FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Recognition based Tracking

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computational Intelligence

eingereicht von

Werner René Kloihofer
Matrikelnummer 0425065

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Privatdoz. Dipl.-Ing. Dr.techn. Martin Kampel

Wien, 17. Juni 2010

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Erklärung

Werner René Kloihofer
Sturgasse 7/4/3
A-1020 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 17. Juni 2010

Abstract

In this thesis, known techniques from the research area of object recognition are analyzed according to their ability to track objects in video sequences. The topic of this thesis, namely object tracking, belongs to the field of **image sequence analysis**, but the recognition based approach used comes from the fields of **image understanding**, as well as **machine learning**. This approach is chosen to overcome the problems of traditional object tracking algorithms, relying on temporal coherence, which are infeasible in some situations – for instance in cases of video material with low frame rate. Objects move too far from one frame to the next, therefore it is hard to predict their movement. Another example is a nonstationary camera, where one cannot use a background model and has to deal with object motion as well as camera motion. Most object recognition methods rely on supervised training from examples, but the nature of the tracking task suggests utilizing online learning methods.

The main contribution of this thesis is the IPTracker, a novel interest point based object tracking method. In the first step, interest points are detected and feature descriptors around them are calculated. Sets of known points are created, allowing tracking based on point matching. The set representation is updated online in every tracking step. The method uses one-shot learning with the first frame. Thus, neither offline nor supervised learning is required. An object recognition based approach does not need a background model or motion model, and therefore allows tracking of abrupt motion as well as with nonstationary cameras.

In the evaluation, we experiment with the IPTracker as well as with an own implementation of an online boosting based tracker. The methods are compared to the state of the art mean shift method using a simple tracking rate metric and widely used evaluation datasets. The influence of image quality, object size and partial occlusions is analyzed, showing the benefits of the IPTracker. Long-term stability is evaluated and the speed, which is very important for real-time tracking, receives particular attention.

Kurzfassung

In dieser Diplomarbeit wird die Anwendung von Objekterkennungstechniken (Object Recognition) zur Objektverfolgung (Object Tracking) untersucht. Mit diesem Ansatz werden klassische Einschränkungen von Object Tracking Methoden, die im Zusammenhang mit der Verwendung eines Hintergrund- oder Bewegungsmodelles auftreten, implizit gelöst. Diese betreffen etwa Videos mit niedriger Bildrate, in denen sich Objekte von einem Bild zum Nächsten weit bewegen, oder sie treten im Zusammenhang mit plötzlichen Bewegungsänderungen auf, was ein Bewegungsmodell an seine Grenzen bringt. Ein Hintergrundmodell kann nur gemeinsam mit einer stationären Kamera funktionieren, was die Anwendung mit Schwenk-Neige-Zoom (SNZ) Kameras ausschließt.

Während Object Tracking eine klassische Aufgabe in der Analyse von Bildfolgen ist, handelt es sich bei Object Recognition um einen Teilbereich aus den Forschungsgebieten Bildverstehen und Machine Learning. Objekterkennungsmethoden beruhen häufig auf Supervised Lernmethoden, bei denen aufgrund eines vorgegebenen Datensatzes offline ein Modell gelernt wird. Object Tracking legt allerdings eine online Methodik nahe.

Zur Lösung dieser Problematik wurde ein neuartiges Tracking-Verfahren entwickelt, welches auf dem Matching von Interest Points beruht. Diese können etwa durch die bekannten Verfahren SIFT oder SURF extrahiert werden. Objekte werden als Menge von Interest Points repräsentiert und online in jedem Schritt angepasst. Es ist daher keine vorhergehende Lernphase erforderlich. Dieses Verfahren, sowie eine eigene Implementierung von Online Boosting, wird mit dem MeanShift Tracker verglichen. Dabei werden die Eigenschaften der Methoden evaluiert, insbesondere mit beweglichen Kameras, Sequenzen mit niedriger Bildrate sowie Verdeckungen.

Danksagung

Allen voran möchte ich meinem Betreuer Martin Kampel danken, insbesondere für seine Geduld, die er während des Entstehens dieser Arbeit mit mir hatte. Ich bedanke mich bei meinen Kollegen bei Center Systems, die mich während meiner ganzen Studienzeit begleitet haben, besonders bei Martin Forster, der diese Arbeit in dieser Form erst möglich gemacht hat. Außerdem möchte ich mich bei meinem Studienkollegen Oliver Hörbinger für konstruktive Diskussionen sowie eine geschaffte letzte Prüfung bedanken.

Weiters möchte ich mich bei meiner Freundin Katharina Jakobi bedanken, die mich nicht nur mental unterstützt und motiviert hat, sondern diese Arbeit auch sprachlich aufgewertet hat. Nicht zuletzt bedanke ich mich auch bei meiner Familie, die mir immer vollstes Vertrauen geschenkt hat.

Contents

Chapter 1

Introduction

1.1 Problem definition and objectives

A variety of methods for **object tracking** have become popular over the last years. An object is *"a thing that you can see or touch"*[1], in the field of image processing its actually a 2-dimensional projection of a 3-dimensional real-world object.

According to [FP03] *"tracking is the problem of generating an inference about the motion of an object given a sequence of images"*. Roughly speaking the goal of object tracking is to track an object over a temporal sequence of image frames to estimate its trajectory. In the case of a single object and stationary background a simple frame differencing approach [YJS06] leads to success – just assume that everything that moves is part of the object. So depending on the assumptions which can be made for the given scenario the problem is more or less challenging. There is no general technique in this field. Depending on the preconditions of the scenario, a large number of tracking methods are available [YJS06], which are summarized in chapter 2.

The goal of object recognition is to find and locate a given object within an image. Object recognition methods usually work with single images, while object tracking algorithms emphasize the temporal coherence of an image sequence.

The object recognition approach is chosen because there are situations where classical object tracking algorithms, which rely on temporal coherence, are infeasible. An example is video material with low frame rate. Objects move too far from one frame to the next, therefore it is necessary to use methods which are able to identify objects for assigning without relying on prediction only. Other examples can be crowded places with lots of

[1] Cambridge Advanced Learner's Dictionary 18. 4. 2009, actually the full quote is *"a thing that you can see or touch but that is not usually a living animal, plant or person"*, but since the image of something does not live the definition can be considered appropriate for our case.

objects, which are partly occluded due to overlap, or nonstationary cameras, for instance Pan-Tilt-Zoom (PTZ) cameras. Methods which can handle video sequences with reduced frame rate are expected to boost tracking performance for video sequences with full frame rate (25 FPS) as well.

There are approaches which integrate object recognition into tracking algorithms, like [LAY+07, WKZL08], but we tackle the problem from the other side: Directly use object recognition techniques for tracking.

So, the **goal of this thesis** is to solve the problem of object tracking for videos with nonstationary cameras. The outcome should be stable even under challenging conditions, which include partial occlusions and low frame rate videos. Common problems which occur in tracking, especially with nonstationary cameras, namely the change of appearance and the change of object scale, need to be considered. Thus, recognition based tracking methods are studied and developed. Investigating the properties, strengths and limitations of such methods, like the influence of image quality and long-term stability, is also necessary.

This chapter first gives an introduction to the problem and its theoretical background, as well as the theoretical background for the fields of object tracking and object recognition. Then preconditions and requirements resulting from the problem formulation are stated.

1.2 Motivation

Object tracking algorithms are tools for the extraction of information from temporal image sequences. Especially over the last few years cameras have become widely used in the surveillance and security field. For example there was an estimated amount of 500 000 security cameras in London 2003[2]. These cameras generate huge amounts of information which makes it impossible to do a complete manual analysis. Computers allow for an automatic or at least semi-automatic analysis for many applications to deal with the data. Some of them are, according to [YJS06]:

- **Motion based recognition**, which is, for example, the identification of humans based on detection and tracking
- **Automated surveillance**, this is scene monitoring for any suspicious activity or abnormal events; Video surveillance systems can point the attention of security staff to suspicious scenes.
- **Video indexing**, the automated annotation of video material in a database for retrieval purposes

[2]Taken from English Wikipedia article "closed-circuit television", 20. 4. 2009

- **Human-Computer interaction**, cameras with object tracking allow new user interface technologies, for example controlling a computer program simply by moving an item in your hand
- **Traffic monitoring**, which means gathering traffic information allowing to control the situation or as a surveillance application
- **Vehicle navigation**, this allows autonomous vehicles to plan their path or avoid obstacles

Even if the quality of algorithms does not allow for a fully automatic system, it still helps to draw the attention of security personnel to the detected situation enabling a single person to monitor hundreds of cameras, an otherwise infeasible task.

1.3 Object Tracking

The goal of object tracking is to estimate object trajectories from image sequences. We start by giving an introduction to object tracking and related terms and we also state why this problem can be difficult and how it can be tackled.

1.3.1 Motion Analysis and Tracking

Sonke et al. [SHB99] make use of the term motion analysis, it stands for a family of image processing problems which have a temporal image sequence as input data. They divide the field into three main groups:

- **Motion detection** deals with detecting whether there is motion in the image. This is the simplest form used for security purposes. Usually, a single static camera is used.
- **Moving object detection and location** is about detecting and locating objects in the image sequence to estimate their trajectories of motion and to predict their future trajectories. Usually, either the camera or the object is stationary. In comparison to the first group, this problem is considerably more complex.
- The third group is about **deriving 3D object properties** from a set of 2D projections acquired at different time instants of object motion.

Our problem falls into the second category, although we do not assume stationary cameras and objects. This makes the problem even harder since we have less prior knowledge.

1.3.2 Assumptions

To make the task of object tracking tractable several assumptions can be made. These assumptions are derived from available prior knowledge. In general, prior knowledge can help to decrease the complexity of problems. Of course making any assumptions, in other words relying on prior knowledge, leads to errors in case the assumptions do not hold in the given scenario.

Motion assumptions

According to [SHB99], several assumptions based on motion can be made. **Maximum velocity** assumes a maximum velocity c_{max} an object can have. As a result, you only have to search for the object in the next frame within a circle, where the radius depends on the maximum velocity. The radius depends also on the time intervals dt between two consecutive frames, so the actual radius is $c_{max}dt$. **Small acceleration** means that the change of velocity in time is bound by some constant. This could also be considered as smoothness. **Common motion** means that all points of an object move in a similar way. **Mutual correspondence** is the assumption that each point of an object corresponds to exactly one point in the next frame and vice versa.

For our application, we expect at least the last two assumptions to hold. Obviously, this is only possible to a certain degree, since for instance pose changes will cause the assumptions to fail for some points.

Appearance assumptions

It is unrealistic for real-world tracking scenarios to expect constant appearance. Appearance can change due to alterations in lighting or in perspective, the latter of which is due to object or camera movement. Evidently the appearance of an object from the front and from the side can be different. Nonrigid objects, like persons with arms and legs moving independently of the objects' movement, also have a problem of changing appearance.

Trackers which model the objects appearance need to rely on appearance assumptions and suffer from appearance changes, while for example a blob tracker relying only on motion detection does not. A tracker based on object recognition has to rely on such assumptions. The only way to handle changes in appearance is to quickly adapt the model. So we need to assume that the objects appearance does not change too fast from frame to frame.

Background clutter, which is background looking similar to the object, is also an appearance problem. Tracking an object becomes a lot easier when it moves in front of

a homogeneous white background, while tracking a person moving in the crowd is more challenging. We do not assume the absence of background clutter.

Observability assumptions

In order to track an object it must be visible in the scene. Unfortunately it can be temporarily or permanently occluded, either by background or other overlapping foreground objects. This occlusion can either be partial, which can be tackled by a object representation invariant to partial occlusions, or full occlusions, which can only be tackled by predicting the objects movement and trying to recover the object as soon as it is visible again. For our purposes we assume the object to be at least partially visible at all times.

Static foreground or background

When using stationary cameras one would assume a static background, which is not the case. Scenes recorded with a camera change over time, even when no object is present, like trees waving in the wind or changes in illumination. Javed and Shah [JS08] describe the challenges when developing a background model, which occur in real-world situations. Though we choose a recognition based approach and avoid using a background model, the same issues have to be accounted for with any tracking algorithm.

- **Gradual Illumination Changes:** This occurs particularly outdoor, caused by the sun.
- **Sudden Illumination Changes:** They completely change the color characteristics of the background. They are a problem especially for algorithms with an adaptive model, which can not adapt fast enough to the changes. The appearance of objects is also changed, so a recognition based method has to cope with them.
- **Uninteresting movement:** This is the actual movement of things which are not considered interesting objects, also known as the waving trees problem. Other examples are rain, snow or waves.
- **Camouflage:** This occurs with objects which are similar to the background. A recognition method has to distinguish between background and the object even in cases of camouflage. Specifically, a tracker should not switch to tracking the background.
- **Shadows:** Objects cast, depending on the illumination, shadows which may have a shape similar to the object itself. These shadows change with the change of

illumination, and they can also suddenly appear or disappear, for instance because of clouds occluding the sun.

- **Relocation of the background object:** Movement of an object causes changes at two positions of the image, both the old position of the object and the new one. Only the latter should be detected as foreground.

- **Initialization with moving objects:** Background models have to be able to handle initializations while moving objects are present. They occlude the actual background so accurate modeling of the background is not possible using only one frame. This problem becomes serious in scenes filled with moving objects all the time, for example highways or crowded places.

The static camera assumption will not hold for our application which rules out the utilization of a background model based technique. Still the list of challenges in this section have to be considered for our method.

1.4 Object Recognition

Object recognition is a central research topic in computer vision. Its goal is to detect and locate an arbitrary object in an image [Szu08]. The term object detection is also used with a similar meaning, though basically it is just a problem of finding out whether the object is present in the image or not. Object tracking is divided into initialization and actual tracking. In the initialization step, we want to find **every** not yet initialized **object** in the image **of an object class** – for example finding any face in the image for face tracking. For general tracking, we could also use any moving or non-background area for initialization. In the tracking phase, we want to detect the **specific initialized instance of the object class** in every frame. This differentiation becomes important when tracking multiple objects, which is not a focus of this thesis.

Knowledge is required about an object in order to detect it, which is the knowledge representation problem known from the field of artificial intelligence [SHB99]. An object could be described formally, using for instance predicate logic or other formalisms. To avoid this step we want to obtain a representation using statistical pattern recognition methods.

As figure 1.1 shows, the first step is the construction of a formal representation – the actual pattern or feature vector. In image processing, you have a discrete 2D image of the original object, basically a matrix of color values. Each cell of the matrix is called a pixel (picture element). You could directly use the pixel values as the formal representation, but

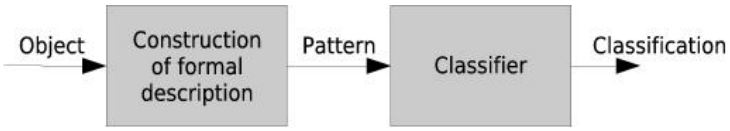


Figure 1.1: Pattern Recognition Steps, from [SHB99]

the dimensionality is hard to handle in the following step. An 100x80 pixel image would lead to an 8000-dimensional feature vector. The goal of this step is to get a representation as compact as possible while still being distinctive. Another reason for choosing a different representation is to decrease the variability within the object class while increasing the variability between different objects, which can be done by including additional "ad-hoc" knowledge [Gra08]. This first step is mainly dependent on the designer, although there are algorithms which integrate feature selection into the learning procedure, for instance [GB06]. Still, the choice of representation is up to the designer and depends on the application. Some features are well suited for one application while others are not. Even for algorithms with integrated feature selection, you need to specify a set of feature types to choose from or a whole feature family. The term feature itself is extensively used in literature. You could call the image pixels itself features, as well as values derived from them. You can also do post-processing with features resulting in new features expected to produce better results, for instance principal component analysis.

In the field of image processing, the term **local feature** is used for image patterns that differ from their immediate neighborhood [TM07]. By taking measurements centered at this local feature, a **local descriptor** can be calculated which can be used as feature vector according to our definition in this section.

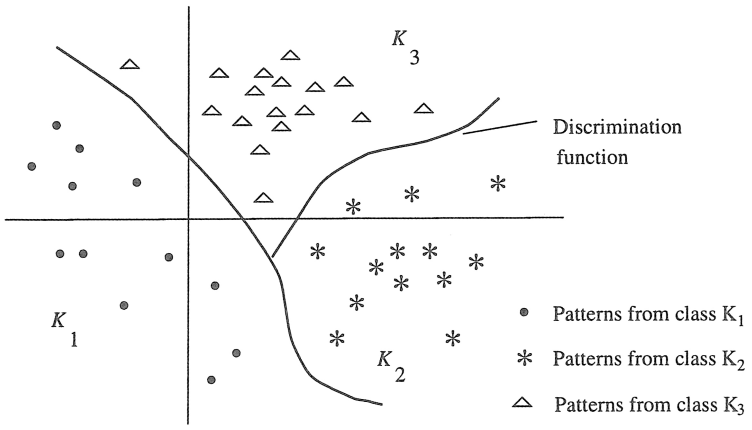


Figure 1.2: Discrimination Functions, from [SHB99]

The second step is the classification. Based on the feature vector, a decision on the object class has to be made. For object tracking, these classes could be the tracked object versus background and other objects. Since there are two classes, this is called a **binary classification problem**. The pattern or feature vector $x = (x_1, x_2, ..., x_n)$ describes an object and is a vector of elementary descriptions. All possible feature vectors form the pattern or feature space X. The descriptions should be chosen in a way that similar objects are in close proximity in feature space and all objects corresponding to the same class form a cluster. These clusters can be separated by a discrimination curve (or a hypersurface in the multi-dimensional case), see figure 1.2 for an example with a 3-class problem. When these hypersurfaces are hyperplanes, the problem is called linearly separable [Web02]. The process of automatically finding this hypersurface leads us to the field of machine learning.

1.4.1 Machine Learning

According to [Web02] *"Machine learning is the study of machines that can adapt to their environment and learn from example"*, which is also required from a recognition based tracking method. In order to use machine learning methods, one needs to understand the fundamental principles of the field.

Machine Learning and Artificial Intelligence

Machine learning is a branch of the field of artificial intelligence. We want to start by giving a grasp on the history of artificial intelligence. We focus on machine learning, especially on neural networks and classifiers, though there are other fields and applications like planning and scheduling, game playing and robotics [RN03].

According to [RN03] the first work which is now recognized as AI was McCulloch and Pitts model of a neuron which we describe later in this chapter. The first neural network computer was built in 1950 by Marvin Minsky and Dean Edmonds. Also in 1950 Alan Turing introduced the Turing test, as well as machine learning, genetic algorithms and reinforcement learning. The first successes, based on simple examples, in this field led to great expectations, which were rarely fulfilled. In 1957 Herbert Simon stated, that

> *"It is not my aim to surprise or shock you—but in the simplest way I can summarize is to say that there are now in the world of machines that think, that learn and that create. Moreover, their ability to do these things is going to increase rapidly until—in a visible future—the range of problems they can handle will be coextensive with the range to which the human mind has been applied."*

He made a more precise prediction on visible future, he claimed that in ten years a computer would be chess champion and that a significant mathematical theorem would be proved by machine. These predictions came true, or at least approximately true, just that it took 40 years. Research on automated natural language translations also didn't lead to the first expected success and is even nowadays an imperfect, but widely used, tool. Minsky later on (1969) proved the limitations of Perceptrons, which is that they can only represent very limited functions, namely only linearly separable problems, so for instance no logical "exclusive or". Though the back-propagation learning rule, which permits building more complex networks and thus solving the problem, was discovered in 1969, Minsky's result led to a cut in the funding of neural networks research. This lasted until neural networks were rediscovered in the 1980s, where at least four different groups reinvented the back-propagation learning rule [RN03]. Since the 1990s AI has advanced more rapidly, with subfields including vision and robotics, because of the greater use of scientific methods like experimenting and comparing approaches [RN03].

Types of Learning

In [RN03], learning methods are divided into three categories: Supervised, Unsupervised and Reinforcement learning. **Supervised learning** is about learning a function from sample inputs and the expected outputs, also called labels. Examples for supervised learning methods are Perceptrons or Support Vector Machines. It is also known as learning through a teacher. **Unsupervised learning** is about learning patterns from input data without providing an output. One group of unsupervised methods is also known as cluster analysis and can be done for instance by k-Means. **Reinforcement learning** is learning without a teacher, but with reinforcement. This means that a reward and not the label itself is provided as feedback [Gra08]. There is another form called **Semi-supervised learning** which is supervised learning where additional data without a label is given.

For tracking, all kinds of learning techniques can be useful. Supervised learning for training to recognize specific object classes and unsupervised learning could be used to cluster tracking points to deal just with a subset of points instead of the whole set. Reinforcement learning is useful when taking the tracking success of points into account, giving positive feedback when the points are plausible and negative feedback otherwise. Semi-supervised learning, as suggested in [GLB08], can be used to combine supervised learning with additional unlabeled data collected during tracking.

Generalization

One important property of machine learning methods is their generalization performance. *"The goal of [..] training is not to learn an exact representation of the training data itself, but rather to build a statistical model of the process which generates the data."* [Bis95] This means that the goal is not to just precisely model the input data, but also to find a model which makes correct predictions on new data. Input data has to be considered as **noisy**, especially in the field of image processing where even physical noise is introduced by the camera sensor. The goal of **model selection** is to choose a model which best fits the process generating the input data.

The main question here is how complex this model should be[3]. For example, for the problem of fitting curves by polynomials given noisy sample points, you could use a simple 1-dimensional polynomial, thus fitting a line as shown in figure 1.3, or a n-dimension polynomial. The complexity of the model can be measured for instance in the number of free parameters. If the complexity of the model is too small, this leads to a high error on the known input data as well as for generalization. On the other hand, the risk of a complex model is to model the noise of the input data and therefore get a bad generalization performance.

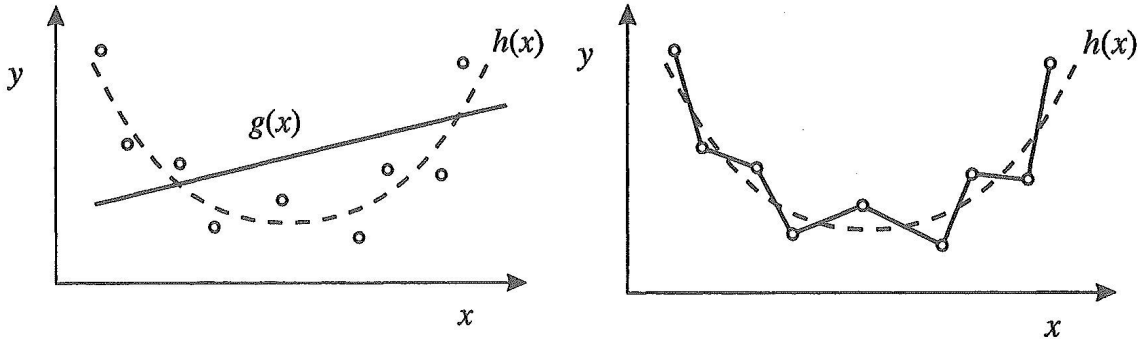


Figure 1.3: Modeling the original function h(x), represented by noisy sample points, by a linear function (left, high bias, zero variance) and by interpolation (right, low bias, high variance). Taken from [Bis95].

The error of a classifier, which is the deviation between the predicted and correct results, can be decomposed into two categories, a bias and a variance term, which is defined as the squared bias plus the variance[4]. If the classifier function is on average different from the original function it is called bias. On the other hand if the classifier function is sensitive to a particular dataset, meaning it is larger for some datasets and

[3] see page 409 of [Web02]

[4] see page 333 of [Bis95] for the derivation

lower for others, it is called variance. A model which is simpler than the original function will have a large bias while a model which is more complex will have a large variance. There is a natural trade-off between those values [Bis95] and you can always improve one by decreasing the other. Still, the goal is to obtain a minimal generalization error which is reached by optimally modeling the process which generates the data. Since this process is usually unknown, otherwise there would be no need to apply learning procedures, one needs to find other ways to reach the goal.

There are various ways to minimize bias and variance with a supervised learning method. The aim is to avoid overfitting the data while still allowing a sufficiently complex model which is capable of modeling the original function. Possible approaches are by using regularization or by stopping the training procedure early. Learning methods usually decrease the error gradually with the number of iterations, with the biggest improvement in the first steps. The idea of early stopping is to stop before the model starts fitting the noise.

Given a classification function, the **margin** is defined as the distance between the function and the closest data point. It can be shown that the margin has an influence on the generalization capabilities of the classifier [RN03], so maximization of the margin is a desirable property. In figure 1.4, the margin is shown which is the area between the chain dotted lines and the discrimination function. The sample points defining the margin are marked with circles.

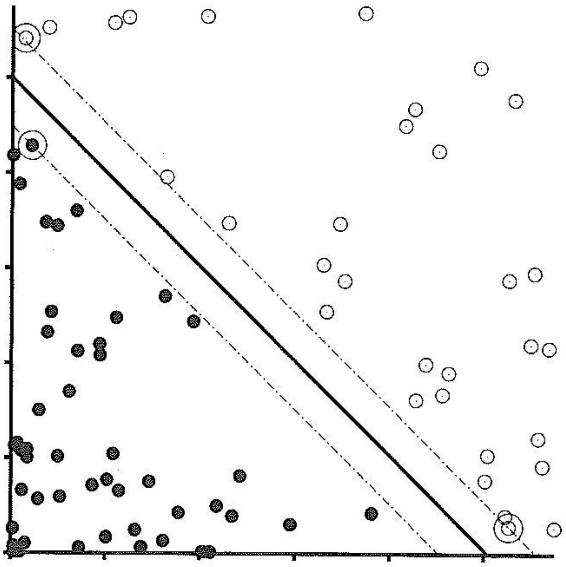


Figure 1.4: Two sets of data separated by a discrimination function, showing the margin, which is the area separating the two sets (chain dotted lines) and support vectors (points on the margin). Taken from [RN03].

Model Selection

We have introduced what is required from a model, namely a minimal generalization error. Usually, a set of sample points in the input data space is given for training. By just using the whole input data set as a training set, you have no way to determine the generalization error, or how the classifier will perform on samples not in the training set. The best results on the whole input data set will be produced by a function which just maps input data to the expected output by interpolation, as in figure 1.3 on the right, which would result in a high variance. So the model having the best performance on training data set will usually not give the smallest error. Since the original process is assumed to be unknown one needs to find another optimization criterion. There are various ways of tackling this problem, following [Web02], four of them are separating training and test set, cross-validation, using a Bayesian approach and using Akaike's information criterion[5]. We want to explain the first two in detail.

The simplest way is to separate the input data set into two sets, one training and one test set. The model is trained with the training set and afterwards the results are verified using the test set. The test set can be used to select a model and optimal parameters, the goal of the training procedure is not to minimize the error on the training set alone but also on the test set. In this case, these two sets are not independent anymore and one needs a third set, called validation set, to estimate the error.

Since the number of input data samples is in practice limited, it is not always possible to keep aside a test set [Bis95]. A solution is provided by cross-validation. With this method the training set is divided into S parts, where $S-1$ are used as a training set and the remaining set is used to estimate the error. This procedure is done S times, which is the disadvantage of the method since it increases the time needed for training S times. A typical choice for S is 10 [Bis95], which is known as 10-fold cross-validation.

Online and Offline Learning

For offline learning all training samples $\mathbf{x} \in \mathcal{X}$ and labels $y \in \mathcal{Y}$ must be given in advance. The actual use of the method happens strictly after the training phase. For online learning methods, the inputs become available over time, usually one at a time. Learning might need to go on indefinitely, as long as it is needed by the task to solve, so there are no separate training and usage phases. According to [Gra08], an algorithm is online if it produces for any training sample $(\mathbf{x}_t, y_t)$, a hypothesis h_t, which only depends on $h_t - 1$ and the current sample $(\mathbf{x}_t, y_t)$. This means that h_t is always the best approximation so

[5]for details on this methods see page 410 of [Web02]

far, and the method can always produce answers on queries, the quality just increases over time. Online learning methods therefore have a classify method $h_t : \mathcal{X} -> \mathcal{Y}$ and an update step $h_t = update(h_{t-1}, (\mathbf{x}_t, y_t))$. This means that every sample is only presented once to the algorithm, otherwise one would have to keep all previous samples in memory, and the time and space consumption would increase over time. This is contrary to offline methods where multiple training iterations over the whole input set are performed.

Since an offline algorithm has the possibility to analyze all samples at once, it potentially performs better than an online variant. An online algorithm is lossless if it returns exactly the same hypothesis given the same input data compared to its offline variant. Still, online algorithms have advantages and are even required for specific applications like tracking because of the following properties. They are able to cope with **large training data**, which would not fit into memory at once. Offline methods tend to consume a long time for training, caused by presenting every input multiple times to the method, which allows better results than online learning. For online methods this increase of quality might even be unnecessary, since instead of presenting the same samples multiple times new ones are sampled in every step of online learning. Another property is the **availability** of training data. Not all the data are available right from the beginning, so the data generation process might change over time. In which case, the goal of online learning is also to adapt to these changes, or in other words to forget irrelevant information and concentrate on the current situation.

So online learning has a different kind of application: For offline learning, a model is trained on prelabeled samples, requiring a training phase and those samples, only once, which is then used for its purpose. With online learning, a small prelabeled set with a short training period is sufficient. Further learning and specialization happens live while the method is applied. Providing labels for the online learned samples is of course a problem, which can be solved by using reinforcement learning or semi-supervised methods.

To compare and validate online learning algorithms, the same steps for offline methods, for example cross-validation, can and should be used. These steps have to be taken offline with a limited set of input data. Another option is to compare it based on the application, for example by its tracking results in cases of object tracking.

1.5 Preconditions and Requirements

Automated surveillance systems can be classified into categories characterized by:

- The **environment** they are designed for, with the main categories outdoor and indoor, but also for instance airborne or in a tunnel [JS08].

- The **number of sensors** involved, especially single or multi-sensor. A system with multiple sensors working together only on the highest layer, i. e. the presentation of results in the user interface, can be considered a single-sensor system from the algorithmical perspective, in contrast to a system handling object tracking across cameras [JS08].
- The **modality** of the sensors, for example there are video, audio and vibration sensors and also sensor combinations.
- The **mobility of the sensor**, stationary vs. mobile cameras and also combinations like Pan-Tilt-Zoom (PTZ) cameras [JS08].
- The **expected events** in the scenario, like detecting and tracking incidents with pedestrians in a station concourse. You would not expect cars in such a case, therefore there is no need to use an algorithm for that.

These basic characteristics have a strong influence on the design of a surveillance system and its algorithms. Since this thesis only deals with the lower level tracking algorithm, we only cover a single video sensor. We do not expect the sensor to be stationary nor do we presume any sensor movement constraints. Tracking is only the first step to an event detection, but as far as this thesis goes we handle only fixed scenarios like tracking one object in each sequence only. We do not want to create a specialized one-purpose (like pedestrians, cars, faces ...) tracker, the same tracker with the same parameterization and prior knowledge should be able to track different object classes. We also want to avoid being dependent on a manual training procedure for every object class.

Additional Input Parameters

We do not need additional input parameters like camera calibration, additional information about camera location or the cameras field of view. We also use only one single camera for our methods.

Low Frame Rate

Our methods should be able to cope with low frame rate (LFR) sequences. This results in both abrupt changes of appearance and fast motion. We consider 5 frames per second (FPS) as a low frame rate, which also corresponds to the definition used in [LAY+07]. Frame rate is not the only number to be considered. The speed of the object itself has a similar influence, so the faster an object is moving the more frames per second are required

for successful tracking. Fast and abrupt object movement both lead to an increased search space for finding the object in the next frame.

Realtime

Most of the applications mentioned in section 1.2 require real-time processing of data. In the field of object tracking, this means usually 25 frames per second, which is the frame rate a PAL camera delivers after deinterlacing. There is no sharp definition of what kind of properties an algorithm is required to have to be called real-time capable. Obviously faster computers allow higher frame rates. If an algorithm does not require all frames to reach its desired quality it could still be called real-time, even when processing just one frame per second. Henceforth, we define real-time as being able to process 25 frames per second on current hardware, which is an Intel Core2 Quad with 4 x 2.6 GHz – though only one processor core is utilized unless otherwise noted.

1.6 Thesis Structure

This thesis is structured as follows: Chapter 2 gives an overview of the literature in the field as well as brief summarizations of the methods available. Chapter 3 explains the methods used in detail as well as the methodology utilized for evaluation. In chapter 4 the results of the evaluation are presented and discussed. Chapter 5 summarizes the thesis and also gives an outlook on possible future work.

Chapter 2

State of the Art

In literature, the use of object detection and recognition for tracking is mentioned together with the problem of tracking in low frame rate video (LFR) and the tracking of abrupt motion [LAY+07]. In this chapter, we first give an overview of tracking techniques in section 2.1. Then object recognition methods are described in regards to their application to object tracking in section 2.2 and at the end of this section we present approaches which are already used for object tracking in the literature. We conclude the chapter with a discussion in section 2.3.

2.1 Object Tracking

Using the taxonomy of Yilmaz et al. in their object tracking survey [YJS06], tracking methods can be divided into 3 groups: Point tracking, kernel tracking and silhouette tracking. We describe those methods following their survey.

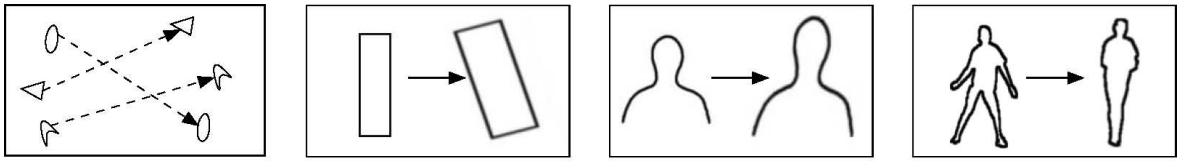


Figure 2.1: Taken from [YJS06]. (a) Multipoint correspondence in point tracking, (b) parametric transformation of a rectangular patch (kernel tracking), (c,d) Contour evolution for silhouette tracking

2.1.1 Point Tracking

In point tracking, the object is represented by points. The correspondence of points is determined by their previous state, meaning position and motion. Point tracking algorithms are divided into two groups, deterministic and statistical methods.

Deterministic Methods

For deterministic methods, costs are associated to each object in frame $t-1$ for associating it to a single object in frame t. The goal is to minimize the correspondence costs, which can be formulated as a combinatorial optimization problem. It can be solved for example by greedy search methods. The cost function relies on various constraints, basically the motion constraints from section 1.3.2.

Statistical Methods

These methods use a statistical method to model measurement noise as well as the inexactness of the motion model. Basically, a dynamic system is modeled using a statespace model. With the point tracking task, the state changes over time (frames), but the measurement of the current state itself is noisy. This statespace approach consists of a model for the change of state over time (system model) and a model for the noisy measurements (measurement model). This leads to the Bayesian approach: Based on the information available (measurements and earlier states) a posterior *probability density function* (*pdf*) is constructed. From this *pdf*, an optimal estimation can be derived, as well as a measurement of the accuracy of the estimate [YJS06].

Tracking requires an estimate with every new measurement received, therefore a recursive filtering approach is used. This avoids the necessity of storing every state which would be required for a batch processing method.

For tracking a moving object, the change of state over time is modeled as

$$X^t = f^t(X^{t-1}) + W^t$$

In this formula, X^t is the information representing the object at state $t = 1, 2....$ The function f describes the state transition and W^t is a white noise term. The measurement Z^t is described from the state with the function h^t using the formula

$$Z^t = h^t(X^t, N^t)$$

where N^t is white noise and independent of W^t. The goal of tracking is now to estimate

the state X^t using all prior measurements, in other words the *probability density function* $p(X^t|Z^{1,\dots,t})$. Theoretically, this can be optimally solved using a recursive Bayesian filter in two steps. The first prediction step uses a dynamic equation and the already computed *pdf* from $t-1$ to derive the prior *pdf* $p(X^t|Z^{1,\dots,t-1})$ for the current state. Then the correction step uses the likelihood function of the current measurement $p(Z^t|X^t)$ to compute the posterior *pdf* $p(X^t|Z^{1,\dots,t})$. Assuming linear functions f^t and h^t, as well as the initial state X^1 and the noise terms having a Gaussian distribuation, we can determine an optimal estimate with the Kalman Filter [WB95][1]. The optimal solution using the Bayes theorem is unfortunately only tractable when certain conditions hold. The Kalman filter assumes that the posterior *pdf* is Gaussian, hence parameterized by mean and covariance. It has been extensively used for tracking [YJS06]. Clearly, the technique leads to non-optimal results when the Gaussian assumption does not hold. In this case, approximations are necessary instead of the optimal, but intractable solution.

When these assumptions do not hold, Particle Filters [Kit87] still provide a solution. They are described later in this chapter. The methods as described here are only for tracking single objects as this thesis does not focus on multi-object tracking – though it is possible to extend them to multi-object tracking.

2.1.2 Kernel Tracking

With kernel tracking, the motion of the object is computed, where the object is represented by a primitive object region. The algorithms in this field differ in the object representation, the number of objects tracked and the motion estimation. Yilmaz et al. [YJS06] divide kernel tracking into template and density-based appearance models and into multiview appearance models.

Template and Density-Based Appearance Models

One method in this field is template matching. With this brute force method, a template of the object is searched in the whole image (search window). The template can be formed by features based on image intensity or colors, or by gradients which are less sensitive to illumination changes. The problems with this method are the high computational costs because of the brute force search, which can be improved by limiting the search region to the area around the old position, or by using a motion model.

An alternative to brute force is the MeanShift tracking approach by Comaniciu and Meer [CM02]. With this method color histograms of circular regions are used for object

[1]For a good introduction to Kalman filtering see also the slides of Alexander Stoytchev at http://www.ece.iastate.edu/~alexs/classes/2007_Spring_575X/slides/, last visited 5. 5. 2009

representation. The region is moved iteratively so that the histogram similarity increases. This procedure is repeated until convergence is reached, which is usually after five to six iterations [YJS06]. Parts of the object must be inside the initialization region, which is the position in the last frame, or the position predicted by the motion model. If this is not the case the iterative procedure, which is basically a local search, will probably result in a local maximum instead of the correct object position.

The KLT (Kanade-Lucas-Tomasi) Tracker [LK81] tracks rectangular regions around interest points using optical flow. Optical flow methods compute dense flow fields by computing the flow vector for each pixel. They first appeared in the work of Berthold Horn and Brian Schunck at the MIT in 1980 [HS80]. After the optical flow step, one knows the motion vector of every pixel, which can be easily extended to rectangular regions. With KLT tracking, interest points are determined and their movement is calculated with the optical flow in a patch around the point. When the new position is calculated the quality of the patch is evaluated by projecting the old patch to the new position and by calculating the sum of squares difference between the two patches. If it is too high the feature point is dropped.

Multiview Appearance Models

Since objects may appear different from different views there are methods which incorporated multiple views which are learned offline. Approaches in this field use Principal Component Analysis (PCA) [BJ98] or Support Vector Machines (SVM) [ATJ01].

2.1.3 Silhouette Tracking

Silhouette tracking methods focus on an objects silhouette rather than their internal features, which is done by describing the objects shape. Following Yilmaz et al. [YJS06], we divide the field into shape matching and contour tracking methods. While shape matching methods search for the object silhouette in the current frame, contour tracking methods evolve an initial contour to its new position in the current frame by using state space models or minimizing some energy functional.

Shape Matching

With shape matching, the object's silhouette and the associated model are searched in the current frame. This model is based on the last frame and the similarity of this hypothesized model is calculated in order to find the correct position in the current frame. Since this search is only from one frame to the next, nonrigid object motion is not handled. It has

to be handled by updating the object model in every step, in the same way changes in appearance and the viewpoint are handled. The object model is usually in the form of an edge map.

The Hausdorff distance can be used to emphasize the tracker on parts of the object which are not drastically effected by object motion. For example excluding the arms and legs of a person will lead to better results, since they change a lot from frame to frame.

A different approach in this field is to first extract silhouettes and afterwards use them for matching. The silhouette itself is mostly detected by background subtraction. The matching step compares silhouettes using a distance measure, for instance cross-correlation, Bhattacharya distance or Kullback-Leibler divergence [YJS06]. An example for such a method is [KCM04].

Another example is the work of Sato and Aggarwal [SA04] which uses trajectories based on the dominant flow within the object silhouette. To determine the dominant flow, they use Hough transform.

Contour Tracking

For contour tracking, in contrast to shape matching, an initial contour from the previous frame is evolved to its new position. This requires the object regions to overlap. One approach is to use state space models to both model contour shape and motion. The Kalman Snakes method by Terzopoulos and Szeliski [TS92] uses the dynamics of control points to define the object state. The state parameters are predicted using the Kalman filter.

Isard and Blakes Condensation method [IB98] models the object state with spline shape parameters and affine motion parameters. The measurements are image edges computed in normal direction to the contours. They use the particle filter to update the state. Initial samples are calculated during a training phase. The approach is later extended by Isard and Blake to track multiple objects by using the *exclusion* principle in the sampling step of the particle filter, which allows features to contribute more to the occluding object when the feature lies in the observation space of two objects. This is only defined for two objects occluding each other.

The second approach is to minimize the contour energy using direct minimization techniques. This is, contrary to the explicit representations before, an implicit representation, which allows topology changes like region split or merge. There are approaches, for instance [BSR00], which are based on the optical flow calculated at the object boundary. Other approaches, like [YLS04], are based on minimizing appearance statistics computed inside and outside the object region.

2.2 Applying Object Recognition

In order to give an overview of object recognition methods which can be used for tracking, we divide such approaches into two groups. The first group of algorithms determine interest points and calculate features around them. The second group is search window based. Therefore, it calculates its features at every position in the image. In either case, the search region is constrained to a region of interest. Though, when handling abrupt motion, one must adapt the search window size to cover the maximal expected motion.

But before coming to object recognition methods, we want to give an overview of the use of learning techniques in the literature, especially semi- or unsupervised online learning.

2.2.1 The Role of Training

Since the aim of this thesis is not to build a method specialized on a specific task, like tracking cars, supervised learning methods which require large amounts of training data are not adequate. Supervised methods, like Support Vector Machines [PP99] or Adaboost [PP96] require a two-phased process, at least this is the case for offline training. In the first phase the classifier has to be trained with many positive and negative samples. A classifier benefits from a larger number of samples, so Zang et al. [ZLP04] used 11 000 positive and 100 000 negative samples. Depending on the methods used, this process can be really time-consuming, for example it took hours to days to train a support vector machine for pedestrian detection. During the second phase, the classifier is used for its purpose and can only be changed by redoing the training phase. Therefore, handling changing conditions like a change of lighting or perspective, is impractical. Another problem with supervised training is that it needs huge amounts of prelabeled samples, and creating such a training set has to be done manually[2], which is an awkward task. For this reason, we want to focus on unsupervised or at least semi-supervised online methods. The concept of unsupervised learning of a detector from image sequences can be found in literature [Jav05, CHHB08, GLB08].

Omar Javed proposes an online co-training approach in his PhD thesis [Jav05]. The idea of co-training is based on Blum and Mitchells contribution [BM98]. Basically, the co-training algorithm uses two independent views trained with a small number of pre-labeled samples. Then, the results of one classifier are used to train the other one further. Only samples where the classifier is sufficiently confident are used. These concepts are also studied in Roth's PhD Thesis about On-Line Conservative Learning [Rot08]. In

[2] Alternatively, a different "reference" method may be available in some cases to do the labeling

Javed's thesis, he introduces an online variant of co-training with boosting. The method is successfully employed for classification of moving objects in real-time. The author states that the method can be trained in advance on a general scenario and can afterwards adapt itself online to the specific scene.

One of the problems with online adapting algorithms is **drifting** [LAY+07, GLB08]. Each time the tracker is updated, an error is introduced, which accumulates over time. This means that the tracker starts to track something different, especially if the background is similar to the object and the original object is lost. The two mentioned papers introduce solutions for this problem, both are basically hybrid approaches. They allow adaption but they still keep the originally learned representation. In the publication of Li et al. [LAY+07] different observers (detectors) with different lifespans (trained offline vs. fast online adaption) are used. The approach of Grabner et al. [GLB08] uses a fixed prior which is learned offline in addition to the online updated classifier.

2.2.2 Learning Methods

Learning methods are a valuable tool for developing a recognition based tracking algorithm. Unsupervised learning methods only have a set of samples as an input, while supervised learning methods also need the expected class label for all samples.

Supervised Learning Techniques

Examples for supervised classifiers are Neural Networks, Support Vector Machines (SVM), k-Nearest Neighbors (kNN) or Decision Trees. For a detailed explanation of all these methods see Webb's book titled "Statistical Pattern Recognition" [Web02]. To further unterstanding of how supervised learning techniques work, we will briefly explain one of the most simple methods, namely kNN, as well as the basics of neural networks and Support Vector Machines. Boosting, which is also an important supervised learning concept, is explained later in this chapter.

The k-Nearest Neighbor classifier is a special case, since no actual training is involved. The whole training set $X_{train} \subset \mathcal{X}$ is required for classification. Still, it is a supervised learning technique since a training set with its labels is necessary. Given a sample point $\mathbf{x}$, the k closest points in the training set X_{train} are determined and $\mathbf{x}$ is classified with the same label the majority of the k points have. This method has its drawbacks. One must store the entire training set and compare every new point with every point in the training set, which results in a computational complexity which depends on the number of training samples. A large parameter k, leads to a smoother result (large bias) while a small

parameter k, leads to a spikier result, which also means it fits the noise in the training set (large variance). There are many variants of kNN, allowing faster computation (for example by precalculating a distance matrix) or different classification functions. Andrew Webb gives a more detailed explanation in his book[3]. kNN is suitable not only for binary classification problems but is also directly applicable to multi-class problems.

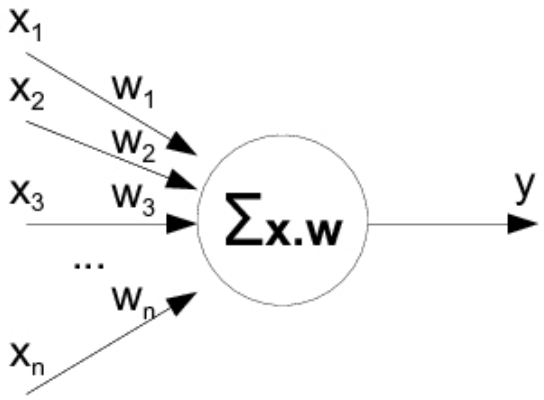


Figure 2.2: A McCulloch-Pitts neuron.

Neural networks are another important machine learning concept [SHB99]. It started with the McCulloch-Pitts neuron [MP43] in 1943. The idea is to use elementary processors (neurons), where each one has multiple weighted inputs and generates a single output. Figure 2.2 shows a general MP-Neuron. As the name suggests, this concept is modeled on the biology of the brain. The Perceptron, which was developed by Rosenblatt [Ros62], adds a learning algorithm which is guaranteed to converge to a solution, if such a solution exists. There are many variants of neural networks, like back-propagation networks, which use multiple layers of neurons, or Hopfield networks, which allow unsupervised learning and use recurrent networks, which means that the connections of neurons form a cycle.

Support vector machines (SVM), originally developed by Cortes and Vapnik [CV05], are related to McCulloch-Pitts neurons and the Perceptron, with two major extensions. The first one is a new learning algorithm. In contrast to the Perceptron learning algorithm, it maximizes the margin, which is important for generalization. The second concept, which allows use of the classifier on problems which are not linearly separable, is the use of a kernel function. The input data is transformed using the kernel function and the actual learning and classification happens in this higher dimensional space where the data is linearly separable. Kernel functions are for instance radial basis functions (Gaussians) or polynomials. The classification step is efficient since it is only dependent on the support

[3]see page 93 in [Web02]

vectors, which are items from the training set which lie on the margin, shown as the circled points in figure 1.4. The actual kernel function does not need to be computed for every point. Support vector machines became very popular in the last years [RN03] as a general machine learning technique and are used for instance for pedestrian tracking [PP99].

Unsupervised Learning Techniques

Besides the supervised learning techniques, there are also unsupervised ones which do not need a teacher, they can be used for instance when no class information is available. One group of unsupervised methods is **cluster analysis** [SHB99]. These methods separate the data into subsets, also called clusters, based on their mutual similarity. Every cluster should have a high similarity within the cluster and a low similarity to other clusters. One method in this field is called k-Means [Mac67] using Lloyd's algorithm [Llo82] which we describe here are exemplary of the concept of cluster analysis.

The k-Means method requires the number of clusters to be given in advance. There are methods to determine the optimal number for the given input data, for instance by clustering with various values for k and using the value with the result optimal to some criterion, or by using other methods which do not need this information. In the first step k samples are chosen from the set of points, either randomly or deterministically, or by using prior knowledge, their values are used as the initial cluster centers. Then, for every point, determine the closest cluster center according to a distance function. This leads to a separation of the points into disjoint subsets (clusters). After this step, the center of gravity of each cluster is determined and they are used as the new cluster centers. These steps are repeated until the centers remain fixed or until a maximum number of iterations is reached. This results in the original data separated into k clusters represented by their cluster centers. Applications for such methods are for instance lossy data compression (representing data points with minimal error) or image segmentation [KMN+02].

2.2.3 Local Descriptors around Interest Points

One way to reduce computational complexity with local descriptors is to apply them only to a subset of the image. The points in this subset are called interest points. The first step of such a method is to locate appropriate interest points. The repeatability of this procedure is an important quality criteria. To allow successful matching between different images, it is necessary to detect key points at the exact same locations. Then a feature descriptor is calculated at these points locally. The matching step compares two images with detected feature vectors around interest points. One simple option is to compare all

pairs of interest point descriptors using Euclidean distance. However other methods, for example using geometric properties, lead to faster matching (for instance sign of Laplacian [BTG06]) or additionally even better performance (best-bin-first [Low04]).

Interest Point Detectors

According to [Szu08], interest point detectors can be divided into four categories. **Blob detectors** are usually based on scale-space theory and detect blob-like structures. Methods based on scale-space determine location and scale at once. The idea is to use a kernel function modeling the blob and convolution to detect the presence of these blobs. Doing this at various scales, by either scaling the image or the kernel size [BTG06], allows one to detect the scale of the point. Examples for such methods are Difference of Gaussians used in SIFT [Low99] or detectors based on the determinant of the Hessian matrix used in [Lin98] and [BTG06]. **Corner detectors** are used to detect local curvature maxima in the gradient. Examples for such methods are the Moravec [Mor80] and Harris [HS88] corner detector. **Symmetry detectors** try to find mirror or rotational symmetry in the image intensity distribution. The forth category contains **saliency based detectors**, where the saliency of an image feature is inversely proportional to the probability of its occurrence. For a more detailed description of various interest point detectors see [Szu08] or [TM07].

Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT) [Low99] is one of the most popular image descriptors available. Several other methods [BTG06, MS05, GGB06a, KS04] are influenced by SIFT so we are still describing its basic steps, though it is for instance three times slower compared to SURF [BTG06], due to its computational complexity.

SIFT uses the Difference of Gaussians (DoG) method for finding interest points (also called key points). For this method, first the convolution of the image with a Gaussian function with $\sigma = \sqrt{2}$ is calculated. The 2D Gaussian function is separable, so the convolution in 2D can be calculated by applying the 1D Gaussian function in horizontal and vertical direction, which is defined as

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}.$$

This is discretized as a 1D-kernel with size seven. After the first convolution, the resulting image A is again filtered, which results to image B with an effective smoothing of $\sigma = 2$. Subtracting those two images results in the Difference of Gaussians in the first

pyramid level. The next level is generated by resampling image B with a pixel spacing of 1.5 in each direction. Maximas and minimas are determined for each pixel by first comparing it to the neighbors in the same pyramid level, then it is compared to the lower and higher pyramid level. Since the procedure stops after the first comparison fails it is efficient and faster compared to the step of creating the pyramid.

At each pixel in the pyramid, the image gradient magnitude and orientation are computed using pixel differences. For each key point, the orientation is determined by using the peak in an orientation histogram, where the gradient orientations are accumulated in a 36 bin histogram, each bin covering ten degrees.

Now that we have an interest point with a corresponding scale and orientation the local descriptor can be calculated, which is based on gradient features in the pyramid level where the keypoint was detected and in one higher level.

There are other variants of SIFT like PCA-SIFT [KS04] or GPU-based versions which are described later.

SURF: Speeded Up Robust Features

The SURF method by Bay et al. [BTG06] uses a keypoint detector based on the Hessian matrix, but with a basic approximation using integral images. The feature description consists of Haar-wavelet responses in the neighborhood of the interest point. It is well known that these responses can be computed in constant time using the integral image [VJ01][4] again. One orientation is determined for each point and the descriptor is only calculated for this orientation and the detected scale for the interest point. The region around the point is split into 4x4 sub-regions. With four features at each region, the feature vector has a length of 64. The authors also evaluated some variants called SURF-128 (more features) and SURF-36 (3x3 sub-regions). According to the authors, the normal SURF-64 procedure is approximately three times faster than SIFT (354 vs. 1036 ms) while providing better results.

The version using only the upright orientation called U-SURF is not invariant to rotation and cannot be used for our setting, though it is faster.

Gradient Location and Orientation Histogram (GLOH)

Though GLOH uses a feature vector with the same number of dimensions as SIFT, the vector is a more distinctive representation [MS05]. Unfortunately, it is computationally more expensive [BTG06] than even SIFT. Therefore, currently it cannot be utilized for

[4]Actually the integral image, formerly known as summed area tables, is based on [Cro84] and [FS84]

our purpose.

Fast Approximated SIFT

In [GGB06a], the authors describe another optimization of the basic SIFT algorithm. Similarly to SURF [BTG06], this approach makes use of the integral image. Instead of a Difference of Gaussians for keypoint detection in SIFT, a Difference of Means is utilized, since the mean of an image area can be calculated in constant time from the integral image.

For calculating the SIFT descriptor integral histograms are used. A single integral histogram is calculated for each gradient orientation bin. With this pre-calculation the value of any bin in histogram can once again be calculated in constant time, independent of the size of the region patch. The authors showed that the high precalculation costs of the integral histograms pay off when calculating at least 300 key points. This obviously depends on patch size and the number of bins used, for details see [GGB06a]. One disadvantage of this method is that only rectangular regions can be calculated, which has a negative influence on orientation invariance. It is worst at an orientation of 45 degrees. Still, it is only slightly (˜ 10 % matching performance) worse than SIFT itself.

The results are promising. The non-optimized version is around eight times faster than the original SIFT binaries with similar performance. In the conclusion, the authors mention that they intend to try a SIFT-based tracking approach with this algorithm, although there is no publication on this topic yet.

General Purpose GPU for SIFT

There is a lot of literature concerning the implementation of the SIFT algorithm on GPUs, e. g. [HMS+07, SFPG06]. This allows around seven times faster computation [HMS+07] of keypoint extraction and descriptor calculation. Of course, these numbers depend on the type of CPU and GPU. The optimized implementations using the integral image [BTG06, GGB06a] will not profit that much from a GPU implementation, since the integral image cannot be calculated in parallel.

Tracking with Interest Points

Using interest point based recognition methods for tracking requires additional techniques for matching. The problem is that interest points do not directly represent objects. An interest point detector will not distinguish between foreground and background. Given a set of interest points and their feature descriptors in every frame, one has to do some

kind of learning to distinguish between foreground and background points. Besides the feature vector itself, the location of the points to each other can also be used for tracking an object.

Zhou et al. [ZYS09] propose a tracker using SIFT features in combination with mean shift.

2.2.4 Search Window based Recognition

Another approach without the requirement of determining interest points is to use a search window. This search window can be moved across the image to find the objects of interest. Once again local descriptors are calculated and used for classification. It is necessary to find a fast way of computing the features inside the window, since this operation has to be done at each window position. With the local descriptors, the next step is to build a classifier. One way of doing that is to use boosting.

Boosting

The idea behind classifier committees is that, given a task which requires expert knowledge, many experts perform better than just one. For committees, the classifiers used should be as different as possible, but a special case called boosting utilizes the same learning technique for all classifiers. Weak classifiers are combined to make a single strong classifier. The only requirement for every weak classifier is to perform slightly better than 50 % classification rate (in case of a binary classification problem). Every weak classifier gets a weight, which could also be interpreted as confidence. The process of training is done using a training set, better classifiers get a higher weight. By using an additional importance weight for every sample, the process focuses on the more difficult samples. To get an overview of classifier committees and boosting see [Seb02]. Adaptive Boosting (AdaBoost) [FS97] is one of the most popular methods. Begard et al. [BAS08] published a study on various optimizations of the AdaBoost algorithm which they propose, namely a minimization of the necessary operations and an improved weak learner using decision trees.

For our task, a long explicit learning phase is not desired. Boosting is still interesting since [JAS05] introduces an online learning alternative of the boosting algorithm. Grabner et al. apply it to the field of object tracking [GB06, GGB06b, GLB08]. In the latter paper, a semi-supervised approach is introduced to tackle the drifting problem.

Javed's online co-training approach introduced in [Jav05] utilizes boosting as a classifier, though co-training is a general concept for any training task. Javed uses the method for classification of tracked objects, but it is possible to use the algorithm for tracking itself

too. Tang et al. published a paper [TBZT07] describing a method called co-tracking which applies co-training with different classifiers (color histograms and Histograms of Oriented Gradients [DT05]) which train each other.

Histograms of Oriented Gradients

Histograms of Oriented Gradients (HoG) were first introduced by Dalal and Triggs [DT05]. The descriptor is basically similar to the SIFT descriptor, though the application of it is different. The search window block is divided into smaller cells (e.g. 8x8 pixels). For each cell, an orientation histogram is calculated, the gradient value at every pixel casts a weighted vote for a histogram bin. The weight is determined by the gradient magnitude or a function of it. The histograms of the cells form the descriptor for the block. The blocks are tiled dense or even overlapping within the image. This descriptor is applied to tracking in [TBZT07]. Zhu et al. [ZAYC06] as well as Wang and Lien [WL07] used boosting with the HoG descriptor allowing a fast cascade detector (30 fps), while the original algorithm allows just around 1 FPS. Celik et al. [CHHB08] use the HoG descriptor for completely unsupervised learning of a SVM classifier from video sequences, though it relys on a background model. Wu et al. [WZSN08] propose a tracking method with HoG/SVM along with boosted edgelet features (multiple detectors combined), but the classifiers are trained offline.

2.2.5 Integrating Detection into Tracking Methods

There are tracking methods which already integrate object recognition techniques, though this classification is arbitrary. Even a mean shift algorithm does some object recognition, by modeling the object by its histogram. Still, we describe methods which include object recognition to solve the LFR tracking problem.

Particle Filters

Tracking with particle filters is a point tracking approach. Though it is not directly related to object recognition, Yuan Li et al. [LAY+07] combine it with detectors to solve LFR tracking tasks.

Particle filtering is a Monte Carlo (MC) method, it implements a recursive Bayesian filter by MC simulations. The posterior density function as described in the Kalman filter section is represented by a set of random samples with associated weights.

Here the conditional state density function $p(X_t|Z_t)$ is represented with a set of samples, or particles, each with an associated weight. The weight defines the importance of a

sample which is the frequency of observation. At each step t, new samples are drawn from the samples at the previous state $t-1$. The question which arises is which samples to take. One solution is the importance sampling scheme [YJS06], which chooses samples stochastically according to their importance. With these samples, the new object position can be calculated as a weighted sum of a function of each particle and an additional noise term. The function does not need to be Gaussian, so particle filters can be used for non-Gaussian posterior densities.

With a large number of samples, the method becomes equivalent to the normal posterior *pdf* representation and approaches the optimal Bayesian solution.

Cascade Particle Filter Yuan Li et al. [LAY+07] directly address the problem of tracking in low frame rate video (LFR) with a particle filter based approach by integrating detection and tracking into one process. The approach follows the work of Wang et al. [WCG05], which introduces a particle filter approach with Haar-like features, and Ensemble Tracking [Avi07] which uses boosting in combination with a mean shift algorithm. The Cascaded Particle Filter uses multiple observers with different "lifespans". By lifespan, they mean the learning and service period of an observer. An offline trained detection algorithm for instance has both a long training and service time, while a two-frame template matching tracker has a training and service time of one frame each. Short lifespans can cope with appearance changes (like the aforementioned online learning approaches) while long lifespans can prevent the drifting problem. Observers with short lifespans are also used to ruling out non-target candidates early. When particle filters are facing LFR conditions, the search space in the sampling step grows which makes the procedure inefficient. Like cascade based detection algorithms [VJ01], the cascade approach with multiple observers makes it possible to rule out particles without evaluating all of them. For more details see [LAY+07]. According to the authors, this method allows real-time tracking of LFR material with better results than mean shift and a normal particle filter approach.

Particle Filter using SIFT features Wu et al. [WKZL08] proposed a particle filter approach with SIFT features for particle weight calculation. Another approach uses an Unscented Kalman Filter [CSMC08]. Like particle filters, it is an extension to the Kalman filter for tackling nonlinear problems, with SIFT features.

Mean Shift

Porikli and Tuzel propose a low frame rate tracker based on mean shift [PT05], though it cannot cope with moving cameras because they rely on a background model.

2.3 Discussion

Object recognition methods are incorporated into tracking in various ways. For the selection of methods for further investigation the following criteria were taken into consideration:

- The method must not rely on a **background model**, otherwise it is bound to stationary cameras
- The method should not rely on a strictly linear **motion model**, like the Kalman filter, otherwise tracking of abrupt motion in LFR scenarios is impossible.
- The method should do without an offline training phase which is dependent on the specific object class that should be tracked.

The state of the art section shows that there are multiple tracking algorithms fulfilling this criteria, like [CM02, GGB06b, LAY+07] as well as object recognition methods which can be used for tracking [Low99, DT05, BTG06, GGB06a]. Two categories of object recognition techniques were described, those around interest points and search window based techniques. We decide to experiment with the online boosting algorithm by Grabner et al. [GGB06b], which is a search window based technique, since it has promising results. The technique itself is being developed over four years with many extensions. Unfortunately there is no code available, so we have to use our own implementation.

We decide to develop a novel algorithm using interest points, since there are rarely any methods in this field. The first step is to find interest points in the image and extract feature vectors. This step can be seen as an isolated step, so we decide to use the SURF method for interest point detection and feature extraction. A SURF implementation called OpenSURF is freely available[5]. According to the authors of SURF, it provides a better performance compared to SIFT while allowing faster calculation, which is necessary for developing a real-time tracking algorithm. SURF is already used for tracking by a recent publication of Ta et al. [TCGP09], but the authors focus more on the efficient organization of the keypoint extraction, while we take the keypoints and their feature descriptors as given. In [HYLL09], SURF in combination with a workflow similar to ours is used, but with an online EM algorithm for modeling the relation between point and object motion and a maximum likelihood method to estimate object motion. Moreover, they incorporate object structure with a graph matching approach.

We compare our algorithms with mean shift, which is a widely used technique. The implementation we use does not incorporate any background model, contrary to the

[5] see `http://code.google.com/p/opensurf1/`, last visited 8. 11. 2009

technique of Porikli and Tuzel mentioned earlier [PT05]. Although it benefits from color images, it still works with greyscale sequences. It inherently provides rotation invariance, which helps when tracking objects with changing pose, since it is based on color histograms.

Chapter 3

Methodology

In this chapter, we describe our methods in detail. We start by stating the general preconditions in section 3.1. Then we continue with interest point based tracking (IPTracker) in section 3.2 and with tracking via online boosting (OBTracker) in section 3.3. Speed is a major requirement to real-time tracking methods, so we show how the methods can be parallelized to increase speed on modern processors in section 3.4. We conclude this chapter by describing the metrics we use for evaluation in section 3.5.

3.1 General Preconditions

Both implemented algorithms, namely online boosting and the interest point tracker, require the object to be located in the first frame of the sequence to allow one-shot learning. It is not necessary to do any offline learning. Locating the object in the first frame is not handled in this thesis, but it would be easy to use an additional algorithm for initialization, for example a face detector for face tracking. There is a good reason why we do not cover the initialization: Such algorithms are always application-dependent and have to use additional assumptions, like a specific object detector for face detection or a background model to allow detection of foreground (moving) objects.

3.2 Interest Point Tracking

We describe a novel interest point based tracking method [KK10] which we developed. It uses the SURF keypoint extraction method and an online method based on point matching, which does not require offline learning, for tracking. We use the SURF implementation called OpenSURF which is freely available[1] for keypoint extraction.

[1] see `http://code.google.com/p/opensurf1/`, last visited 8. 11. 2009

3.2.1 Interest Point Detection and Feature Extraction with SURF

The first step is to find interest points in the image and extract feature vectors locally around the points. This step can be seen as an isolated step. We decided to use the SURF method [BTG06], because it provides a better performance and faster calculation compared to SIFT. Still any other interest point method would also be possible, since the result required from this step is just a set of interest points, each containing a 64-dimensional vector of features and a position $\mathbf{x}$.

Interest Point Detection

An interest point method consists of two steps. First, interest points need to be located (interest point detection), then feature vectors are extracted at those points using information from the surrounding area. The authors of SURF name the detector Fast Hessian [BTG06], because it is a basic approximation of a Hessian matrix based detector, which relies on the determinant of the Hessian matrix. The Hessian matrix $\mathcal{H}(\mathbf{x}, \sigma)$ at the point $\mathbf{x}$ with scale σ is defined as

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}.$$

L is the convolution with the respective second order derivate of the Gaussian function. This function has to be discretized in order to use it for image convolution and it also needs to be cropped.

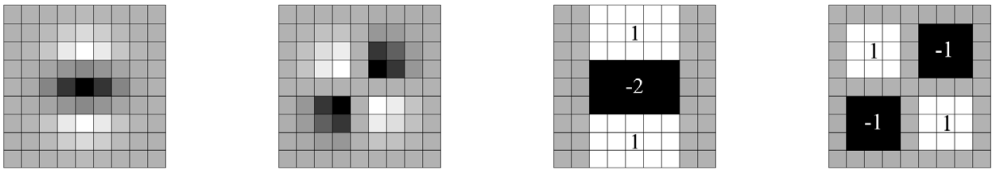


Figure 3.1: The discretized and cropped Gaussian second order partial derivatives (y and xy direction) and the box filter approximations. Figure taken from [BTG06].

In SURF, an approximation based on the integral image is used. In the integral image, the value at every point $\mathbf{x}$ is defined as the sum of all values in the region formed by $\mathbf{x}$ and the origin, or formally

$$I_\Sigma(\mathbf{x}) = \sum_{i=0}^{i<x} \sum_{j=0}^{j<y} I(i, j).$$

This allows the calculation of image areas in constant time. To calculate the area of an axially parallel rectangle determined by the 4 points **a**, **b**, **c** and **d**, where **a** is the top left and **d** is the lower right point, one just needs to calculate $I_\Sigma(\mathbf{d}) - I_\Sigma(\mathbf{b}) - I_\Sigma(\mathbf{c}) + I_\Sigma(\mathbf{a})$.

According to the author's evaluation, the performance of this approximation is comparable to the implementation using discretized and cropped Gaussians. In figure 3.1 the original function and the approximation is shown. The approximations are denoted as D_{xx}, D_{yy} and D_{xy}. As seen in the figure the weights are kept simple using -1 and $+1$ or $+1$ and -2 respectively. In order to keep the approximation similar to the original determinant the weight of the D_{xy} component is reduced by $\frac{|L_{xy}(1.2)|_F \times D_{xx}(9)|_F}{|L_{xx}(1.2)|_F \times D_{xy}(9)|_F} = 0.912... \simeq 0.9$, where $|x|_f$ is the Frobenius norm. This follows on from the idea of weighting the approximations D with a factor in the form of $\frac{L}{D}$, which is done to guarantee energy conservation. Though this factor is not equivalent for different scales it is kept constant since it had no significant impact in the originators experiments [BETVG08]. Actually, the reweighted determinant would have the form $w_1 D_{xx} D_{yy} - w_2 (D_{xy})^2$, but since the authors assume a constant weight for all scales only the relative weighting between the two terms is important. With this weighting the approximated determinant is defined as follows:

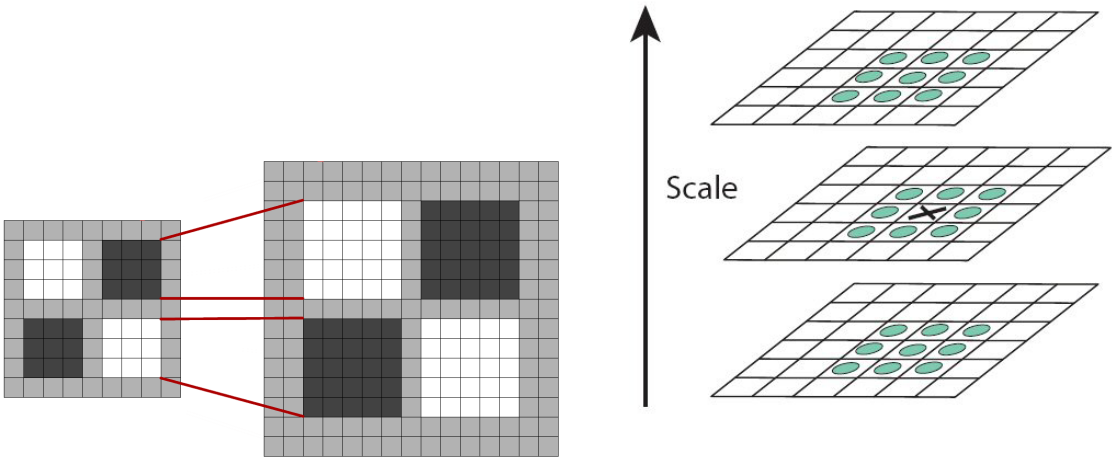
$$\det(\mathcal{H}_approx) = D_{xx} D_{yy} - (0.9 D_{xy})^2.$$

SURF follows a scale-space approach like for instance SIFT. With the box filters there is no need to scale the image itself. It is sufficient to scale the filters, which does not need additional computations. It starts with a 9×9 filter, which is the smallest scale. This corresponds to a Gaussian second order partial derivative with $\sigma = 1.2$. The filter sizes increase at least by 6 with every scale, which is necessary to keep the filter structure, see figure 3.2(a). The filter response is normalized with respect to the filter size. The scale-space is divided into a number of octaves, which correspond to a doubling of the scale. Every octave is again divided into a constant number of scale levels.

Keypoints are localized by using a non-maximum suppression in a $3 \times 3 \times 3$ neighborhood, including the adjacent scales, this can be seen in figure 3.2(b). The detected maximas are interpolated in scale and image space.

Feature Extraction

In order to extract features at the localized points, orientation is determined first. This is done by calculating Haar wavelet responses in x and y direction in a sliding window around the interest point according to its scale. The responses are interpreted as coordinates in a 2-dimensional orientation space. A sliding orientation window of size $\frac{\pi}{3}$ is used around the

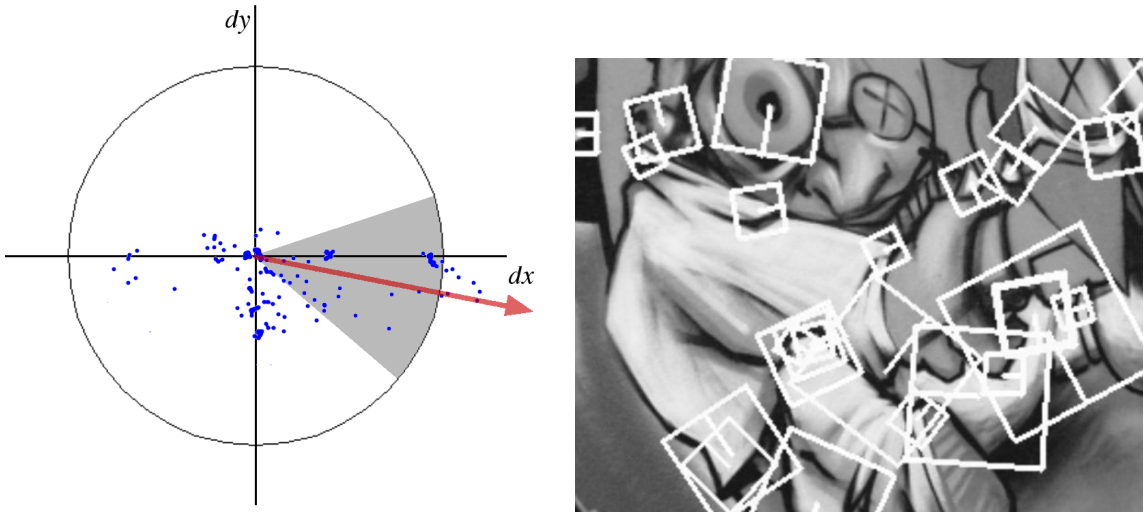


(a) Filters for two successive scale levels (from [BETVG08])

(b) Non-Maxima suppression[2]

Figure 3.2: Scaling and non-maxima suppression

center of the orientation space. All points in this window are summed up, which leads to an orientation vector. The longest of these vectors defines the dominant orientation. Figure 3.3(a) shows this space and the calculation of the orientation vector.



(a) Sliding window for orientation assignment

(b) Regions for feature extraction with their orientation

Figure 3.3: Determining orientation and descriptor region. Figures taken from [BETVG08].

[2]Taken from `http://opensurf1.googlecode.com/files/OpenSURF.pdf`, last visited 1. 2. 2010

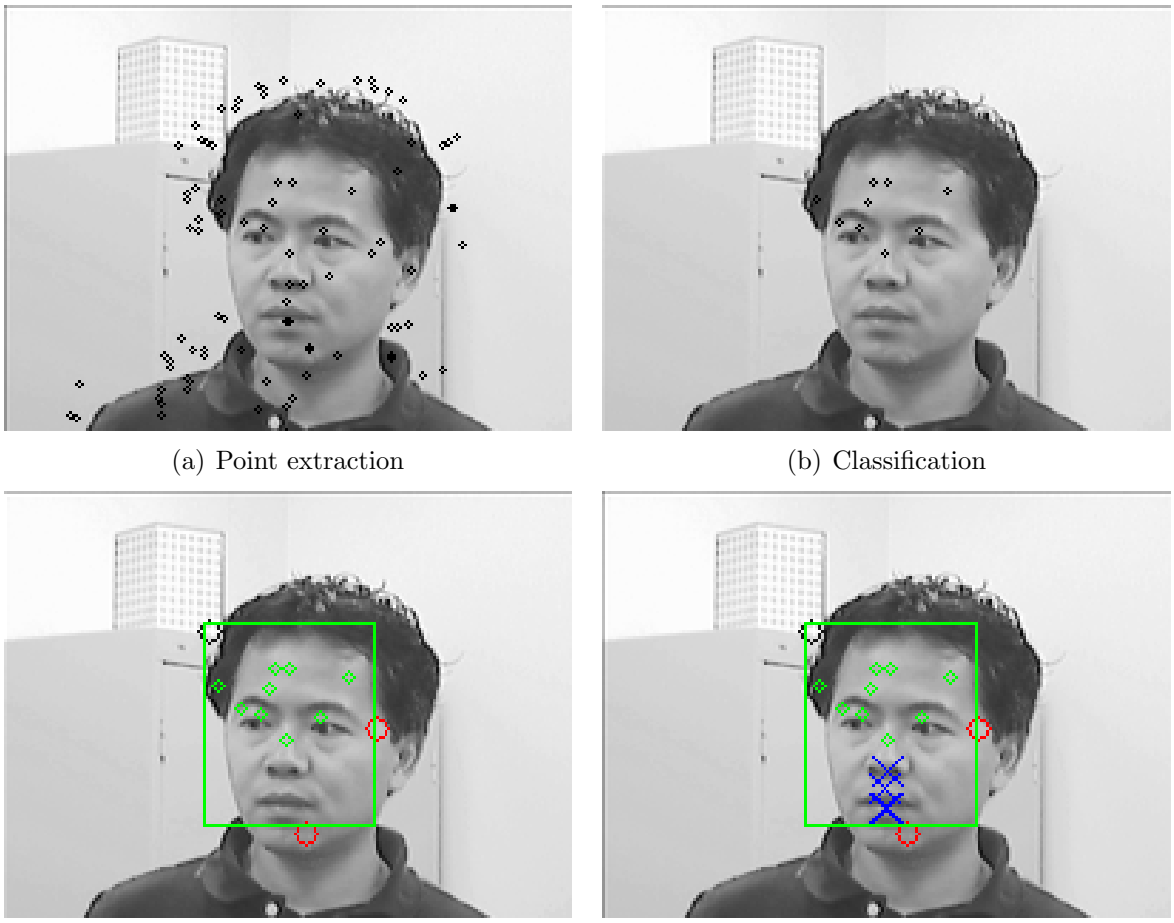
Aligned with the orientation and according to the scale, a rectangular area is selected for feature calculation. Figure 3.3(b) shows samples for such regions. These regions are split into 4x4 square sub-regions to preserve spatial information. In each of these sub-regions 5x5 regularly spaced sample points are taken where Haar wavelet responses in x and y direction are calculated. These values are summed up and weighted with a Gaussian function around the interest point to increase robustness. This leads to the first $4 \times 4 \times 2 = 32$ entries of the 64-dimensional feature vector. The second 32 values are the sum of the absolute wavelet responses in both directions, which is done to include information about polarity.

The nature of the Haar features makes the descriptor invariant to an additive change (change in illumination). It is normalized to a unit vector in order to achieve invariance to a scale factor (change in contrast).

3.2.2 Basics and Definitions

The keypoint extraction step results in a set of keypoint locations with their corresponding feature vectors, which is the input of the tracker. Figure 3.4 shows the basic workflow of the tracking method. Before coming to the detailed explanation we first define and explain functions, variables and constants used in the algorithm description. There are

- the image sequence $I_0..I_{N-1}$,
- interest point locations $\mathbf{p}_1..\mathbf{p}_k \in \mathcal{I}$ for each frame,
- relative interest point location $\mathbf{p}_{rel} \in \mathcal{I}$ of a specific interest point within the bounding box rectangle,
- feature vectors for those points $\mathbf{f}_1..\mathbf{f}_k \in \mathcal{F}$,
- initial bounding box rectangle $R_0 \subseteq \mathcal{I}$,
- bounding box rectangle for every frame $R_1..R_{N-1} \subseteq \mathcal{I}$,
- $t \in \{1..N-1\}$ is the index of the current frame,
- two sets S_{fg}, S_{bg} with points and their corresponding feature vectors,
- weight adaption constants w_{Cool}, w_{TP}, w_{TN} and initial weight w_{init},
- feature vector matching threshold c_{dist} and feature vector adaption factor α,
- size limits for the sets $n_{fg,max}$ and $n_{bg,max}$,



(a) Point extraction

(b) Classification

(c) Bounding box and weight update

(d) Replacing points

Figure 3.4: This illustration shows the tracking algorithm works. (a) In the first step, interest points are extracted from the whole image. (b) In the second step the points are classified into foreground (those displayed) and background points. (c) Then a bounding box is calculated according to the foreground points and false detections get a penalty (big circles). (d) In the last step, new points are added to the background and foreground sets (blue crosses).

- a distance threshold in feature space g_{thresh} which prohibits new foreground points too close to known background,
- UpperLeft$(R) \in \mathcal{I}$ is the top left point (the one with lowest x and y coordinate) of axially parallel rectangle R,
- Dimension$(R) \in \mathcal{I}$ is a 2-component vector with the width and height of the rectangle R and
- $\times$ denotes a component-wise multiplication.

The image space $\mathcal{I} = \mathcal{N}^2$ is the 2D coordinate system of the image which contains the point locations $\mathbf{p}$. Feature space $\mathcal{F}$ is the high-dimensional (for SURF 64) feature space which contains the feature vectors f.

3.2.3 Initialization

Given a bounding box for initialization and a set of interest points $\mathbf{p}_{1..k}$ with their corresponding feature vectors $\mathbf{f}_{1..k}$ for the first frame, we separate the interest points into two sets S_{fg} and S_{bg}, with an initial weight of w_{init}. The two sets of interest points with their feature descriptors and their weights are the learned knowledge on which tracking is based. Additionally we calculate for all points in the foreground set $\mathbf{p}_{rel}$, using $\mathbf{p}_{rel} = \mathbf{p} - UpperLeft(R_0)/Dimension(R_0)$ (/ here denotes a component-wise division), which is the relative position in the bounding box. This knowledge is updated in every following tracking step to allow adaption to changing object appearance. The update step is also necessary for improving our model which was just created by one-shot learning.

Our set of foreground and background points has no fixed size, but is limited to the upper bounds $n_{fg,max}$ and $n_{bg,max}$. The sizes of these sets have a direct influence on the runtime of the algorithm. Limiting these sets is also important to enable selection of the best, which means the most distinctive, points.

3.2.4 Tracking

Once the initialization is done, the object is tracked in every consecutive frame. The algorithm requires a set of non-fixed size of extracted interest points $E = \{\langle \mathbf{p}_0, \mathbf{f}_0 \rangle .. \langle \mathbf{p}_k, \mathbf{f}_k \rangle\}$ with their descriptors.

First of all, the interest points are divided into three pairwise disjoint sets S_{pos}, S_{neg} and $S_{unmatched}$ (and $S_{pos} \cup S_{neg} \cup S_{unmatched} = E$). This is done by matching them to the foreground and background sets S_{fg} and S_{bg}, for details see algorithm 1.

After these sets are created the bounding box R_t is determined. This is done by calculating the upper left corner of the rectangle using the relative positions of the interest points which are known from the first frame in which the interest point was added to the set S_{fg}. For all points in S_{pos}, which corresponds to a subset of S_{fg} because of the matching step, the upper left corner of the rectangle is predicted by subtracting $\mathbf{p}_{rel} \times \text{Dimension}(R_0)$. The average of those predictions is taken for determining the bounding box R_n with the size of R_0.

Using this information, all points in S_{pos} are classified into two sets TP (true positives) and FP (false positives) according to their location. The weights for all matched points

Algorithm 1 Point matching with S_{fg} and S_{bg}

```
for i = 0..k do
  ⟨best_0, best_1⟩ = min(distance(f(E_i), f(S_fg) ∪ S_bg))
  if best_0 < c_dist best_1 then
    if best_0 ∈ S_fg then
      S_pos = S_pos ∪ best_0
    else
      S_neg = S_neg ∪ best_0
    end if
  else
    S_unmatched = S_unmatched ∪ best_0
  end if
end for
```

from TP are updated using $w = w + w_{TP}$. The points in S_{neg} are also classified into FN (false negatives) and TN (true negatives) and the weights of the points in TN are updated using $w = w + w_{TN}$. After this step, all weights in both sets S_{fg} and S_{bg} are decreased using $w = w - w_{Cool}$. This cooldown ensures that points which are rarely matched are getting a low weight which allows replacement, later on.

After these classification steps are done, the update step is performed. All matched points which were correctly classified get their feature vectors updated using $f_t = \alpha f_t + (1 - \alpha) f_{t-1}$. Then new candidates are inserted into the sets S_{fg} and S_{bg} by either adding them, when the set has not reached $n_{fg,max}$ or $n_{bg,max}$ yet, or by replacing existing points. For replacement in S_{fg} three input parameters are important: The number of replacements to be made n_{torepl}, the set of new candidate points P_{new} and the set of candidates for replacement P_{old}. Then $\min(n_{torepl}, |P_{new}|, |P_{old}|)$ elements are replaced by choosing randomly (uniformly distributed) from the sets P_{new} and P_{old}.

n_{torepl} is calculated with the goal of filling up the number of positively classified interest points (TP) in every frame to the number of initial interest points (TP_{init}). This is done by using $n_{torepl} = \frac{TP_{init} - TP}{2}$. If TP is greater than TP_{init}, n_{torepl} is simply set to zero. Factor two is included to learn more conservatively by replacing less points.

P_{new} is the set of candidates for replacing existing interest points, which is a specific subset of $S_{unmatched}$. Namely, all those points that are within the rectangle R_t and which are in feature space more than g_{thresh} away from the closest background interest point.

P_{old} are points in S_{fg} which have a weight lower than w_{min}.

Candidates for S_{BG} are added to the set, then set items are removed using roulette wheel selection according to their weights to reach $|S_{bg}| = n_{bg,max}$ again.

3.2.5 Limitations

The algorithm uses a fixed object size (size in the first frame) and relies on that size to identify foreground interest points (around the centroid of positively detected points). If the object changes in size, actual object points may be rejected (if it becomes bigger) or background points are classified positively (if it shrinks). Since the interest point detector itself is scale invariant this is an unnecessary limitation which could be avoided by applying a more sophisticated technique, for example by calculating and storing the relative position of points to each other to detect changes in scale and adapt the expected object size.

3.2.6 Other attempts

Some attempts did not lead to success in our experiments, but we still want to document them. We tried using kNN for distinguishing between foreground and background points in the matching procedure. It turned out to be a bad idea, since there are a lot more background points than foreground points, in our experiments by a factor of ten. This means that it is much more likely to have more background than foreground points in the neighborhood in feature space. The second reason is that there is no dense foreground point class in feature space. There are many different foreground points since every key point of the object corresponds to a different region, and the various regions of the object do not have a uniform appearance. For instance the eye in a human face looks different to the mouth. So, when looking for the k nearest neighbors, the next closest point after the matched eye might be from the background instead of other points of the face region with a different appearance.

We also tried classifying key points with a Support Vector Machine (SVM). Even when using a long offline training phase, the classification performance was only around 60 %, which did not allow tracking. Single object keypoints are not sufficiently discriminative for SVM classification. The reason is similar to the problem with kNN. An object contains keypoints with a potentially very different appearance, so one needs to create a single classifier classifying multiple unknown classes as positive. It would be better to distinguish between the different classes in the supervised training process, but that would require labeled data for the different classes.

3.3 Boosting

Online boosting can be applied to object tracking [GGB06b]. We describe the method, as well as its theoretical foundations and the implementation.

Boosting is an ensemble method, which combines different weak classifiers $h_1(\mathbf{x})..h_m(\mathbf{x})$ to a single strong classifier $H(\mathbf{x})$. It is important to have diversity in the weak classifiers, since using the same classifier on the same data multiple times would not lead to an improvement. According to [Gra08], different hypotheses can be constructed by using the same classifier on different subsets of the training data (bagging), differently weight the training data (boosting) or by using different learning algorithms (stacking). Each individual classifier should be as accurate as possible, but they should also provide diversity, which is contradictory. The more accurate the classifiers become, the more they have to agree. Boosting goes back to [Sch90] and [Fre95], and became popular in the field of computer vision with the work of Viola and Jones [VJ01].

3.3.1 AdaBoost

One variant of boosting is the AdaBoost algorithm by Freund and Schapire [FS97]. This method is also the motivation for the online variant used in the tracking algorithm. We now describe the discrete AdaBoost algorithm, where the weak classifiers h are binary classifiers, or $h : \mathcal{X} \rightarrow \{-1, +1\}$. Recall that the requirement for weak classifiers is to have an error rate which is less than 50 %. The strong classifier is created by using the weighted weak classifiers

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

Now we have to learn the weak classifiers h and the voting weights α. A training set $S \subseteq \mathcal{X}$ with positive and negative samples and initial sample weights $\mathbf{w}_0 = (w_{0,1}..w_{0,L}), w_0, l = \frac{1}{L}$ as well as the target labels y_l for all samples $l = 1..L$ is given. A new hypothesis is chosen at every step based on the training set S and the current sample weights $\mathbf{w}$

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

In this step a new weak classifier with minimal error on the training set with current sample weights is selected. The error e_n is also implicitly calculated with this operation, it is the sum of all weights of the samples which were classified incorrectly $e_n = \sum_{l:h_n(\mathbf{x}) \neq y_l} w_{n,l}$.

Based on this error, the weak classifier gets a voting weight

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

Since the sample weights get normalized after every update the error e_n is between 0 and 0.5. It cannot be greater because weak classifiers have to be better by definition. If 0, the algorithm terminates, since a weak classifier is found which perfectly classifies the training set. The smaller the error, the greater becomes the influence of the new classifier h_n. Finally, the new weight distribution is calculated so that misclassified samples get a higher weight in order to focus on the harder samples.

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

These weights get normalized afterwards. In each iteration of the process a new classifier is chosen, decreasing the error.

For a good graphical explanation of AdaBoost, see the slides of Sochman and Matas[3]. Worth mentioning for this method are the interesting properties. There is an upper bound for the training error of the strong classifier [FS97] and the training error drops exponentially fast in the number of iterations [FSA99]. The method also provides good generalization performance since it is related to maximization of the margin [SFBL97].

3.3.2 Online Boosting

Grabner et al. [GB06] introduce a special variant of online boosting for feature selection, it is based on the online variant of AdaBoost.

Since it is an online method, it has two main steps: classification, which is the actual classification of an unlabeled sample, and update, which is the update step with a labeled (known, or at least assumed label) sample. This approach consists of N selectors, each containing a set of M weak classifiers. The purpose of a selector is to select the currently best weak classifier in its set, since for classification only the best weak classifier is utilized. This results in a classification step complexity of $O(N)$ in the number of selectors. The update step updates every weak classifier in every selector, so the complexity is $O(N \times M)$.

Algorithm 2 describes the update step of online boosting. The input for an update step is the feature vector $\mathbf{x}$ and the target label y. Every selector is updated. They are independent from each other, except the importance weight of the sample. First of all, the

[3] see `http://cmp.felk.cvut.cz/~sochmj1/adaboost_talk.pdf`, last visited 28. 2. 2010

Algorithm 2 Update-Step of Online Boosting

for $n = 1..N$ **do**
  //update weak classifiers, estimate errors
  for $m = 1..M$ **do**
    $h_{n,m}^{weak} = update(h_{n,m}^{weak}, \langle \mathbf{x}, y \rangle, \lambda)$
    //check if sample is classified correctly
    if $h_{n,m}^{weak}(\mathbf{x}) = y$ **then**
      $\lambda_{n,m}^{corr} = \lambda_{n,m}^{corr} + \lambda$
    else
      $\lambda_{n,m}^{wrong} = \lambda_{n,m}^{wrong} + \lambda$
    end if
    $e_{n,m} = \frac{\lambda_{n,m}^{wrong}}{\lambda_{n,m}^{corr} + \lambda_{n,m}^{wrong}}$
  end for
  //choose best weak classifier
  $m^{best} = argmin_m(e_{n,m})$; $m^{worst} = argmax_m(e_{n,m})$
  $e_n = e_{n,m^{best}}$; $h_n^{sel} = h_{n,m^{best}}^{weak}$
  if $e_n = 0 \wedge e_n > \frac{1}{2}$ **then**
    exit
  end if
  //calculate weight for selector n, lower error $\Rightarrow$ higher weight
  $\alpha_n = \frac{1}{2} \ln(\frac{1-e_n}{e_n})$
  //update importance weight
  if $h_n^{sel}(\mathbf{x}) = y$ **then**
    $\lambda = \lambda \frac{1}{2(1-e_n)}$
  else
    $\lambda = \lambda \frac{1}{2(e_n)}$
  end if
  //replace worst classifier
  $h_{n,m^{worst}}^{weak} = new\ h \in \mathcal{F}_n$
end for

error $e_{n,m}$ is calculated for all weak classifiers in the selector, based on the classification result and the importance weight. The higher the importance, the more influence the sample has on the estimated error of the classifier. Thus, the best and worst weak classifiers $h^{weak}_{n,m^{best}}$ and $h^{weak}_{n,m^{worst}}$ are determined. The best weak classifier will be used for the next classification step, representing its selector. The worst weak classifier is replace by a randomly picked new one at the end of the update step. Based on the error of $h^{weak}_{n,m^{best}}$, the weight α_n of the selector is calculated. The importance weight λ of the current sample is also updated and used for further selectors, starting with a default value for the first selector.

The classification is done by $\text{sign}(\sum \alpha_n \cdot h^{sel}_n(\mathbf{x}))$, the weighted sum of the weak classifier results, taking the best classifier from each selector.

Using the selector approach, a huge underlying feature space is possible, while only the current $N \cdot M$ classifiers have to be updated at every step. By replacing the worst classifier in every update step, the whole feature space is explored over time. Every selector has its own feature pool, $\mathcal{F}_n$, allowing different kinds of features. The algorithm adapts the weight α_n of the selectors and therefore, chooses the best suited features for the problem. Therefore, the algorithm allows feature selection. Grabner et al. [GGB06b] show that the algorithm is able to switch between different feature types when the classification problem changes its characteristics.

3.3.3 Online Boosting for Tracking

The task of object tracking suggests an online learning approach. With every step one gets the same object again, just in a different pose. When assuming that the area around the object does not contain an object of the same type, the surrounding area can be used as negative training samples and the object itself as a positive sample. During the tracking procedure, the online learner should be able to learn the object representation better and better over time.

Applying online boosting to object tracking is straight forward. In the first initialization step of a new object, the object itself is trained as a positive sample while overlapping areas around the object are taken as negative samples. Under the assumption that this initialization is correct (the given object is really the correct object) and there are no other similar objects in the surrounding background, this step will not introduce errors. In every further frame, the object is searched in the area around the last occurrence by using a full window scan. It is possible to include a motion prediction model to the system, which would increase speed and improve quality, while it limits the allowed movement of the object. The increase in speed is due to the smaller search space. The improved

quality is caused by avoiding potential false positives in the larger search space. For our experiments, we chose a fixed size region around the object to allow even fast moving cameras and abrupt changes in motion.

The object is assumed to be at the maximum of the feature responses from the window search. An additional threshold is included to cope with situations where object is lost and to avoid training with poorly fitted windows. Only objects where the classifier is sufficiently confident that it is correctly classified are used for further training. This reduces the problem of learning the background or other objects and allows recovery when the object is lost.

3.3.4 Classifiers and Features

In our implementation we use Haar-like features, which were also used by Viola and Jones [VJ01]. In figure 3.5, we show various Haar feature types. The feature value at a specific image location is determined by summing up the areas marked in the figure. The white area is subtracted from the black area in order to calculate the feature value. Determining the areas can be done in constant time using the integral image. These Haar-like features at various scales represent the possible feature types we allow. The classification decision is based on a threshold which is determined online by modeling the probability distribution of both positive and negative samples. The boosting procedure enables one to use any classifiers and features. In [GGB06b], also orientation histograms and local binary pattern based features are used.

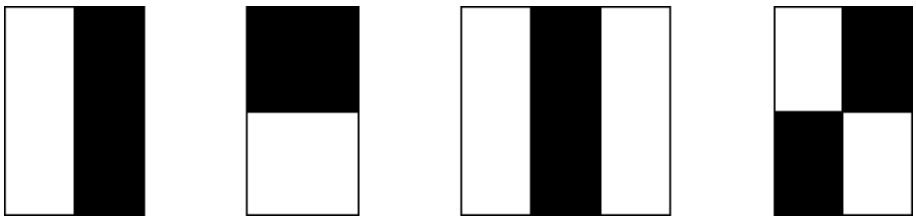


Figure 3.5: Haar-like features

3.3.5 Limitations

The presented tracking algorithm shows some limitations. It works with a fixed size search window, which leads to tracking errors when the size of the object changes, which is most commonly caused by a changing perspective. Depending on the underlying features, it would be possible to scale the search window, but we performed no experiments in

this direction, since it would require a fundamental redesign of the implementation and constrain the choice of features. There is also an inherent limitation due to the search window, which limits the maximal motion of the object from one frame to the next. This could be improved by using a motion model.

Our implementation of online boosting is very limited compared to the original implementation by Grabner et al., whose development took more than four years. We did not implement the improvements of the later papers, like Semi-Supervised Learning [GLB08, GGLB09], nor did we implement all features used in the original paper, but instead relied on Haar features only.

3.4 Parallelization

Both the IPTracker and the OBTracker would benefit from the use of multiple processing units (CPUs). Parallelization is in both cases straightforward. In case of a modern quad-core CPU, which nowadays has become a standard, the calculation can be up to four times faster, depending on how many steps can be calculated in parallel, and on synchronization overhead. Both algorithms rely on massive parallel computations and the results should therefore be close to the theoretical maximum. The final performance optimization is no primary goal of this thesis, so we just describe the steps theoretically.

IPTracker The interest point calculation and detection can be calculated in parallel by dividing the image into regions, in which each thread calculates points. The matching step can also be parallelized directly, since the foreground and background sets are not manipulated. For the later steps (classification), it is not that easy, but these steps act only on a small subset of the original points (the matched points). Therefore, it would be advisable to analyze the potential gain before occupying oneself with parallelization.

OBTracker For the classification step of online boosting, the classification of every search window position can be done independently in parallel. The update step can be parallelized by updating the classifiers in parallel for every selector. This is possible, since the classifier update operates on constant data and just the identification of the best and worst classifiers has to be synchronized.

3.5 Evaluation Methods

All our tracking methods have a fixed object size for the whole video and they only give a bounding box of that size. Therefore, a pixel-based labeling does not make sense which is quite fortunate since the task of labeling long videos pixel-wise is extremely time-consuming. We also did not find any suitable available ground truth tracking data which is labeled pixel-wise. Thus, we are satisfied with ground truth bounding boxes of the object.

One of the evaluation videos (Dudek Face) provides stable points for the videos. In every frame, major points of the face around the eyes and mouth are labeled. Since this is not directly suitable for evaluating our algorithms, we use the centroid of the points and put a bounding box around it according to the object size. Alternatively, we could use the data by directly using the stable points and require all of them to be inside the calculated bounding box. However, we decide to use the indirect method because this is the more general approach. It is easier to label a bounding box in any image domain than special stable points. So it is possible to use labeled data from other sources in the same evaluation method or label new videos, not only in the face tracking domain.

What we actually want to evaluate is whether the object is tracked over the whole sequence. We do not want the precise positioning of the bounding box to have an influence on the results, since this would introduce ambiguities related to the definition of the object, like the question of whether the hair of a person is part of the face in the face tracking domain.

There are many approaches to tracking evaluation in the literature, especially around the PETS (Performance Evaluation of Tracking and Surveillance) workshops. A very detailed survey was published by Baumann et al. [BBE+08]. Collins et al. [CZT05] define an evaluation framework including metrics with overlapping bounding boxes, which is basically all we need. Based on their metrics, we define the following: The tracking rate is the number of frames in which the object has been correctly tracked, divided by the number of frames N in the sequence. Correctness is defined in terms of overlapping bounding boxes of the ground truth bounding box $R_{GT,i}$ and the tracker bounding box R_i.

$$TR = \frac{\sum_{i=0}^{N-1} x_i}{N}$$

$$x_i = \begin{cases} 1, & \text{when } R_{GT,i} \cap R_i > t_{ov} \max(R_{GT,i}, R_i) \\ 0, & \text{otherwise} \end{cases}$$

Instead of the union, the maximum function is chosen to keep the penalty of a deviation

of the bounding box smaller. It is not possible to use just the area of $R_{GT,i}$, otherwise a bounding box R_{tricky} covering the whole image would always be correct. This is also the reason why t_{ov} is an important parameter. For our evaluation, we have set it to 0.1, which is small, but since we are comparing areas, even on first sight strongly overlapping bounding boxes have a smaller overlap than expected.

Chapter 4

Experiments & Results

In this chapter, we start by describing the sequences we use for our evaluation in section 4.1. Then we describe the methodology and show the results of our experiments in section 4.2. These include comparisons to other trackers as well as the analysis of specific properties of the methods. In section 4.3, the acquired results are discussed and set in relation to the objectives of this thesis.

4.1 Evaluation Data

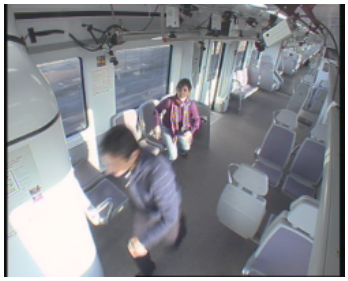
To carry out the evaluation, we need sequences which can be used for experiments. For comparison with other methods, we prefer to use datasets which were already used by other researchers instead of recording our own.

4.1.1 Summary of Tracking Evaluation Sequences

In order to find data for the evaluation, we carried out an extensive research on sequences which are used in other publications. Here, we give a summary of publicly available sequences. We list all tracking sequences alphabetically, and not only those suitable for our purpose. Unfortunately, some of the older sequences are not available on the internet anymore (for instance CLEAR 2006, PETS2005 or PETS2006). So, since the field is vast, this summary focuses on sequences which are still freely available.

Babenko Boris Babenko et al.[1] collected a number of sequences which they used for evaluating their MIL Tracker [BYB09]. Parts of them are created by themselves, parts are from other institutions. All videos datasets show a single object to track and they all

[1] see `http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml`, last visited 19. 04. 2010



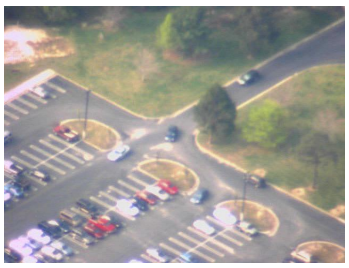
(a) BOSS dataset

(b) CANDELA dataset (abandoned object)

(c) CANDELA dataset (intersection)

(d) CAVIAR Vivid dataset

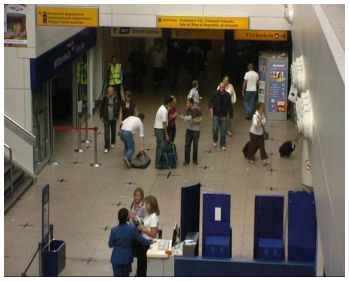
(e) CLEAR 2007 dataset



(f) DARPA Vivid dataset

(g) ETISEO dataset

(h) i-LIDS dataset (abandoned baggage)



(i) PETS 2007 dataset

(j) SELCAT dataset

(k) TRICTRAC dataset

Figure 4.1: Sample screenshots from various video datasets

include a ground truth. There are sequences with partial occlusions, cluttered background and with nonstationary cameras. Sample screenshots of the "Occluded Face" sequences are shown in figure 4.2.

BOSS The BOSS (On Board Wireless Secured Video Surveillance) dataset[2] shows a public transport surveillance scenario with multimodal (audio and video) data and contains events like fighting and theft. See figure 4.1(a) for a sample image.

CANDELA The CANDELA (Content Analysis and Network DELivery Architectures) dataset[3] provides two scenarios, one abandoned object scenario and one intersection scenario with people and cars. See figures 4.1(b) and 4.1(c) for sample images.

CAVIAR This dataset depicts a public safety scenario in a shopping center. It includes people walking, meeting and fighting as well as an abandoned baggage scenario. It is freely available for download[4]. See figure 4.1(d) for a sample image.

CLEAR 2007 At the Classification of Events, Activities and Relationships (CLEAR) Workshop 2007 a challenge took place where multimodal sequences were provided. They show a meeting room scenario recorded with multiple cameras and microphones. See figure 4.1(e) for a sample image.

DARPA Vivid The DARPA Vivid dataset provides airborne ground surveillance videos[5]. See figure 4.1(f) for a sample image.

ETISEO In the scope of the Evaluation du Traitement et de l'Interprétation de Séquences Video (ETISEO) project various video sequences for evaluation are provided, all with stationary cameras[6]. See figure 4.1(g) for a sample image.

i-LIDS The i-LIDS dataset provides videos of abandoned baggage, parked vehicle, doorway surveillance, sterile zone and multiple camera tracking scenarios[7]. See figure 4.1(h) for a sample image.

[2]see `http://www.multitel.be/image/research-development/research-projects/boss.php`, last visited 8. 12. 2009

[3]see `http://www.multitel.be/~va/candela/`, last visited 8. 12. 2009

[4]see `http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/`, last visited 22. 11. 2009

[5]see `https://www.sdms.afrl.af.mil/request/data_request.php#anotcollect`, last visited 22. 11. 2009

[6]see `http://www-sop.inria.fr/members/Francois.Bremond/topicsText/etiseoProject.html`, last visited 22. 11. 2009

[7]see `http://scienceandresearch.homeoffice.gov.uk/hosdb/cctv-imaging-technology/video-based-detection-systems/i-lids/ilids-datasets-pricing/`, last visited 22. 11. 2009

PETS 2007 The IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) provides a challenge with sequences for evaluation every year. In 2007 the scenario was left luggage[8]. See figure 4.1(i) for a sample image.

SELCAT The SELCAT (Safer European Level Crossing Appraisal and Technology) dataset[9] provides videos of a railway crossing. The goal in the scenario is to detect dangerous situations. See figure 4.1(j) for a sample image.

TRICTRAC The TRICTRAC dataset[10] contains videos of a rendered soccer field scenario with multiple camera views. See figure 4.1(k) for a sample image.

4.1.2 Requirements

There are several constraints on the type of evaluation data which need to be satisfied in order to be able to use them for an evaluation of our algorithms. Most important is the size of the object to be tracked. It has to have a minimum area (in pixels), otherwise the algorithms will not find enough distinctive information to track it. Another issue is the change of the object size in the sequence, since all evaluated methods use a fixed size bounding box. When the object becomes smaller, the trackers start to track parts of the background. When it becomes larger, only the inner part of the object is tracked. The trackers in this thesis are built for single object tracking without handling the initialization problem, so videos with multiple objects entering and leaving the scene cannot be employed. Using only fixed camera tracking videos does not make sense since the actual strength of the methods cannot be demonstrated with such a sequence.

4.1.3 Videos

For the evaluation we use the videos provided by David Ross et al. [RLLY08][11], since they fulfill the requirements described above. The videos are also used in [GGB06b]. Unfortunately, no ground truth data is available for the videos, so we had to create it manually using Viper GT[12]. We also use two videos to demonstrate the method's capabilities to handle occlusions, namely the "Occluded Face" sequence by Adam et al.

[8] see `http://pets2007.net`, last visited 22. 11. 2009

[9] see `http://www.multitel.be/~va/selcat/index.html`, last visited 8. 12. 2009

[10] see `http://www.multitel.be/trictrac/?mod=3`, last visited 8. 12. 2009

[11] The videos are freely available at `http://www.cs.toronto.edu/~dross/ivt/`, last visited 15. November 2009

[12] see `http://viper-toolkit.sourceforge.net/docs/gt/`, last visited 15. November 2009

[ARS06] and the "Occluded Face 2" sequence by Babenko et al. [BYB09], both available on Boris Babenko's homepage[13]. See figure 4.2 for sample screenshots of all the videos.

Figure 4.2: Samples from Dudek, Ming-Hsuan, Sylvester, Occluded Face and Occluded Face 2 video datasets

Dudek In this face tracking sequence a single person first sits on a chair and then moves around. The camera is nonstationary. There is a ground truth in the form of seven stable points which are marked in every frame. We converted these points into bounding boxes by calculating the centroid of the points and putting an axially parallel fixed size rectangle around it. The video has a resolution of 720×480 pixels with 8 bits depth (greyscale) and consists of 1169 frames.

Ming-Hsuan This is also a face tracking sequence with a single person moving back and forth so that the scale of the face varies. At the end of the sequence, the person moves out of sight and another person enters the scene. Thus, one can verify if the tracker can distinguish between multiple objects of the same class. This video is recorded with a stationary camera. Ground truth was created completely by hand. The video has a resolution of 320×240 pixels with eight bits depth (greyscale) and consists of 1804 frames.

Plush Toy aka Sylvester In this sequence, a plush toy which is moved around by hand should be tracked. The pose of the object changes during the video, so for some frames the object is shown from the side. This video is also recorded with a stationary camera

[13]see `http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml`, last visited 19. 04. 2010

and the ground truth had to be created by hand. It has a resolution of 320×240 pixels with eight bits depth (greyscale) and consists of 1344 frames. During our experiments, we found that this video was the greatest challenge for our task. The difficulty is due to the heavy appearance changes, caused by turning the object 90 ° around in the sequence.

Occluded Face This video, recorded with a nonstationary camera, shows a woman who covers her face with a magazine. It has a resolution of 352×288 pixels with eight bits depth (greyscale) and consists of 888 frames. We use the ground truth provided by Adam et al., which supplies a bounding box for every fifth frame. In order make use of it we linearly interpolated the bounding boxes for every frame.

Occluded Face 2 This face tracking sequence includes occlusions caused by a book moved in front of the head as well as changes in appearance when the man puts on a cap. He also turns the head 45 ° to the left. The video has a resolution of 320×240 pixels with eight bits depth (greyscale) and consists of 816 frames. It is recorded with a stationary camera. We use the ground truth provided by Adam et al. like with the "Occluded Face" sequence.

4.2 Evaluation

Having introduced our datasets, we want to start with the description of the evaluation itself. We conduct all our experiments with exactly the same algorithm parameters. There was no tweaking for specific sequences. The only input changed for the various sequences is the different initial bounding box. At first we compare the trackers, then investigate specific properties of the algorithms.

4.2.1 Comparison

The first evaluation step is to undertake a quantitative comparison of the interest point tracker [KK10]. It is compared to our implementation of the online boosting method [GGB06b] and to the MeanShift tracker [CM02]. We take the MeanShift implementation of Cuce at Bilkent University in Ankara[14], because of its simple usage and because it is parameter-free, which allows repeatable comparisons. The evaluation method itself is described in section 3.5. We compare the algorithm using five sequences, namely "Dudek", "Plush Toy", "Ming-Hsuan", "Occluded Face" and "Occluded Face 2".

[14] available at `http://www.cs.bilkent.edu.tr/~ismaila/MUSCLE/MSTracker.htm`, last visited 3. 12. 2009

Table 4.1 shows the results of this evaluation step. For MeanShift and the OBTracker applied to the "Ming" video, the result had to be adjusted (marked with a * in the table): The trackers have no handling of object loss, they keep tracking even when the object has left the scene. Therefore we do not consider the frames where no object is present. So, the results including object loss handling would be equal or worse compared to the result stated here.

	Dudek	Ming	Sylvester	Occ. Face	Occ. Face 2
IP	99.83	98.95	100.00	100.0	100.0
OB	91.40	*99.08	92.56	99.89	73.92
MeanShift	89.05	*83.09	83.93	93.68	48.95

Table 4.1: Result of Tracker Comparison, Tracking Rate in % as defined in section 3.5

The **IPTracker** did not reach 100 % in the "Ming" sequence because in the ground truth, the object is still marked while it is leaving the scene and therefore its just partly visible, but the tracker loses the object when parts of the bounding box are out of the image area. In the "Dudek" sequence, the object is not tracked for a total of two frames.

The **OBTracker** has problems in the "Dudek" video with the change of background as well as with short-time appearance change (hand in front of the face), while in the "Sylvester" sequence the main difficulties are due to the viewpoint changes of the object. In the "Occluded Face 2" sequence, the tracker adapts to the book which partially covers the object and thereafter the tracker no longer tracks the object in a stable way.

The tracking error of the **MeanShift** tracker that remains in the "Ming" video is due to bad positioning of the bounding box, as seen in figure 4.3. The error does not lead to a tracking loss. A few frames later the object is tracked correctly again. The same problems also occur in the "Dudek" and "Sylvester" sequence. In the "Dudek" sequence, the object is lost when the person walks around and consequently, the background changes. In the "Sylvester" sequence, the object is lost at the end when a strong appearance change occurs due to a change of the viewpoint. In the "Occluded Face 2" sequence, the object is first lost in frame 372 when the person turns their head to the left. Then it is recovered for a short period and lost again when the person puts on a cap. This shows the mean shift tracker's lack of adaptability to appearance changes.

4.2.2 Low Frame Rate

To examine the performance with LFR-Videos, we subsample the input videos to 5 FPS. Table 4.2 shows the results of this evaluation step. For the OnlineBoosting and MeanShift

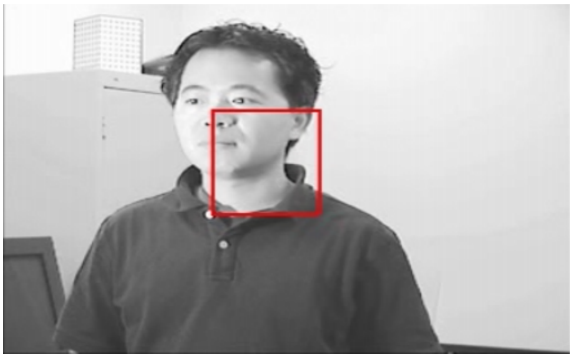


Figure 4.3: Tracking error of mean shift with the “Ming” dataset

algorithm with the “Ming” video, the result is once again adjusted (marked with a * in the table). The LFR situation does not make much difference to the **MeanShift** method, just

	Dudek	Ming	Sylvester	Occ. Face	Occ. Face 2
IP	82.29	81.06	40.63	100.00	99.26
OB	75.52	*99.60	97.32	100.00	60.74
MeanShift	73.96	*84.13	82.59	93.92	80.74

Table 4.2: Result of Tracker LFR Comparison, Tracking Rate in %

with the “Dudek” sequence, the object is lost for the last 30 frames. The reason for not being influenced by LFR is that the histogram features used with MeanShift are not that sensitive to appearance changes and tracking is still possible because of the smaller search space due to the mean shift procedure. Furthermore, the mean shift implementation does not update the initially calculated histogram. This has the advantage of not being at risk of drifting, but it cannot adapt to object appearance changes. Using online adaption would be precarious because of the imprecise placement of the bounding box, which would introduce a drifting error.

When using the **OBTracker** in the “Dudek“ sequence, a drifting problem starts along with the background change when the person walks around. There are 4 to 19 frames long phases where the object is not tracked and the bounding box just jumps around. The object is tracked incorrectly for a total of 47 frames. In the ”Occluded Face 2“ sequence, the one-shot learning seems to cause a problem, since the tracker is not able to recognize the object in the following frames with a sufficiently high confidence in order to allow learning. The tracker with the one-shot learned classifier is not able to track in a stable way.

As can be seen in the table, the **IPTracker** does not perform well with the LFR videos.

A detailed analysis of this problem shows that it is caused by the keypoint extraction method. We analyzed the number of matches within the object region from one frame to the next. While with the non-LFR videos the minimum of a 20-frame average is at least 11, in LFR situations it is 8.75 for the "Dudek", 9.35 for the "Ming" and only 2.4 for the "Sylvester" sequence, which explains the tracking loss after 40 % of the video. In this sequence, there are 18 frames without any match. In other terms, the object appearance changes too much so that the keypoint method cannot find matches anymore, not even from one frame to the next. The correlation between tracking quality and matching performance is evident. The tracker cannot compensate the problems of the underlying keypoint extractor. The results of this analysis can be seen in table 4.3.

In figure 4.4 frames 90 and 91 of the "Sylvester" sequence can be seen, five frames before the tracking loss occurs. While there are many new keypoints added to the set S_{fg} in the first frame (marked with blue crosses within the marked bounding box), there is only one keypoint in the latter frame which is classified positively (marked with a green circle within the bounding box).

	Dudek	Ming	Sylvester
Normal Average	24.88	23.54	19.64
Normal Minimum	11.10	15.75	11.55
LFR Average	17.96	18.11	7.91
LFR Minimum	8.75	9.35	2.40

Table 4.3: Matching Performance

Figure 4.4: IPTracker LFR tracking problem with "Sylvester" sequence

4.2.3 Minimum Area and Scale changes

Now we investigate the minimum object size in pixels which the **IPTracker** method requires. The main problem here is detecting a sufficient amount of interest points in the area. There is a simple theoretical limit, the algorithm needs at least one correctly positive detected interest point in each frame in order to be able to locate the object. As we have already done with the LFR tracking problem, we analyze the minimum area in terms of frame-to-frame matched interest points. We calculate one value for every frame of the sequence, which is the number of matched interest points. For this analysis, a meaningful criterion for matching quality is required. In order to plot it in 2D, we need a single value for the whole sequence for each scale. Once again, we use both the average matching count and the minimum of a 20-frame average. The idea behind using a 20-frame average, instead of the direct minimum, is to avoid a significant influence of single outliers.

We conduct this evaluation with the "Ming" sequence and use only the first 1514 frames, the period where the object is visible. As figure 4.5 shows, the behavior is almost linear down to a scale factor of around 0.5, which corresponds to an image size of 160×120 or an object size of 30×35 pixels. After this, the gradient becomes steeper. At this point, there is an average amount of 9.5 interest points and a minimum of 4.9, while at a scale of 0.6 the minimum comes to 8.3. Our analysis shows that tracking is still possible with a minimum value between 8 and 9, which corresponds to a scale of 0.6 or an object size of 36×43 pixels. Certainly other image sequences and especially other interest point detectors might lead to different results.

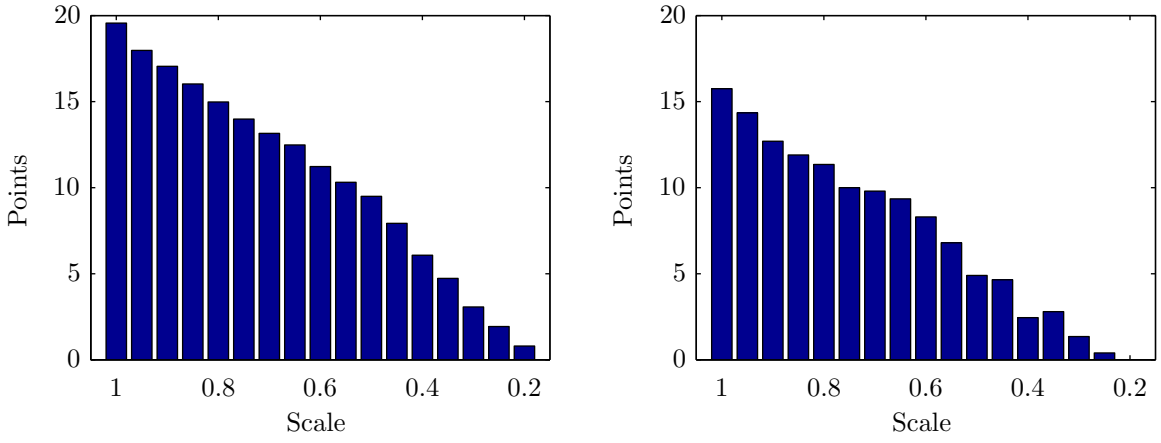


Figure 4.5: The average (left) and the 20-frame averaged minimum (right) number of matched interest points at various scales.

4.2.4 Video Quality

To analyze the influence of image quality on the **IPTracker**, we use a Gaussian kernel filter to blur the image. We vary the kernel size and use the same methodology we have used for the scale analysis, also with the "Ming" sequence. Of course there would be other options for decreasing image quality, such as adding noise. According to Sonka et al. [SHB99] Gaussian noise is a good approximation of noise which occurs in practical cases. It is achieved simply by adding a normally distributed zero-mean noise term to every pixel. Nevertheless, we use blurring for this experiment, since fast moving objects also lead to blurring, though this is only in the direction of motion, while the Gaussian blur is equal in all directions.

The results of our analysis, which can be seen in figure 4.6, show again a linear tendency. It becomes less steep at kernel size 15, as far as the average number of interest points is concerned. But the minimal number of interest points, which is the more important value, makes a leap at a kernel size of 13. At 11 there are 10.1 interest points while at 13 only 7.2 remain. Therefore, we assume that tracking is possible up to a kernel size of 11. Figure 4.7 shows blurred images with a Gaussian kernel size of 13 and 11.

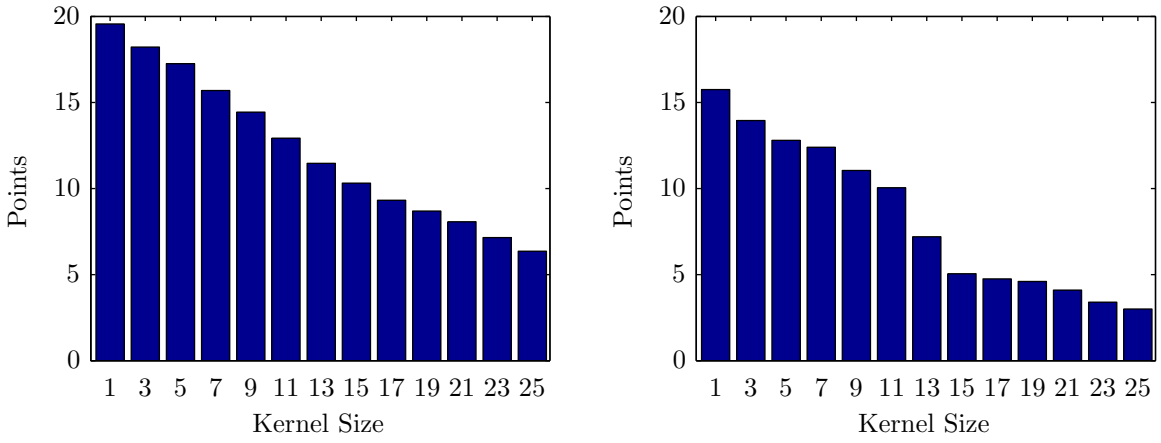


Figure 4.6: The average (left) and the 20-frame averaged minimum (right) number of matched interest points after a Gaussian blur with various kernel sizes.

4.2.5 Long Term Stability

The long term stability of the **IPTracker** is demonstrated by starting an image sequence in a loop, playing the video every second time in reverse to avoid jumps. This evaluation is carried out with both the "Dudek" and the "Ming" sequence.

Figure 4.7: Blurred images with a Gaussian kernel size of 11 (left) and 13 (right).

In the "Dudek" video, tracking is still stable after processing more than 37 000 frames, and thus more than 31 iterations of the sequence. At this point, we stopped the test manually. The "Ming" sequence shows the same behavior. The object is also tracked correctly for more than 37 000 frames, or 20 iterations. In this sequence, we have cut out the last frames where the object leaves the scene. When leaving it in, the object is lost after a few iterations. This is likely caused by drifting due to the background learned while the object leaves the scene and is therefore only partly visible. This experiment demonstrates that the tracker is stable as long as the object is tracked correctly. However, it reveals instabilities caused by tracking errors.

The drifting problem could be tackled, similarly to the methods in the literature [LAY+07, GLB08], with a hybrid approach. A second set of foreground points could be kept, which would not get adapted once the object representation is learned. Obviously this approach introduces adaptability limitations to the algorithm, so we performed no experiments in this direction yet.

4.2.6 Occlusions

We present the ability of all three methods to handle partial occlusions in two different ways. First of all, we use sequences including partial occlusions, namely the sequences "Occluded Face" and "Occluded Face 2". We have already stated the results on these sequences in figures 4.1 and 4.2, which show the trackers' capabilities of handling occlusions.

The **MeanShift** tracker has problems with the "Occluded Face 2" sequence which are not related to the occlusions, but to the appearance changes. The **OBTracker** tracker in the same sequence to the occluding object (book) in the normal sequence and has general tracking problems because of the one-shot learning in the LFR sequence.

For this evaluation, we additionally create an artificial occlusion by overlaying a black bar on 50 % of the object's bounding box (see figure 4.8) over a period of 20 frames in the three sequences without occlusions. The results are shown in table 4.4.

As you can see the occlusions have no influence on the tracking quality of the **IPTracker**. This is because of the interest point based concept, with the only requirement being that there are enough remaining interest points (at least one in each frame). Invisible object parts do not cause problems in locating the object. There is just a minor shift of less than five pixels, because the occluded points are removed from the calculation of the position average, which is based on the stored relative position of all detected foreground points.

To show that the tracker does not just adapt to the black box (which is actually easy to track), we conduct the same experiment with the IPTracker using a box filled with random color values. This experiment has the same result. However, using this method still does not prove that the tracker does not adapt to the box. Calculating the box features used in the SURF interest point detector, which are based on summing up image areas and subtracting them from each other, will have statistically the same result as with a black overlay. This is especially true for the larger scales. Fortunately, this does not hold for the whole feature descriptor and it is only statistically true for the detector. We could not undertake these experiments with the "Occluded Face" sequences, since multiple occlusions would occlude the whole object, and since our algorithms have no motion prediction component, tracking is impossible in this case.

The **OBTracker** yields worse results with the "Sylvester" sequence and similar results with the other two sequences, while the **MeanShift** tracker is not influenced by the artificial occlusion.

As both of our occlusion experiments show, the **IPTracker** handles the occlusions in our scenario perfectly. The experiment also identifies two possible problems resulting from occlusions. On the one hand, the object can be lost when it is not recognized while occluded, because the occlusion changes the appearance. On the other hand, in case the object is recognized, one must avoid adapting the occluding object in the learning algorithm. When this does not succeed, the tracker either tracks the occluding object and stops tracking the original one, or it becomes unstable and fails to track either object in a stable way. To avoid such a behavior, the same measures can be taken as with the drifting problem, for instance by keeping parts of the original object-representation to verify if it is still the same object.

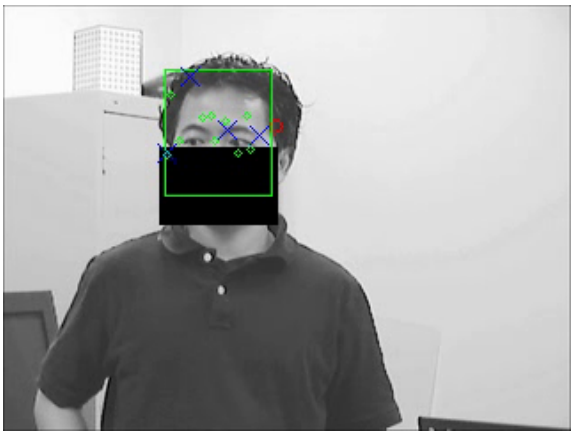


Figure 4.8: Sample screenshot from the occlusion sequence

	Dudek Occ	Ming Occ	Sylvester Occ
IP	99.83	99.06	100.00
OB	93.14	*94.98	66.89
MeanShift	89.05	*83.09	83.93

Table 4.4: Result of Tracker Occlusions Comparison, Tracking Rate in %

4.2.7 Speed

The **IPTracker** reaches a frame rate of 9 frames per second for the 320×240 videos running on a single CPU core with 2.6 GHz. The time is divided equally into keypoint extraction and tracking itself. For the 720×480 video, the algorithm runs with 2.2 frames per second, while the keypoint extraction step takes around two thirds of the time. This difference is easily explained by the fact that the keypoint extraction step depends on the image size, while the tracking step depends on the number of interest points. There are more interest points in a higher resolution video but the number of points which they have to be matched to is limited by the constants $S_{fg,max}$ and $S_{bg,max}$, in both cases.

Our implementation of the **OBTracker** reaches a frame rate of 1.1 frames per second for the 640×480 sequence. According to the authors, the original implementation, which is not freely available, is much faster with 20 frames per second at the same resolution [GGB06b]. This is justifiable since we have not made any efforts to optimize the tracker code yet.

4.2.8 Outlook

Besides our extensive evaluation, there are still open issues which should be tackled in the future.

Comparison to other techniques

It would be interesting to compare our online boosting implementation to the original one, which is unfortunately not freely available. Yuan Li et al. [LAY+07] use their own videos for evaluation which are not available online, otherwise we could compare our method to their LFR tracker. They utilized a scale error for their evaluation, which we do not calculate, since we keep the object scale fixed.

Ta et al. proposed an efficient SURF based tracking algorithm [TCGP09], which was published too late to be covered in this thesis. We could compare it to our method in terms of performance and speed and there are also optimizations, like computing descriptors only for certain detected points, which would be worth experimenting with.

A comparison to the recently published Multiple Instance Learning (MIL) Tracker [BYB09] would be interesting. The authors of this paper provide their results on various datasets, which would even allow for the usage of our own metrics with their tracker.

Other Detectors and Features

In this thesis, we evaluate neither the interest point detector nor the feature descriptor. Instead, we take SURF as given, since our focus is on the tracking part. There are two things that would be of particular interest: Does the choice make a significant difference, or do all algorithms perform equally in this application? If the latter is true, a simpler and faster algorithm could be used. In case there is a difference, it would be necessary to analyze what requirements a detector and feature descriptor have to fulfill to be suitable. One obvious requirement for the detector is that it finds enough keypoints to allow successful tracking. We could also automatically adapt the choice of the detector and its parametrization according to the current scenario.

As the evaluation shows, there are problems with the SURF method. It does not provide enough matching keypoints to allow successful tracking with our LFR sequences. Therefore, it would be very interesting to carry out an evaluation with various interest point detectors and feature extractors to find out, if there are methods which provide a significantly better matching performance with our LFR material.

Fixed Object Size

The methods currently allow only a fixed window size, so the trackers have to adapt their appearance representation partially to the background, when the object size decreases. With the IPTracker, an adaption of the bounding box would be possible, since every interest point has an associated scale. The distance between the interest points could also be used for the detection of the object's scale. The same is applicable to the object orientation too, because the interest points directly provide an orientation. The interest point locations relative to each other could be used to determine the object orientation as well.

4.3 Discussion

The evaluation demonstrates the properties of the investigated methods. The comparison between the methods shows the benefits of the IPTracker, but also the problems with low frame rate videos, which are related to the interest point extractor, are analyzed in detail. With the low frame rate setting, the online boosting method performs better, but it still has problems with one of the sequences. The trackers can cope with scale changes occurring in the sequences. A theoretical analysis of the IPTracker shows that a sufficient number of point matchings is reached down to a object size of 30×35 pixels; tracking is even possible with heavily blurred images. We experimented with partial occlusions, which are no problem for the IPTracker, while the other trackers produce minor errors in some of the experiments.

The goal of this thesis is to find a tracking method for nonstationary cameras which works even under challenging circumstances. We show in the evaluation that recognition based tracking methods are able to fulfill this goal. The IPTracker, which was developed, performs well in all experiments, with the exception of low frame rate sequences. We analyzed this problem and discovered how it can be tackled in the future.

Chapter 5

Conclusion

In this chapter, we want to conclude our work by giving a summary and describing additional insights which were gained. Furthermore, we give an outlook on possible future work.

Summary

In this thesis, we have analyzed the capabilities of object recognition techniques when applied to object tracking. The motivation behind such an approach is to allow for tracking under circumstances where other methods are not suitable. These include scenarios with abrupt motion, where the predictions of a motion model fail, or tracking with low frame rate videos. Such approaches are also applicable to sequences recorded with a nonstationary camera, because they do not rely on a background model. The basic idea behind such a "tracking by recognition" approach is to recognize the object in each tracking step, using the knowledge gathered in the preceding frames. In order to achieve this goal the object appearance has to be learned in the first frame, or additional prior knowledge about the object can be used. This learned model can then be improved during the tracking procedure with online learning. Fortunately, the expectations set for such an approach were fulfilled, we have been able to reach our goal of developing a tracking method working in challenging scenarios.

We started the thesis by explaining the underlying theoretical background. This includes object tracking itself, where we have defined the possible assumptions which can be made to render the task feasible and we have defined the parameters of our tracking task. Besides tracking itself, we have given an overview of object recognition. Since a representation of the object has to be found, this task has led to the field of machine learning, where we have explained the principles and challenges in this field and also the basic methodology. Object tracking implies a focus on online learning techniques, which

we have dealt with in more detail.

After the theory, we have continued with the description of a state of the art research on object tracking techniques, object recognition, machine learning and also combined techniques which have already been published. We have identified promising approaches and have concluded the chapter with the decision to develop a novel interest point based tracking technique and to evaluate this technique, as well as the state of the art methods online boosting [GGB06b] and mean shift tracking [CM02].

Our novel tracking algorithm, the **IPTracker** [KK10], is the main contribution of this thesis. In our implementation, we used the SURF detector and the SURF feature extractor [BTG06], though any method could be used, since the tracker just needs interest point locations and corresponding feature vectors. Sets of known points are created, allowing tracking based on point matching. The foreground and background points are stored separately and points are matched to either set, thus making the exclusion of background points possible. Every point in these sets has a weight or score, which reflects its quality and observation frequency. The set representation is updated online in every tracking step, where the matched points are updated, and points with a low score are replaced. We have also implemented tracking via **online boosting**, which follows a search window approach with Haar features and a boosting based online learning technique. With those trackers and a freely available implementation of the mean shift tracker we have carried out our experiments.

With these experiments, we have demonstrated the methods' capabilities as general-purpose tracking algorithms, as they are not specialized for any specific object type. There is also no need for learning, the only input required is a bounding box in the first frame. The evaluation consists of a general comparison of the trackers and the analysis of specific properties focusing on the IPTracker method. The IPTracker shows superior performance to the other methods in our experiments with five freely available sequences. When undertaking the experiments with low frame rate videos, the results were comparable, but significantly worse with one sequence, which shows one limitation of the technique. The problem originates from the interest point extractor which it is based on, since even frame-to-frame matching was not possible with the rapidly changing appearance in this sequence.

We also showed the long-term stability of the IPTracker with a video loop, as well as stability under partial occlusions. Moreover, we analyzed the influence of video quality and object size and showed the minimal requirements in quality and size.

LFR tracking requirements for an algorithm

LFR tracking requires the algorithm to adapt fast to changing conditions. The main problem, at least with challenging sequences, is not the fast movement of the object, but the rapid change of object appearance. While our algorithm is invariant under rapid movement, since no motion model is behind and therefore the position of the object within the image does not matter, its adaptability to sudden appearance changes is still limited. Our experiments showed that all three algorithms suffer from this limitation. Although this is a question of parametrization, the trade-off for fast adaptability is a higher risk of drifting – by adapting to errors.

A method needs to be based on a representation which is insensible to appearance changes, at least to a degree which allows recognition of the object from one frame to the next. Matching SURF features did not achieve this for all of our LFR sequences. A learning technique cannot compensate such problems. Another solution is to use additional assumptions, like MeanShift does, by requiring the object to be overlapping from one frame to the next to allow a local search. We did not want to do this in our thesis, since it leads to a loss of generality, but in practical applications it is commonly a good idea to use valid assumptions, in order to avoid unnecessary complication.

Future Work

For future research we plan to do an evaluation with various interest point detectors and feature extractors to find out if there are methods which provide a significantly better matching performance. Furthermore, we want to conduct experiments with deformable objects, like pedestrians, and cluttered background. It would be interesting to compare our methods to Tracking with Multiple Instance Learning, for which the authors provide both the code and their results on publicly available datasets.

In this thesis we have studied tracking methods which perform well even in challenging scenarios. The recognition based approach avoids many issues occurring with a traditional tracking technique, like the problems introduced with a background model or motion model.

Bibliography

[ARS06] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 798–805, 2006.

[ATJ01] S. Avidan, M.E.V. Technol, and I. Jerusalem. Support vector tracking. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 184–191, 2001.

[Avi07] S. Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, 29(2):261–271, 2007.

[BAS08] J. Begard, N. Allezard, and P. Sayd. Real-time human detection in urban scenes: Local descriptors and classifiers selection with adaboost-like algorithms. In *Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008.

[BBE+08] A. Baumann, M. Boltz, J. Ebling, et al. A review and comparison of measures for automatic video surveillance systems. In *EURASIP Journal on Image and Video Processing*, volume 2008, page 30. Hindawi Publishing Corporation, 2008.

[BETVG08] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110:346–359, 2008.

[Bis95] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[BJ98] M.J. Black and A.D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[BM98] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Workshop on Computational Learning Theory*, pages 92–100, 1998.

[BSR00] M. Bertalmío, G. Sapiro, and G. Randall. Morphing active contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):733–737, 2000.

[BTG06] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, volume 3951/2006, pages 404–417. Springer Berlin / Heidelberg, 2006.

[BYB09] B. Babenko, Ming-Hsuan Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 983–990, 2009.

[CHHB08] H. Celik, A. Hanjalic, E.A. Hendriks, and S. Boughorbel. Online training of object detectors from unlabeled surveillance video. In *Computer Vision and Pattern Recognition Workshops*, pages 1–7, 2008.

[CM02] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 603–619. IEEE Computer Society, 2002.

[Cro84] F.C. Crow. Summed-area tables for texture mapping. In *Conference on Computer Graphics and Interactive Techniques*, pages 207–212, 1984.

[CSMC08] G. Carrera, J. Savage, and W. Mayol-Cuevas. Robust feature descriptors for efficient vision-based tracking. In *Iberoamerican Congress on Pattern Recognition*, volume 4756/2008, pages 251–260, 2008.

[CV05] C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:273–197, 2005.

[CZT05] R. Collins, X. Zhou, and S.K. Teh. An open source tracking testbed and evaluation web site. In *IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, 2005.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893, 2005.

[FP03] D.A. Forsyth and J. Ponce. *Computer Vision: A modern approach*. Prentice Hall Professional Technical Reference, 2003.

[Fre95] Y. Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2):256–285, 1995.

[FS84] L.A. Ferrari and J. Sklansky. A fast recursive algorithm for binary-valued two-dimensional filters. *Computer Vision, Graphics, and Image Processing*, 26(3):292–302, 1984.

[FS97] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[FSA99] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[GB06] H. Grabner and H. Bischof. On-line boosting and vision. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 260–267, 2006.

[GGB06a] H. Grabner, M. Grabner, and H. Bischof. Fast approximated sift. In *Asian Conference on Computer Vision*, pages 918–927, 2006.

[GGB06b] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *British Machine Vision Conference*, volume 1, pages 47–56, 2006.

[GGLB09] M. Godec, H. Grabner, C. Leistner, and H. Bischof. Speeding up semi-supervised on-line boosting for tracking. In *33rd annual Workshop of the Austrian Association for Pattern Recognition*, pages 261–272, 2009.

[GLB08] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. *European Conference on Computer Vision*, 2(6):234–247, 2008.

[Gra08] H. Grabner. *On-Line Boosting and Vision*. PhD thesis, Graz University of Technology, Institute for Computer Graphics and Vision, Graz, Austria, 2008.

[HMS+07] S. Heymann, K. Maller, A. Smolic, B. Froehlich, and T. Wiegand. Sift implementation and optimization for general-purpose gpu. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 317–322, 2007.

[HS80] B.K.P. Horn and B.G. Schunck. Determining optical flow. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1980.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, pages 147–151, 1988.

[HYLL09] W. He, T. Yamashita, H. Lu, and S. Lao. Surf tracking. In *International Conference on Computer Vision*, pages 1586–1592, 2009.

[IB98] M. Isard and A. Blake. Condensation: conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[JAS05] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 695–700, 2005.

[Jav05] O. Javed. *Scene Monitoring with a Forest of Cooperative Sensors*. PhD thesis, University of Central Florida Orlando, Florida, 2005.

[JS08] O. Javed and M. Shah. *Automated Multi-Camera Surveillance: Algorithms and Practice*. Springer Science+Business Media, 2008.

[KCM04] J. Kang, I. Cohen, and G. Medioni. Object reacquisition using invariant appearance model. In *International Conference on Pattern Recognition*, volume 4, pages 759–762, 2004.

[Kit87] G. Kitagawa. Non-gaussian state-space modeling of nonstationary time series. *Journal of the American statistical association*, 82(400):1032–1041, 1987.

[KK10] W. Kloihofer and M. Kampel. Interest point based tracking. In *International Conference on Pattern Recognition*, 2010. Accepted.

[KMN+02] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[KS04] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 506–513, 2004.

[LAY+07] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different lifespans. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[Lin98] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30:79–116, 1998.

[LK81] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[Llo82] S.P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[Low99] D.G. Lowe. Object recognition from local scale-invariant features. *Seventh International Conference on Computer Vision*, 2(1):1150–1157, 1999.

[Low04] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[Mac67] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium*, volume 1, pages 281–197, 1967.

[Mor80] H.P. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. PhD thesis, Stanford University, 1980.

[MP43] W.S. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, 1943.

[MS05] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1615–1630, 2005.

[PP96] C. Papageorgiou and T. Poggio. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[PP99] C. Papageorgiou and T. Poggio. Trainable pedestrian detection. In *International Conference on Image Processing*, volume 4, pages 35–39, 1999.

[PT05] F. Porikli and O. Tuzel. Object tracking in low-frame-rate video. Technical Report TR2005-013, Mitsubishi Electric Research Laboratories, March 2005.

[RLLY08] D. Ross, J. Lim, R.S. Lin, and M.H. Yang. Incremental learning for robust visual tracking. In *International Journal of Computer Vision*, volume 77, pages 125–141. Springer, 2008.

[RN03] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach Second Edition*. Pearson Education, 2003.

[Ros62] R. Rosenblatt. *Principles of Neurodynamics*. Spartan Books, Washington, DC, 1962.

[Rot08] P.M. Roth. *On-line Conservative Learning*. PhD thesis, Graz University of Technology, Institute for Computer Graphics and Vision, Graz, Austria, 2008.

[SA04] K. Sato and JK Aggarwal. Temporal spatio-velocity transform and its application to tracking and interaction. *Computer Vision and Image Understanding*, 96(2):100–128, 2004.

[Sch90] R.E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[Seb02] F. Sebastiani. Machine learning in automated text categorization. In *ACM Computing Surveys*, volume 34, pages 1–47, 2002.

[SFBL97] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *International Conference on Machine Learning*, pages 322–330, 1997.

[SFPG06] S. Sinha, J.M. Frahm, M. Pollefeys, and Y. Genc. Gpu-based video feature tracking and matching. Technical Report TR 06-012, UNC Chapel Hill, Department of Computer Science, May 2006.

[SHB99] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision Second Edition*. PWS Publishing, 1999.

[Szu08] L. Szumilas. *Scale and Rotation Invariant Shape Matching*. PhD thesis, Vienna University of Technology, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, Vienna, Austria, 2008.

[TBZT07] F. Tang, S. Brennan, Q. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. In *International Conference on Computer Vision*, pages 1–8, 2007.

[TCGP09] Duy-Nguyen Ta, Wei-Chao Chen, N. Gelfand, and K. Pulli. Surftrac: Efficient tracking and continuous object recognition using local feature descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2937–2944, 2009.

[TM07] T. Tuytelaars and K. Mikolajczyk. Speeded-up robust features (surf). *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2007.

[TS92] D. Terzopoulos and R. Szeliski. Tracking with kalman snakes. In *Active vision*, pages 3–20, 1992.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.

[WB95] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR95-041, University of North Carolina at Chapel Hill, 1995.

[WCG05] J. Wang, X. Chen, and W. Gao. Online selecting discriminative tracking features using particle filter. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1037–1042, 2005.

[Web02] A. Webb. *Statistical Pattern Recognition Second Edition*. John Wiley and Sons, 2002.

[WKZL08] P. Wu, L. Kong, F. Zhao, and X. Li. Particle filter tracking based on color and sift features. In *International Conference on Audio, Language and Image Processing*, pages 932–937, 2008.

[WL07] C. Wang and J. Lien. Adaboost learning for human detection based on histograms of oriented gradients. In *Asian Conference on Computer Vision*, volume 4843/2007, pages 885–895, 2007.

[WZSN08] B. Wu, L. Zhang, V.K. Singh, and R. Nevatia. Robust object tracking based on detection with soft decision. In *IEEE Workshop on Motion and Video Computing*, pages 1–8, 2008.

[YJS06] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):1–45, 2006.

[YLS04] A. Yilmaz, X. Li, and M. Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536, 2004.

[ZAYC06] Q. Zhu, S. Avidan, M.C. Yeh, and K.T. Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1491–1498, 2006.

[ZLP04] D. Zhang, S. Z. Li, and D. Perez. Real-time face detection using boosting in hierarchical feature spaces. In *International Conference on Pattern Recognition*, volume 2, pages 411–414, 2004.

[ZYS09] H. Zhou, Y. Yuan, and C. Shi. Object tracking using sift features and mean shift. In *Computer Vision and Image Understanding*, volume 3, pages 345–352, 2009.