery, M. Rahim, and N. Gupta. Incorporating prior knowledge into boosting. In *Proceedings International Conference on Machine Learning*, 2002.

[114] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rate predictions. *Machine Learning*, 37(3):297–336, 1999.

[115] K. Schindler and L. van Gool. Action snippets: How many frames does human action recognition require? In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[116] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 5(19):530–535, 1997.

[117] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationsships for object recognition. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1998.

[118] P. Sinha, B. Balas, Y. Ostrovsky, and R. Russell. Face recognition by humans: 19 results all computer vision researchers should know about. 94(11):1948–1962, 2006.

[119] D. Skocaj and A. Leonardis. Weighted incremental subspace learning. In *Proceedings Workshop on Cognitive Vision*, 2002.

[120] J. Šochman and J. Matas. Waldboost - learning for time constrained sequential detection. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 150–157, 2005.

[121] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number 2, pages 246–252, 1999.

[122] N. A. Syed, H. Liu, and K. K. Sung. Handling concept drifts in incremental learning with support vector machines. In *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 317 – 321, 1999.

[123] K. Tieu and P. Viola. Boosting image retrieval. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 228–235, 2000.

[124] A. Torralba. Contextual priming for object detection. *International Journal of Computer Vision (IJCV)*, 53(2):169–191, 2003.

[125] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 762–769, 2005.

[126] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Proceedings International Conference on Computer Vision (ICCV)*, pages 255–261, 1999.

[127] A. Tsymba. The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, Department of Computer Science, Trinity College Dublin, Irland, 2004.

[128] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–591, 1991.

[129] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[130] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume I, pages 511–518, 2001.

[131] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Neural Information Processing Systems*. MIT, 2002.

[132] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision (IJCV)*, 2002.

[133] P. Viola, M.J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Proceedings International Conference on Computer Vision (ICCV)*, volume 2, pages 734–741, 2003.

[134] A. Wald. *Sequential analysis*. Dover, New York, 1947.

[135] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, UNC-CH Computer Science Technical Report 95041, 1995.

[136] O. Williams, A. Blake, and R. Cipolla. Sparse bayesian learning for efficient visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27:1292–1304, 2005.

[137] T. Woodley, B. Stenger, and R. Cipolla. Tracking using online feature selection and a local generative model. In *Proceedings British Machine Vision Conference (BMVC)*, 2007.

[138] B. Wu, H. Ai, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real adaboost. In *Proceedings International Conference on Automatic Face and Gesture Recognition*, pages 79–84, 2004.

[139] B. Wu and R. Nevatia. Improving part based object detection by unsupervised, online boosting. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[140] J. Wu, J.M. Rehg, and M.D. Mullin. Learning a rare event detection cascade by direct feature selection. In *Neural Information Processing Systems*. MIT Press, 2003.

[141] L. Xj, T. Yamashita, S. Lao, M. Kawade, and F. Q. Online real boosting for object tracking under severel appearance changes and occlusion. In *Int. Conf. on Acoustics, Speech and Signal Processing*, volume 1, pages 925–928, 2007.

[142] P. Yang, S. Shan, W. Gao, S.Z Li, and D.Zhang. Face recognition using Adaboosted Gabor features. In *Proceedings Conference on Automatic Face and Gesture Recognition*, pages 356–361, 2004.

[143] H. Zhang, W. Gao, Y. Chen, and D. Zhao. Object detection using spatial histogram features. *Image and Vision Computing*, 24:327–341, 2006.

[144] H. Zhang, W. Jia, X. He, and Q. Wu. Learning-based license plate detection using global and local features. In *Proceedings International Conference on Pattern Recognition (ICPR)*, 2006.

[145] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.