A hierarchical adaptive model for robust short-term visual tracking

A dissertation presented
by

Luka Čehovin

to
The Faculty of Computer and Information Science
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy
in the subject of
Computer and Information Science

Ljubljana, 2015

APPROVAL

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

— Luka Čehovin —

September 2015

The submission has been approved by

dr. Janez Demšar
Professor of Computer and Information Science
examiner

dr. Stanislav Kovačič
Professor of Electrical Engineering
examiner

dr. Jiri Matas
Professor of Computer and Information Science
external examiner
Czech Technical University

PREVIOUS PUBLICATION

I hereby declare that the research reported herein was previously published/submitted for publication in peer reviewed journals or publicly presented at the following occasions:

[1] L. Čehovin, M. Kristan, and A. Leonardis. Tracking Non-Rigid Objects by Combining Local and Global Visual Model. In A. Ion, W. G. Kropatsch, editors, *Proc. of CVWW 2009*, 2009.

[2] L. Čehovin, M. Kristan, and A. Leonardis. An adaptive coupled-layer visual model for robust visual tracking. In *Proc. of ICCV 2011*, pages 1363-1370, Barcelona, Spain, Nov, 2011. doi: 10.1109/ICCV.2011.6126390.

[3] L. Čehovin, M. Kristan, and A. Leonardis. Robust Visual Tracking using an Adaptive Coupled-layer Visual Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 233(2):199–220, 2013. IEEE Computer Society.

[4] L. Čehovin, M. Kristan, and A. Leonardis. Is my new tracker really better than yours?. In *IEEE Winter Conference on Applications of Computer Vision*, pages 540–547, Apr, 2014. IEEE Computer Society.

[5] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, et al.. The Visual Object Tracking VOT2013 challenge results. In *Proc. of ICCV 2013*, pages 98–111, Dec, 2013. IEEE Computer Society.

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Ljubljana.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Luka Čehovin
Hierarhični adaptivni model za robustno kratkoročno vizualno sledenje

POVZETEK

Vizualno sledenje je področje v okviru računalniškega vida, katerega rezultate je mogoče uporabiti na mnogih, tako novih kot tudi že uveljavljenih, področjih, kot so npr. robotika, video-nadzorni sistemi, interakcija med človekom in računalnikom, avtonomna vozila ter analiza športa. Glavno vprašanje vizualnega sledenja je razvoj algoritmov (sledilnikov), ki določajo stanja enega ali več objektov v toku slik ob upoštevanju časovne soslednosti le-teh. V tej doktorski disertaciji naslavljamo dve raziskovalni temi iz področja kratkoročnega vizualnega sledenja. Prvi sklop predstavljenih raziskav naslavlja konstrukcijo vizualnega modela, ki ga sledilnik uporablja za opis izgleda objekta. Vprašanje modeliranja ter osveževanje vizualnega modela je eno izmed ključnih vprašanj vizualnega sledenja. V okviru dela najprej predstavimo hierarhični vizualni model, ki izgled strukturira v več plasti. Najnižja plast vsebuje najbolj specifične informacije o izgledu, višje plasti pa opisujejo izgled v bolj posplošeni obliki. Hierarhična urejenost se odraža tudi v posodabljanju vizualnega modela, kjer višje plasti vodijo posodabljanje nižjih plasti, le-te pa v primeru lastne zanesljivosti služijo kot vir informacij za osveževanje višjih plasti. Koristi hierarhičnega modela sta predstavljeni z dvema implementacijama, ki sta primarno namenjeni sledenju netogih in artikuliranih objektov, kot tiste kategorije objektov, ki predstavlja velik problem za marsikateri vizualni sledilnik. Prvi predlagani model združuje lokalno in globalno predstavitev izgleda v sklopljenem vizualnem modelu. Spodnja plast je sestavljena iz več med seboj povezanih delov, ki so se sposobni prilagajati geometrijskim spremembam netogih objektov, zgornja plast pa vsebuje večmodalno globalno predstavitev izgleda, ki vodi proces posodobitve spodnje plasti. V okviru eksperimentalne analize smo pokazali, da se tak sklopljeni model izgleda izkaže v robustnosti, klub dejstvu, da smo za opis izgleda uporabili sorazmerno preproste opisnike. Analiza razkrije tudi nekaj pomanjkljivosti modela, ki se kažejo v znižani natančnosti sledenja. Zato naš drugi predstavljeni model razširja hierarhijo s tretjo plastjo in konceptom sidrnih predlog. Prvi dve plasti drugega vizualnega modela sta konceptualno zelo podobni osnovnemu sklopljenemu vizualnemu modelu, tretja plast pa vsebuje spominski sistem statičnih predlog, ki vizualnemu modelu nudijo močno informacijo o položaju in velikosti objekta v primeru dobrega ujemanja ene izmed predlog s sliko. Na ta način tretja plast pripomore k hitremu okrevanju celotnega vizualnega modela. Predstavljena eksperimentalna analiza koristi tretje plasti potrdi, saj sledilnik s tem modelom izgleda izboljša natančnost, pa tudi splošno kvaliteto sledenja.

Drugo vprašanje, ki ga naslavljamo v tej doktorski disertaciji, je ocenjevanje performans kartkoročnih sledilnikov. V nasprotju s prevladujočimi trendi v zadnjih desetletjih trdimo, da je vizualno sledenje kompleksen proces, katerega lastnosti ni mogoče opisati z eno samo mero uspešnosti, po drugi strani pa tudi ne smemo uporabiti poljubne množice mer, za katere ne poznamo medsebojnih odnosov. V naši raziskavi smo zato pregledali in analizirali pogosto uporabljene mere performans in pokazali, da nekatere izmed njih merijo iste kvalitete ali pa so celo teoretično ekvivalentne. Na temelju te analize smo predlagali par dveh šibko koreliranih mer, ki odražata natančnost in robustnost sledilnega algoritma, ustrezen prikaz takih rezultatov ter analizo celotne metodologije s pomočjo predlaganih teoretičnih sledilnikov, ki izražajo ekstremno obnašanje sledilnih algoritmov. Vse to smo nadgradili še z metodologijo rangiranja večjega števila sledilnikov, ki upošteva morebitno stohastično naravo sledilnikov ter preveri statistično značilnost razlike med njihovimi rezultati. Celotno metodologijo smo implementirali v odprtokodnem programskem orodju, razvili pa smo tudi preprost komunikacijski protokol, ki omogoča preprosto integracijo obstoječih implementacij sledilnikov v sistem. Z uporabo razvitega orodja se predlagana metodologija sedaj uporablja tudi v okviru *Visual Object Tracking (VOT) challenge* delavnic in tekmovanj.

Ključne besede: računalniški vid, vizualno sledenje, vizualni model, kratkoročno sledenje, artikulirani objekti, ne-togi objekti, mere performans, ocenjevanje performans, rangiranje

University *of Ljubljana*
Faculty *of Computer and Information Science*

Luka Čehovin
A hierarchical adaptive model for robust short-term visual tracking

ABSTRACT

Visual tracking is a topic in computer vision with applications in many emerging as well as established technological areas, such as robotics, video surveillance, human-computer interaction, autonomous vehicles, and sport analytics. The main question of visual tracking is how to design an algorithm (visual tracker) that determines the state of one or more objects in a stream of images by accounting for their sequential nature. In this doctoral thesis we address two important topics in single-target short-term visual tracking. The first topic is related to construction of an object appearance model for visual tracking. The modeling and updating of the appearance model is crucial for successful tracking. We introduce a hierarchical appearance model which structures object appearance in multiple layers. The bottom layer contains the most specific information and each higher layer models the appearance information in a more general way. The hierarchical relations are also reflected in the update process where the higher layers guide the lower layers in their update while the lower layers provide a source for adaptation to higher layers if their information is reliable. The benefits of hierarchical appearance models are demonstrated with two implementations, primarily designed to tackle tracking of non-rigid and articulated objects that present a challenge for many existing trackers. The first example of appearance model combines local and global visual information in a coupled-layer appearance model. The bottom layer contains a part-based appearance description that is able to adapt to the geometrical deformations of non-rigid targets and the top layer is a multi-modal global object appearance model that guides the model during object appearance changes. The experimental evaluation shows that the proposed coupled-layer appearance model excels in robustness despite the fact that is uses relatively simple appearance descriptors. Our evaluation also exposed several weaknesses that were reflected in a decreased accuracy. Our second presented appearance model extends the hierarchy by introducing the third layer and a concept of template anchors. The first two layers are conceptually similar to the original two-layer appearance model, while the third layer is a memory system that is composed of static templates that provide a strong spatial cue when one of the templates is matched to the image reliably, thus assisting in quick recovery of the entire appearance model. In the experimental evaluation we show that this addition indeed improves the accuracy, as well as the overall performance of a tracker.

The second question that we are addressing is the performance evaluation of single-target short-term visual tracking algorithms. In contrast to the dominant trend in the past decades, we claim that visual tracking is a complex process and that the performance of visual trackers cannot be reduced to a single performance measure, nor should it be described by an arbitrary set of measures where the relationship between measures is not well understood. In our research we investigate performance measures that are traditionally used in performance evaluation of single-target short-term visual trackers, through theoretical and empirical analysis, and show that some of them are measuring the same aspect of tracking performance. Based on our analysis we propose a pair of two weakly correlated measures to measure the accuracy and robustness of a tracker, propose a visualization of the results as well as the analysis of the entire methodology using the theoretical trackers that exhibit extreme tracking behaviors. This is followed by an extension of the methodology on ranking of multiple trackers where we also take into account the potentially stochastic nature of visual trackers and test the statistical significance of performance differences. To support the proposed evaluation methodology we have developed an open-source software tool that implements the methodology and a simple communication protocol that enables a straightforward integration of trackers. The proposed evaluation methodology and the evaluation system have been adopted by several Visual Object Tracking (VOT) challenges.

Key words: computer vision, visual tracking, visual model, short-term tracking, articulated object, non-rigid objects, performance measures, performance evaluation, ranking

ACKNOWLEDGEMENTS

There are several people that are at least partially responsible for the thesis that you are now reading. Firstly, I would like to thank my supervisor, professor Aleš Leonardis, for giving me a second chance in my academic career and supporting my ideas. I would like to thank my co-supervisor, assistant professor Matej Kristan, for all constructive discussions and assistance that focused and improved my research work. I would like to thank members of the doctoral committee who reviewed the thesis and provided valuable feedback. I am also grateful to the members of the Visual Cognitive Systems laboratory and other colleagues at the Faculty of Computer and Information Science for a wonderful and motivational research environment and the people behind VOT initiative for great past and future collaboration.

On a personal side I would like to thank my fiancée Jožica for all the love and support in the past years, my sister Urška for proofreading Slovenian parts of the text, and my family for putting up with me during my suspiciously long academic "walkabout".

— Luka Čehovin, Ljubljana, September 2015.

CONTENTS

Introduction

The thesis addresses the problem of *visual tracking*, one of the major research topics in computer vision. A novel model of appearance that is suited for tracking deformable objects is described and analyzed. We also propose a new methodology for performance evaluation of visual tracking algorithms. In Section 1.1 we define our research scope by describing the general premise of visual tracking and explore the various visual tracking scenarios along with their assumptions. After that, the core idea that is analyzed in this dissertation is summarized. We conclude the chapter by formally listing the research contributions that are presented in this thesis and providing an outline of the rest of the thesis.

1.1 Visual tracking

Visual tracking (sometimes also called *video tracking*) can be generally described as a process of determining the state, of a moving object, or multiple objects, in a sequence of images using the visual appearance information. An algorithm that performs visual tracking is called a *visual tracker*. Many high-tech applications in numerous emerging, as well as established, technological areas that rely on computer vision require the use of different types of visual tracking algorithms. For example:

- In *mobile robotics*, visual tracking is used by the robot to monitor the changes in its surroundings [1].
- Visual tracking is used in *video surveillance* to determine positions of monitored entities (e.g. people, cars, animals) [2, 3]. This includes applications such as activity detection [4, 5], pedestrian detection [6], and traffic accident detection [7].
- In *human-computer interaction* visual tracking is used to track entire body or hands of a person that is interacting with a computer system. This enables recognition of gestures in time [8–11] and interactive conferences [12].
- In *autonomous vehicles*, visual tracking is used to monitor the surroundings of the vehicle [13], enabling early prediction of events and timely reactions.
- In *sport analytics*, visual trackers are used to assist sports scientists to extract statistics from recordings of games or training sessions [14].
- In *multimedia systems* visual trackers are used in automatic video stabilization [15], modern video compression algorithms [16] and multimedia databases.

The research in visual tracking therefore deals with design and development of visual trackers. Visual trackers rely on different types of appearance information, such as color, edges, local intensity patterns, optical flow, etc. to predict the most likely state of an object in a next frame of the image sequence. In cases, where the exact structure properties of the object are not known in advance, the state of the object is typically defined as its location or occupied region in the image space. Such tracking is also known as *model-free* tracking indicating an absence of a more detailed instance-specific (geometrical) model of an object. The *model* in this case, i.e. the external representation of the object, should not be confused with the trackers internal appearance-based representation that is built during the tracking. This representation, on the other hand, is referred to as *appearance model* (sometimes also *visual model*), it is an internal representation of how the target object looks like or how does it differ from the surroundings. Because of the generality of such representations, visual tracking is widely applicable, but the generality also presents some very difficult problems, many of these still unsolved.

As the appearance of the object can change with time, an appearance model that is built at one point in time may not be suitable anymore. This leads to reduced tracking performance. If an appearance model is corrected with each new processed frame, the negative effects of an outdated model can be reduced. On the other hand, this leads to the phenomenon, known as *drifting*. Drifting occurs when an appearance model is updated with partially corrupted appearance information. The reason for this can be an error in state estimation, either because of violation of assumptions made when designing a visual tracker or because of ambiguous appearance information. These errors tend to propagate and slowly corrupt the appearance representation, leading the visual tracker to failure. Because the process is gradual, these kind of failures are hard to detect. This leads to two important questions that have to be answered when designing a robust appearance model:

1. How to represent the appearance of an object so that the tracker will be able to separate it from the background?

2. How to update this representation so that it will stay relevant with respect to the appearance of the object?

Both aforementioned questions are related - the formalization of the object appearance description determines the possible strategies for its updating, at the same time the choice of the update mechanism determines the robustness of the description. The appearance formalization can be restricted by assumptions, like rigidity of the object. These constraints also simplify the updating strategy and make the tracker computationally efficient as well as robust if the assumption is not violated[1]. Rigidity of the target is a popular assumption in visual tracking research that can be satisfied in many tracking scenarios. On the other hand, scenarios containing *non-rigid objects* that geometrically deform during tracking are also numerous and present an unsolved and challenging research problem.

Because of the large number of highly diverse application scenarios of visual tracking with their set of constraints, the research approaches in visual tracking became quite diverse. Many approaches assume constraints that make them very successful on but also limited to a specific tracking scenario. Other methods are less constrained and sacrifice top performance in a niche field for generality and wide applicability. To better motivate and position our own research work we review the most frequently used aspects of the visual tracking taxonomy that have been present in the research community for decades.

Single-target vs. multi-target tracking. Even though the single-target visual tracking (tracking a single object) can be seen as a special case of the multi-target visual tracking (tracking two or more objects) problem, the research on both topics has gone different ways. Single-target visual tracking research focuses on the accuracy of the visual tracker, its robustness and generality. The goal is to demonstrate the trackers performance on a wide range of challenging scenarios (various types of object, lighting conditions, camera motion, signal noise, etc.).

Multi-target tracking research tends to be more focused on specific applications. A common assumption of multi-target tracking systems is that the all targets are of single class which is known in advance (e.g. people or vehicle tracking for surveillance [17–19], animal groups tracking [20] or sports tracking [14], etc.). Because of this, multi-target tracking algorithms rely on appearance models with a lot of prior knowledge about the appearance of an object. The research is then focused on accurate detection and correct labelling of individual object's identities.

Short-term vs. long-term tracking. Having no direct connection to the actual time-span of the tracking task, the short-term versus long-term visual tracking separation deals with the re-detection of the target. In short-term tracking scenarios the target never disappears

[1]Many trackers consider only in-plane rigidity which means that even out-of-plane rotation of a rigid object violates the rigidity constraint.

from the image completely, therefore some visual association between the previous image and the following one is possible. In long-term tracking the target can completely disappear from the image for a long period of time, either because of an occlusion or because it moves outside of the image frame. Because of this, two important aspects of long-term visual trackers are detection of disappearance and a capability of fast re-detection of the object once it appears again in the image [21, 22].

On-line vs. off-line tracking. One of the constraints that influences the design of the visual tracking algorithms is also the availability of the data. In off-line tracking the entire sequence of images is available to the algorithm which can then arbitrarily access an image from the sequence [23]. In on-line visual tracking the sequence of images is a possibly infinite stream of data. The algorithm can only process an image as it becomes available and is able to retain only a limited number of past images in its memory for further use. In on-line tracking, a common implementation constraint that also influences the design of a tracking algorithm is requirement of real-time computational performance as many tracking scenarios require processing of live-acquisition video streams. Tracking scenarios, where off-line trackers can be applied (and have advantage over on-line trackers) include processing of finite length videos in multi-media collections or post-processing of a recorded video.

Single-camera vs. multi-camera tracking. Primarily, visual tracking involves analysis of a single video sequence, however, in some applications, like video surveillance and motion capture systems, multiple video sequences are available. The question that the research in multi-camera tracking algorithm is trying to answer is how to combine these sources of information to overcome their individual problems. In case, where the views of the individual video sequences are overlapping, this can be used to infer the position of an object in space and resolve ambiguities in case of occlusions [24]. Even if the video sequences are not overlapping, tracking an object in one sequence may give one enough information to remain tracking it in another [25].

Among a vibrant palette of tracking scenarios characterized by the four axes that we have described in the text above, our research work focuses on single-target short-term on-line single-camera tracking (since the last two attributes are less common than the first two we will frequently omit them in the rest of the text). As illustrated in Figure 1.1, a tracking algorithm of this kind receives an input in a form of a potentially infinite sequence of images and an initialization region and is able to provide a region for every image in the sequence. Since the sequence only contains short or partial occlusions of the target, the identity of the target can be unambiguously maintained throughout the sequence.

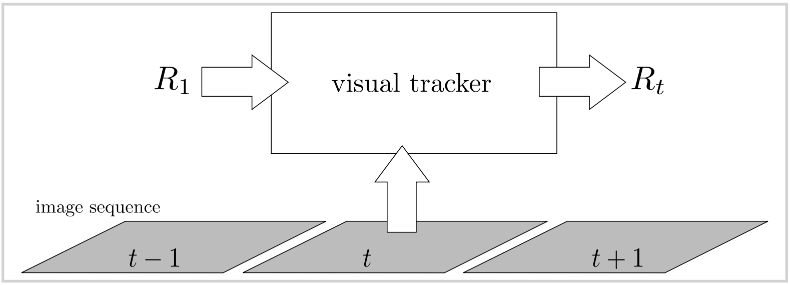


Figure 1.1
The input and output of a short-term single-target visual tracking algorithm.

The challenge in designing such algorithm is to maintain a representative appearance model of the object, without knowing detailed properties of the object in advance even in case if the object is articulated or non-rigid. Our research work involves design and evaluation of such appearance models. Firstly, we are developing new algorithms that are able to robustly predict the position of an object from the initialization region and without any other prior knowledge of the object. Secondly, in the context of this research we are also working on performance evaluation methodology, that is designed especially for single-target short-term visual tracking and highlights the performance aspects that are important in such scenarios.

1.2 Hierarchical models for visual tracking

In this doctoral thesis we present and investigate a new appearance model formalization, which we call a *hierarchical appearance model*. The concept was designed to provide a theoretical framework needed to address some of the problems with existing work that we highlight in the review of the related work in Chapter 2, most importantly non-rigidness of tracked objects with unknown geometrical properties as well as integration of higher-level appearance cues into the model. The idea of hierarchical appearance model for visual tracking is inspired by hierarchical models for object categorization, e.g, [26]. For object categorization the models are constructed so that they can accommodate a large degree of spatial variability among objects within the categories as well as (in some cases) variability between categories. Such models are trained off-line on a large set of annotated images.

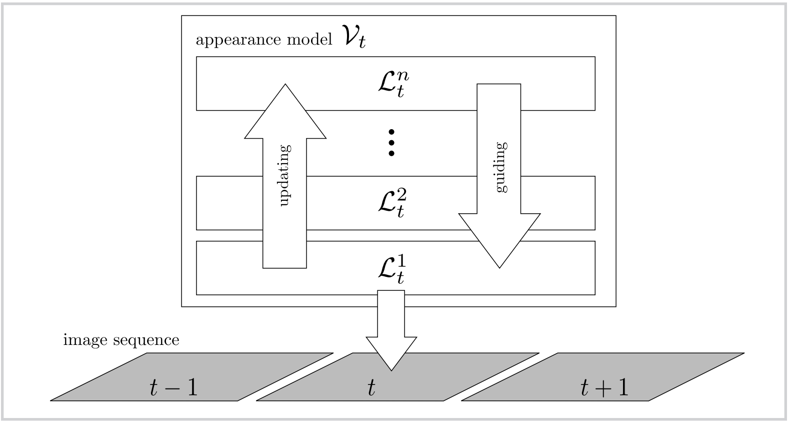


Figure 1.2

Illustration of hierarchical appearance model.

The motivation for using a hierarchy in case of visual tracking is to structure the appearance information of the object spatially as well as temporally. The task of visual tracking by modeling the appearance of an object without any prior information requires an appearance model that is specific enough to locate the object in the next frame, yet also flexible so that it can adapt to any appearance changes of the object. Because of this, the hierarchal structure has to be designed specifically for fast on-line adaptation and integration of multiple visual modalities that can be used together to track objects in various scenarios. Conceptually, a hierarchical appearance model, illustrated in Figure 1.2, is composed of a set of layers, each of them containing a different appearance representation of the object. The bottom layer contains the most specific information about the appearance at a given point in time. The higher layers contain information that is gradually more general and less time-dependent. The state of the appearance model at time-step t is specified by a set of layer states

$$\mathcal{V}_t = \{\mathcal{L}_t^1, \mathcal{L}_t^2, \ldots \mathcal{L}_t^n\}, \tag{1.1}$$

where the $\mathcal{L}_t^1$ denotes the first of the n layers. The function of specific layers is reflected in the update of the appearance model. The update process of the lower layer is performed using the information about the object that is provided by the higher layers, therefore this operation is also called *guiding*. The higher layers are updated using the information from the image and the lower layers, but only when that information is deemed reliable. When the information is not reliable, the updating does not take place and the higher layers are protected from appearance drift and can help the local layer recover. Consequently, an important aspect of the hierarchical architecture is also that the lower layers can perform tracking also without higher layers, however, such appearance model can be more prone to drift. High layers with their slow update rate therefore provide stability that guides the lower layers.

The hierarchical appearance model concept provides a flexible theoretical framework that can be used to better manage organization of appearance information in visual tracking. The bottom layer of a hierarchical appearance model is the layer that is closest to the current appearance of the object and has to constantly adapt to small changes that occur during a time-step. This can be achieved using visual descriptions with high number of free parameters, like geometrically constrained constellation of local parts, where each part has its own simple appearance model. Because of the high number of free parameters that ensure fine-grained flexibility, a constellation of parts is also prone to misrepresent the object on a global level. Therefore the bottom layer has to be guided by higher layers that provide more constraint information about the object that can be only reliably available sporadically. Guiding can be performed by removing outdated parts and adding new parts to the constellation to ensure a good coverage of the object.

Another beneficial aspect of hierarchical appearance model concept is its extensibility. Despite the interest of the research community in general-purpose visual tracking that can be immediately applied to an arbitrary object, real-world applications are rarely so vague. In many tracking scenarios the type of the object is known in advance, which means that some kind of appearance prior can be established. The nature of higher layers of the proposed appearance information hierarchy offers a good point for integration of this prior in a form of a pre-trained detector (e.g. person, face, car). If this representation is rich enough to ensure frequent detections for a specified domain of objects, the entire appearance model becomes more robust and resilient against drift as the sporadic detections keep the appearance model "in check".

1.3 Research contributions

In the proposed doctoral thesis we propose new algorithms and object appearance representations, suitable for tracking non-rigid and articulated objects. Additionally we propose a new methodology for visual tracker evaluation, either for a comparative evaluation or to gain additional insight into the behavior of a specific tracking algorithm. We therefore claim a two-fold contribution to the computer vision research field:

1. *Novel visual tracking algorithms that integrate local and global appearance information.* The proposed visual trackers integrate a novel hierarchical appearance model to track the target through its non-rigid deformations and other appearance changes. We analyze the behavior of the proposed tracker in various tracking scenarios as well as compare it to the state-of-the-art. We propose extensions to the basic appearance model that integrate prior object appearance knowledge about the object in an intuitive way while still enabling on-line adaptation during tracking.

2. *A new evaluation methodology for visual tracking.* We address two problems in evaluation of visual trackers: (i) we define a representative set of measures that can be used to capture different aspects of tracking performance by reviewing the measures that are used in visual tracking and , and (ii) we design a methodology to perform a large-scale comparative experiments. Our proposed evaluation methodology is also supported by a set of open-source software resources that enable easy development and integration of new tracking algorithms.

1.4 Thesis overview

The rest of the thesis is organized as follows. In Chapter 2 we review the related work from on visual trackers and the evaluation methodology. Following the review, we present our work on performance evaluation for visual tracking in Chapter 3 in order to establish means of performance evaluation that is used in the subsequent chapters. We review the core requirements for evaluation methodology and provide an overview of existing performance measures used in evaluation of visual trackers. We point out their advantages and disadvantages with respect to short-term tracking context. Based on experimental evaluation we propose a pair of measures that describe tracker performance better than a single measure. We propose an intuitive performance visualization and extend the performance evaluation methodology effort to comparison of multiple trackers using ranking considering statistical significance. The titular contribution of this thesis is presented in two parts, in Chapter 4 and Chapter 5, where we describe two implementations of the hierarchical appearance model, the Local-Global Tracking (LGT) tracker appearance model and the Anchor Templates (ANT) tracker, respectively. Both appearance models are evaluated in terms of parameter configuration and performance on many testing sequences that depict real-world applications of computer vision using the proposed evaluation methodology. We conclude the thesis work with a summary and general remarks in Chapter 6.

Related work

In this chapter we survey the recent research work in the field visual tracking, with the focus on two main topics. In Section 2.1 we look at on-line single-target visual trackers from the perspective of the appearance model that they are using to model the appearance of the target object. In Section 2.2 we review the recent efforts on visual tracking performance evaluation. In Section 2.3 we summarize the problems of existing work.

2.1 *Modeling appearance in visual tracking*

Appearance models (and therefore the visual trackers that are using these models) can be categorized based on the visual features that they use to describe the object as well as how they structure, store and update this appearance information. In a recent survey of appearance models for object tracking [27] the authors note that the appearance models can be separated in terms of the *level of detail* that they are using to model the appearance. As illustrated in Figure 2.1, one side of the spectrum is represented by *global* or *holistic* appearance models that coarsely model the appearance of the entire object. On the other side, *local* or *part-based* appearance models model the geometrical deformations explicitly and offer a detailed snap-shot of the object. Since our work involves fusion of both extremes we will review the popular approaches from both sides as well as those that try to combine them.

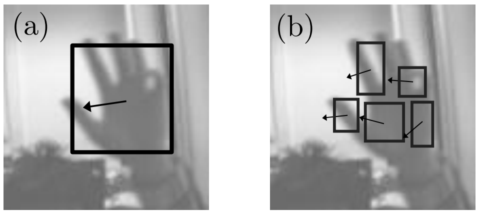


Figure 2.1

Illustration of global (a) and local (b) visual tracking concepts.

2.1.1 *Global appearance models*

Holistic appearance models maintain a global visual representation of the target appearance and have proven to be very successful in many tracking scenarios where the target does not deform significantly. This makes them a popular research topic that has been investigated in terms of many visual features and update mechanisms. Holistic appearance models differ in the type of image features that they use, how do they use these features to find the object and how they update the appearance model with new data.

Low level image features that are used to describe a target are image intensity [28–31], color [9, 32, 33], and inter-frame pixel changes (e.g. optical flow [34]). On a higher level these features can be stored as templates [15] that retain all the geometrical relations between pixels, or summarized as a probability distributions, e.g. using histograms [9, 32] or mixture-models [35] that discard relations between raw image features. Other higher-level descriptions include contours [36], texture [37], and gradients [38].

The problem of locating the tracked object is posed as searching the image for the region with the minimal distance to the appearance model. A popular approach of formalizing this problem is through optimization. Gradient methods, such as sequential kernel-based [9, 32] methods can be used if the distance function can be estimated efficiently and does not contain too many local extrema. In other cases stochastic methods, such as sequential Monte-Carlo optimization [33, 39] have proven to perform more favorable. Besides optimization, machine learning formulation has also become popular, both as a detection [40] and regression [41] problem.

After the target has been localized, the appearance model has to be updated to account for the appearance changes that may be small between two frames but sum up in the long run. The simplest way of updating the model is to replace it entirely, however, this makes the tracker very vulnerable to misaligned localizations. Gradual updating that combines previous information with the new one depends on the types of appearance model. Descriptions like histograms can be updated using autoregressive model. In case of detection or regression, the machine-learning is adapted to support on-line updates [40–42]. Another form of updating is feature selection that switches between multiple representations [43] or even switching between different appearance models [44].

The combinations of image features, localization, and updating techniques are numerous. We will therefore highlight several important research directions that have proven successful in the past decades:

Histogram tracking: Meanshift tracker, proposed by Comaniciu et al. [32] is one of the best known color-based trackers and uses a spatially weighted color histogram in RGB color space to determine a likelihood that a region contains an object using Bhattacharyya distance. The authors show that this can be solved efficiently using by looking for a mode of a distribution in every frame. Zhao et al. [45] use differential EMD (Earth's Mover distance) to match the target color distribution to the distribution of an image re-

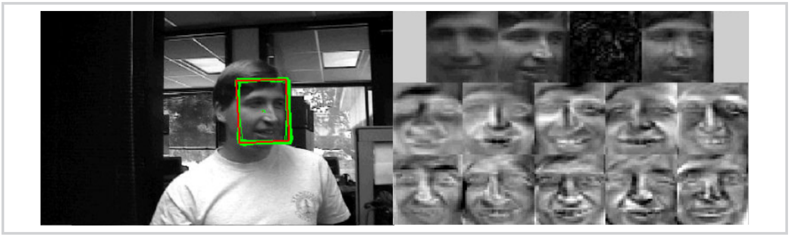


Figure 2.2

Tracking with image subspaces [48]. The current appearance model is an image subspace, represented by the eigenvectors visualized in the bottom-right side of the figure.

gion. Kernel tracking has also been used successfully used with gray-scale images where researchers optimize over sum-of-square-distances distance function for gray-scale histograms [46, 47]. The problem of modeling object with a histogram is a lack of spatial structure which makes the success of tracking dependent on the separability of foreground and background distributions.

Image subspace: As an interesting extension to template description, Ross et al. [48] proposed using a template subspace estimated during tracking using an on-line variant of principal component analysis (PCA). As illustrated in Figure 2.2, the subspace captures some of the deformations that occur during tracking thus allowing the tracker to account for them while remaining discriminative enough for tracking. The idea of tracking using image subspaces is to evaluate the re-projection error that occurs when a given region in the image is described with the subspace and then recovered again to the image form. As subspace projections tend to get computationally expensive for high-dimensional spaces, a particle filter approach is used to estimate the likelihood function fast enough for real-time tracking. As noted by Zhang et al. [49] the template subspace idea can be extended to random projections to reduce the size of the feature vector thus reducing the size of the subspace projection problem.

Classifiers: In the past two decades that were marked with stronger presence of machine learning methods in computer vision, *tracking by detection* [40, 42, 50] became increasingly popular concept in visual tracking. The core idea of tracking using detection methods is to use a discriminative classifier that determines if a given region contains the tracked object or not. This is repeated for a sub-set or all of the regions in the image and the detections are filtered into final prediction. One of the early successful tracking-by-detection approaches has been proposed by Grabner et al. [40], where a boosting

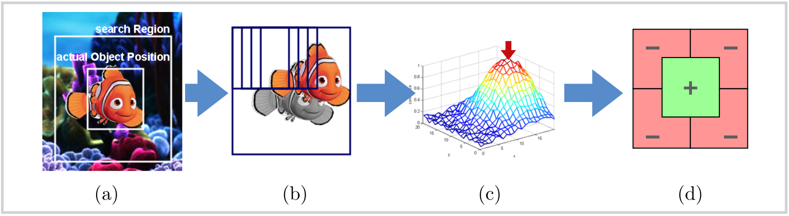


Figure 2.3

An illustration of online-boosting tracker idea according to [40].

cascade detector is used on a pool of Harr wavelet features [51]. The idea is illustrated in Figure 2.3. The tracker scans the local neighborhood of a previous position (Figure 2.3, a) using a sliding window approach (Figure 2.3, b). Based on the responses of the classifier (Figure 2.3, c) a new position of the object is determined and the classifier is updated (Figure 2.3, d) using new positive sample and negative samples from the neighborhood. This way the classifier is updated on-line to maintain a reliable discrimination between the foreground and the background.

This approach was later extended to semi-supervised [50] tracking where appearance information from the initialization stage is retained as a prior that constraints the learning process. Later the concept was also used by Babenko et al. [42] in multiple instance learning (MIL) framework where positive and negative samples are grouped into sets in a way that positive sets contain at least one positive sample and negative sets contain no positive samples. This way the learning algorithm implicitly selects the positive sample from a positive set during training.

The problems of tracing by detection approach is mainly grounded in the type of features that are used. Harr wavelet features, that are most commonly used because they can be efficiently computed in real-time, work well on textured areas, which means that the classifier focuses on textured subregions of the object, ignoring the uniform areas that could represent the majority of the object. The other problem are geometrical deformations of the target that can be accounted for only to some degree and are impossible to predict in the current design of such appearance models.

Soft assignment and regression: Tracking by detection relies on output of a classifier to locate the target - the position with maximum output score of a classifier is taken as a new position of the object. This hard assignment does not take into account the extent by which the training sample overlaps with the target location. Hare et al. [41] extended

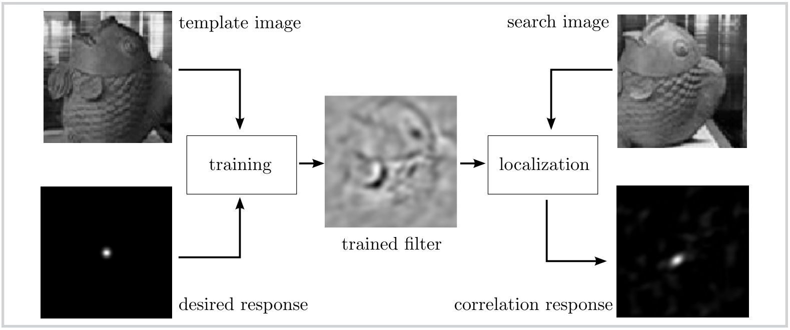


Figure 2.4

Tracking with correlation filters [53]. The idea is to obtain a filter that produces a response with a clear peak for a given template image.

an on-line structured support vector machine (SSVM) to largely alleviate the assignment problem. The SSVM method allows training with a continuous class assignment value and learns a discriminative regression from an image patch into displacement to estimate the target location. Another approach was proposed by Ellis et. al. [52], where offsets pathes are used to train a regression classifier that generates an offset of target center for a given patch in the image.

Recently, Bolme et al. [53] proposed tracking by training a filter whose output is highest at target location and diminishes away from the target. This results in a discriminativelly trained correlation filter for gray-scale images which is conceptually similar to soft-assignment training by [41]. The idea is illustrated in Figure 2.4. This approach was extended to multiple channels [38, 54] where the filter was trained on HoG features. Recently, Zhang et al. [55] adapted the correlation filters to spatio-temporal context learning.

Multiple visual trackers: One way to address the problems of individual holistic trackers is to use several different holistic trackers together to complement each other. Stenger et. al. [44] experimented with various combinations of different trackers, achieving better performance by switching between them as they are deemed more or less reliable. Santner et al. [56] proposed running an on-line discriminative tracker in parallel with motion prediction from a dense optical flow and NCC detection. Trackers, that do not address scale change, are connected into a cascade and only the model in the discriminative tracker is updated. Badrinarayanan et al. [57] proposed running in parallel two particle filters with different appearance models. The two trackers interact by influenc-

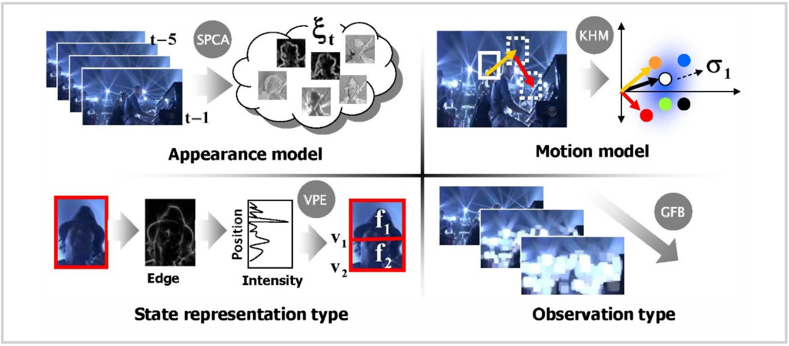


Figure 2.5

Tracking with multiple appearance models [59]. The tracker integrates a set of appearance models that are defined according to their appearance, state representation, and observation types, as well as multiple motion models.

ing re-sampling in each tracker. This approach was generalized by Kwon and Lee [58] who proposed a unified framework for a Monte-Carlo-based integration of several holistic appearance models in a recursive Bayes filter. Combining multiple visual trackers this way still means that one using holistic appearance models and is therefore not especially suited for tracking scenarios where the object is not rectangular and/or it deforms geometrically. Another problem is high computational complexity as multiple appearance models have to be generally matched at every frame.

Despite apparent success of global appearance models in certain tracking tasks, scenarios with rapid structural appearance changes pose a significant difficulty to these models. In holistic appearance models the entire appearance model is updated at once which increases the chance that the valid part of the visual information is corrupted by the new data. This can happen because the tracker fails to optimally predict the new position of the object, therefore updating the appearance model with the new data, that does not belong to the object, or because the tracker simply relies on the ambiguous data (e.g. features that do not separate object from the neighborhood). Another problem of global appearance models is the assumption that the object can be described as a rectangular image patch. While this is a reasonable simplification in many practical cases (e.g. tracking faces or faces from a single view point), there exist multiple scenarios where this is not true, e.g. non-rigid and articulated objects. All the geometrical deformations of the target that could be handled in a geometrical framework have to be accounted for with a holistic model update that can cause drifting.

2.1.2 Local appearance models

The key motivation of local appearance models is to explicitly handle spatial deformations that frequently occur in target appearance change. The general idea is that the appearance model is decomposed in separate elements, each describing a part of the object with its own (local) appearance model. These parts are can be connected together using some type of geometrical constraints. The actual types of appearance models of these parts, as well as the geometrical constraints used can vary from model to model.

Generally, the types of appearance models utilized in each part are simpler than in tracking with a single global appearance model because of increased computational complexity. Frequently used appearance models for parts are single pixels [60], histograms of local regions [61, 62], image patches [63] that can be used simply for optical-flow estimation [10, 64], as well as stable region descriptors [21, 65] and superpixels [66, 67]. Investigated geometrical constraints are are also diverse, ranging from simple flock heuristics [10]to sparsely [68] or densely [69] connected graphs and to rigid constellations [47, 70]. As with the review of global appearance models we will now highlight several important research directions in the past decade.

Optical flow: One of the early part-based trackers that used local optical-flow features was proposed by Kölsch and Turk [10]. It was inspired by simple bird flocking heuristics that were used to geometrically constrain the set of local features. A feature is removed from a set if it drifts too far away from the *flock* or it occludes another feature. The set is renewed by sampling new features given a color model of the object. The flock-of-features tracking approach was later extended by Hoey et al. [11] who integrated flocking constraints directly into particle filter optimization.

Tracking with a set of optical-flow trackers has been also used by Kalal et al. [22, 71] who made tracking more robust with introduction of a forward-backward error where the optical-flow estimate is computed forward and backward in time and is deemed reliable only if the two estimates are similar. Local optical flow estimates are fused via robust median estimator. Vojir et al. [64] replaced median estimator with RANSAC algorithm as illustrated in Figure 2.6.

Stable regions: Another approach that has been investigated for part-based tracking is detection and matching of stable regions. The key property of stable regions is that they can be reliably detected in multiple images of the same scene. Re-identification of these

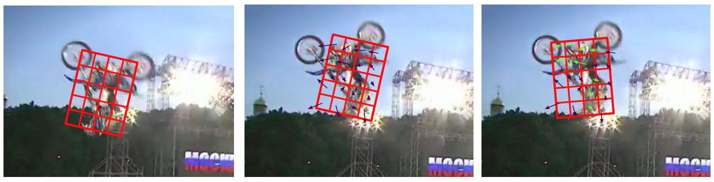

$t-1$: model state
t: optical flow features
t: robust transformation estimation

Figure 2.6

Tracking with optical flow [64]. A robust rigid transform is robustly estimated based on a weakly constrained set of optical flow trackers.

regions is done using one of numerous proposed descriptors. The general idea of tracking using stable regions is to remember the regions that appear on the object in the initial frame as well as their geometrical relations and use this appearance knowledge for detection of the object in the following frames.

Yin and Collins [63] detect feature points and enforce a single global affine transformation constraint to avoid drifting. Since generating ad-hoc reliable geometrical constraints between parts is a weakly defined problem for an unknown object, a relaxed method based on generalized Hough transform has been proposed by [72]. Zhou et al. [73] use a stable region based visual representation that combines this SIFT descriptor representation with the mean-shift. Specifically, SIFT features are used to find the correspondences between the regions of interest across frames. In parallel mean-shift procedure is used to locate the object by color similarity. Tracking is done by mutual support mechanism between SIFT features and the mean shift. The algorithm is sensitive to background clutter that leads to noisy SIFT matches and contradictory predictions by both components. Tang and Tao [65] construct a relational graph using SIFT-based attributes for object representation. The graph is based on the stable SIFT features which persistently appear in several consecutive frames. However, such stable SIFT features are unlikely to exist in complex situations such as shape deformation and illumination changes. Recently SIFT features were also used by Pernici et al. [21] in a more sophisticated tracking system that considers also features in the background for more robust detection as illustrated in Figure 2.7. Other feature descriptors have been proposed for tracking as well. Tran and Davis [74] propose a tracker that constructs a probabilistic pixel-wise occupancy map for each MSER feature detected that is then used for locating the object. Donoser and Bischof [75] improve the stability of MSER features for tracking by taking into account the temporal consistency across frames. He et al. [76] proposed a tracking algorithm using a SURF features that exhibit similar properties to SIFT features in terms

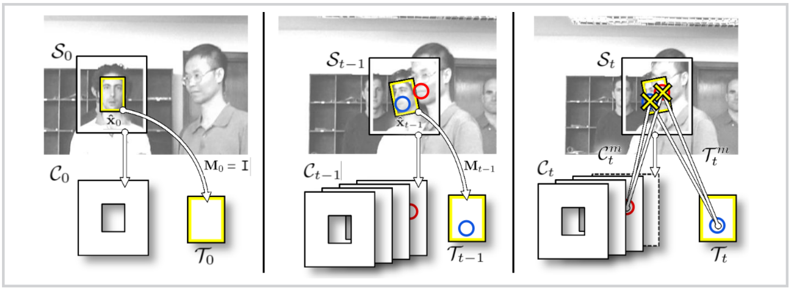


Figure 2.7

Tracking with stable regions according to [21]. The appearance model is represented by an object and context region (left). If similar stable features appears in both regions (middle) it is considered non-discriminative and is not used in estimation object state (right).

of repeatability, distinctiveness, and robustness, but are computed much faster.

Generally the number of stable regions is highly dependent on the appearance properties of the object (e.g. texture). The performance of a tracker depends on the availability and repeatability of stable regions and if the color of the object is homogeneous, no stable features will be found repeatably in the mentioned case and the tracking will fail. An interesting approach to track texture-less object has been recently presented [77] where pairs of edge features are robustly joined together into object position. While this approach works well even for objects without clear texture (e.g. sheet of paper), it does not handle well geometrical deformations of objects.

Superpixels: Superpixel algorithms group pixels into perceptually meaningful regions. The image, partitioned in superpixels is usually over-segmented, however, superpixels capture image redundancy and provide a convenient primitive concept from which to compute image features, that greatly reduces the complexity of subsequent image processing tasks. Superpixels have become popular in many computer vision tasks, most notably as object segmentation [78]. Recently, superpixels have also been used for visual tracking. Wang et al. [79] use superpixels to infer the position of the object using mean-shift clustering and classifier. As shown in Figure 2.8, an image is segmented into multiple superpixels, each of which corresponds to a local region. By building a local template dictionary based on the mean shift clustering, an object state is predicted by associating the superpixels of a candidate sample with the local templates in the dictionary. The visual size of the object must stay the same during tracking as this tracker relies on this information for robust tracking. Superpixels have also been used to track specific class of object with additional prior knowledge about the structure of the objects, e.g. ve-

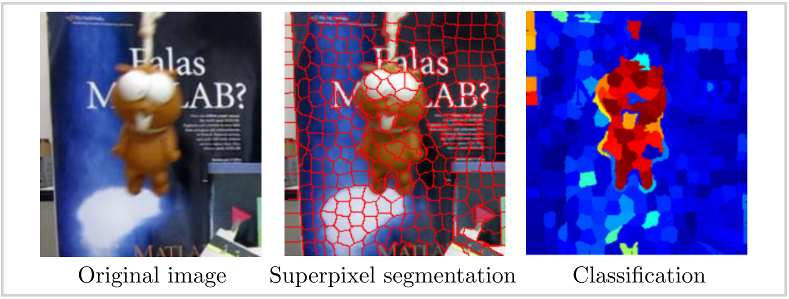


Figure 2.8

An illustration of superpixel tracking from [79].

hicles [80]. Recently Cai et al. [67] presented a graph matching approach for superpixel assignment that achieves competitive results in many challenging tracking scenarios, but is also computationally expensive.

In contrast to stable regions, superpixels always appear in consistent numbers, however, the segmentations are not guaranteed to be temporarily connected which makes the assignment problem hard, especially for multi-color objects or in presence of image blur.

Optimizing geometry: Considering the alignment of connected parts as an optimization problem, Fan et al. [47] proposed to track a target with a set of kernels which are connected by a global affine transformation constraint. This kind of assumption is violated in many visual tracking scenarios, but also significantly simplifies the optimization problem and makes it more robust and also more adaptable than using a single global kernel. The rigidity constraint was relaxed by Martinez and Binefa [61] that proposed a appearance model that connects multiple kernels together in triplets that are locally constrained with a local affine transform. The model therefore contains $N - 2$ constraints for N kernels as shown in Figure 2.9. This kind of structure is successful in addressing minor deformations, such as short out-of-plane rotations as well as handle occlusions of individual parts. On the other hand, the proposed appearance model lacks kernel-set update capabilities needed for longer tracking sessions with significant deformations. The other problem is tedious manual initialization as the performance of the tracker significantly depends on the initial positions of individual kernels.

A fully-connected graph of local appearance descriptors has been proposed Badrinarayanan et al. [69] for face tracking in combination with particle filter optimization.

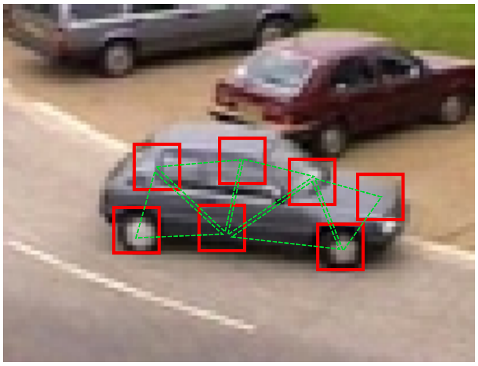


Figure 2.9

An illustration of multi-kernel tracking [61]. The red rectangles show positions of individual kernels, the green triangles show the connected triplets of kernels that are geometrically constraints.

In [81] Markov random fields are used to encode the spatial constraints between the parts. The problem of these approaches is that each part has to be manually initialized based on the target's structural properties. This is undesirable in many tracking scenarios. Furthermore, the set of parts is fixed, therefore the tracker cannot adapt to the larger deformations of the target.

Nejhum et al. [62] propose to track articulated objects by partitioning a segmentation mask in multiple parts using a greedy algorithm. These blocks are re-generated every frame from the updated segmentation that assumes approximately stationary foreground color. A more flexible geometrical constraints that allow long-term updating of part set during tracking have been presented by Kwon and Lee [68]. A simple star-shaped model is used to constrain individual parts that can be removed or added to the set. New parts are added to the appearance model using a global color model combined with a stable region detector, which means that the process fails for objects that lack textured surfaces. Another method that uses high-level global appearance representation to position new parts has been proposed by Godec et al. [72], where a segmentation algorithm is initialized using well-matching features and whose result is then used to sample new features. This approach is directly dependent on the robustness of segmentation which can be low in blurred or cluttered scenes. A simpler, but less reliable, segmentation has been proposed Duffner and Garcia [60], where each pixel is evaluated independently to form a segmentation. The success of these approaches shows the need for some high-level appearance information that would enable part-based trackers to extend the lifespan of the part-set in situations where the appearance of the object changes, e.g. significant deformations. However, the mechanism of the integration of local and global appearance remains poorly investigated and leaves a lot of room for improvement.

The main advantage of part-based appearance models is that they can adapt to geometrical deformations, which is especially useful when tracking non-rigid and articulated objects. On the other hand, part-based models also require estimation of a larger number of parameters compared to global appearance models. Finding the optimal combination of parameter values in this high-dimensional space can be difficult and can quickly result in update errors if such appearance model is updated in a self-supervised manner. Constraining the parameter space by enforcing constraints reduces this risk, but also reduces the adaptability of the model to geometrical deformations that violate the restrictions.

2.2 *Performance evaluation of visual trackers*

Visual tracking is one of the rapidly evolving fields of computer vision. Dozens of new tracking algorithms are presented and evaluated in scientific journals and at numerous research conferences every year. Evaluation of these new trackers and comparison to the state-of-the-art raises several questions:

1. Is there a standard set of sequences that we can use for the evaluation?
2. What kind of performance measures should we use?
3. Is there a standardized evaluation protocol that can compare trackers?
4. How to correctly interpret results to get new research insights?

Unlike some other fields of computer vision, like object detection and classification [82], optical-flow computation [83] and automatic segmentation [84], where widely adopted evaluation protocols are used, visual tracking is still largely lacking a consistent evaluation methodology. In this section we will review the most notable research efforts in this direction.

The absence of homogenization of the evaluation protocols makes it difficult to rigorously compare tracking algorithms across scientific publications and stands in the way of a faster development of the field. The authors of new trackers typically compare their work against a limited set of related algorithms due to the difficulty of adapting these for their own use in the experiments. One of the key issues here is the choice of tracker performance evaluation measures, which seems to be almost arbitrary in the tracking literature. Worse yet, an abundance of these measures are currently in use. Because of this, experiments in many cases offer a limited insight into the tracker performance, and prohibit comparison across different papers.

Until recently, the majority of methodological research that addressed performance evaluation in visual tracking was concerned with multi-target tracking scenarios. Smith et al. [85] explore the tracking characteristics important to measure in a real-life application, focusing on tracking scenario and consistent labeling of objects over time. Kasturi et al. [86] present a framework for evaluating object detection and tracking in video, focusing on face, text, and vehicle object categories, including an annotated dataset, performance metrics, and evaluation protocols. Black et al. [17] present a methodology for evaluating the performance of video surveillance tracking systems pseudo-synthetic video. Brown et al. [18] measure the performance of a surveillance system under several different scenarios. A well-known PETS workshop (e.g. [87]) has also been organized yearly for more than a decade with the main focus on performance evaluation of surveillance and activity recognition algorithms. Kao et al. [88] present a pair of information theoretic metrics with similar behavior to the Receiver Operating Characteristic (ROC) curves from signal detection theory on vehicles and people detection and tracking for surveillance applications. Carvalho et al. [89] propose to use different types of reference information and the combination of heterogeneous metrics to approximate the actual error of a tracker. Leichter and Krupka [90] investigate the theoretical properties of existing multi-target tracking performance measures with respect to two fundamental properties: monotonicity and error type differentiability.

One might view the multi-target tracking as a generalization of single-target tracking, however, as already mentioned in Section 1.1, there is a crucial difference in the objective which leads also to differences in evaluation. In multi-target tracking, the focus is is primarily a reliable registration of multiple same-class objects. This means that the measures are usually sensitive to correctness of target labeling assignments coupled with target detection and occlusion handling. In single-target tracking the focus is on a single object trajectory and the performance measures focus on the accuracy and robustness of the tracking algorithm on a wide range of tracing scenarios.

Recently, parts of research community became aware of the need for a more consistent evaluation and comparison approaches suited for single-target tracking algorithms.

This is noticeable in large-scale evaluation attempts as well as in research in the methodology of the evaluation. Wang et al. [66] compared several trackers using center error and overlap measures. Their research is focused primarily on investigating strengths and weaknesses of a fixed set of trackers. Wu et al. [91] authors perform an experimental comparison of several trackers. The performance measures in this case are chosen without theoretical consideration which results in a poor qualitative analysis of the results. Nawaz and Cavallaro [92] have presented a system for evaluation of video trackers that aims at addressing the real-world conditions in sequences. The system can simulate several real-world sources of noisy input, such as initialization noise, image noise and changes in the frame-rate. They have also proposed a new performance measure to address the trackers scoring under these simulated conditions. These recent experimental evaluations show the need for a better evaluation of visual trackers, however, none of them seems to address an important prerequisite for such evaluation, that is, what subset of the many available measures should be used for the evaluation. Frequently, multiple measures are used to cover multiple aspects of tracking performance without considering the fact that some measures describe the same aspects which leads to bias of the results. Instead, the selection should be grounded in a prior analysis of performance measures which is the main focus of this paper. Recently, Smeulders et al. [93] provided an experimental survey of several recent trackers together with an analysis of several performance measures. The interesting aspect of their general disposition and methodology is that they search for multiple measures that describe different aspects of tracking performance. On the other hand, their selection of measures is from the start biased in favor of detection-based tracking algorithms, which also affects their choice of final measures and the derived conclusions.

Finally, evaluation of tracker performance without ground-truth annotations has been investigated by Wu et al. [94], where the authors propose to use time-reversible nature of physical motion to evaluate trackers performance. As noted by SanMiguel et al. [95], this approach is not suitable for longer sequences. They propose to extend the approach using failure detection based on the uncertainty of the tracker. The problem is that the method has to be adapted to each tracker specifically and is useful only to estimate current performance of a tracker and not for comparative purposes.

An interesting approach to tracker comparison has also been recently proposed by Pang and Habin [96]. They aggregate existing experiments, published in various research paper, in a page-rank fashion to form a ranking of trackers. The authors acknowledge that their meta-analysis approach is not appropriate for ranking recently published trackers for which sufficient evaluation data is not yet available. Furthermore, their approach does not remove bias that comes from correlation in multiple performance measures.

The technical issue that arises in evaluation of visual trackers is how to perform the experiments objectively, in repeatable manner, on a large scale and without a lot of extra work from the researchers. Since the process of visual tracking evaluation, especially in case of on-line data availability constraint, tends to be more complex than that of classification or detection, where classical machine learning approaches can be used, the tools for visual tracking performance evaluation have to be specialized as well. Unfortunately, only few such semi-automatic evaluation frameworks were presented in the past and none of them gained enough popularity to become a de-facto standard. Most notable and general are the ODViS system [19] and the ViPER tooklit [97]. The first one is focused on design of surveillance systems, while the second one is a set of utilities for annotation and computation of different types of performance measures. Several other systems also exist (e.g. [91, 92]), however, they are all limited to a specific evaluation protocol and/or a small set of explicitly integrated trackers. Note that none of the systems above enable performing a fully automatic large-scale tracking experiment nor allowing third-party trackers to be integrated into this experiment easily by a researcher who is not familiar with the systems internal structure. These features are important for the widespread acceptance of a standardized evaluation system that would allow researchers to compare their trackers in a consistent manner.

2.3 Summary

The related research work that we have reviewed in Section 2.1 shows that visual tracking is a challenging research problem that can be approached from multiple angles. Because of this diversity it is impossible to encompass it with a single theoretical concept, even if we focus only on single-target tracking, as we have in this review. A lot of approaches can be characterized with the term holistic tracking, where the tracker used a global appearance model to maintain a global representation of the target appearance. These models have proven to be very successful in many tracking scenarios, however, they assume that the target does not deform significantly. On the other hand, part-based appearance models can easily adapt to geometrical deformations, but also require estimation of a larger number of parameters compared to global appearance models, which presents problems in case of self-supervised updating, meaning that the errors in parameter estimation are amplified with time. Our work on hierarchical appearance models is an attempt to bring together the best properties of both worlds, the flexibility of part-based models and the well-defined nature of global appearance models. This is partially addressed in Chapter 4 with the coupled-layer appearance model, but even more thoroughly in Chapter 5 with a appearance model that switches between both extremes during tracking.

As mentioned in Section 2.2, the methodology of performance evaluation in visual tracking has been a neglected part of the research in visual tracking for a long time. Only recently researchers have become aware of the problem on a larger scale which resulted in many uncoordinated attempts to propose new benchmarks and evaluation protocols. Despite this, none of them acknowledges the fact, that many performance measures that are still being used today suffer from serious design flaws or are being misused. In our work on evaluation, that is presented in next chapter, we primarily focus on this aspect of performance evaluation as we believe that a selection of well-behaved performance measures is the foundation for reducing the bias and increasing the interpretation power of the evaluation methodology.

3

Performance evaluation methodology

Due to algorithmic complexity and sequential nature of visual tracking, detailed evaluation of visual tracking algorithms is a non-trivial task that has only recently gained increasing attention of research community. Either for evaluation of research progress in sequential iterations of the same visual tracker or in comparison of a new tracker to the established state-of-the-art, the absence of a widely accepted evaluation protocol is slowing down the progress of the field due to the difficulty to rigorously compare trackers across publications. The key issues here are the choice of tracker performance evaluation measures and the choice of the dataset, which seems to be almost arbitrary in the tracking literature. Because of this, experiments in many cases offer a limited insight into the tracker performance, its strengths and, equally important, its weaknesses.

In this chapter we focus on the problem of performance evaluation in monocular short-term single-target visual tracking. We first discuss the requirements that can be expected of a comprehensive evaluation methodology in Section 3.1. We then set the foundation for a new performance evaluation methodology based on these requirements. Our proposed methodology is based on theoretical and empirical analysis of the set of most widely used performance measures that we describe in Section 3.2 and Section 3.3 respectively. We show that from a standpoint of tracker comparison, some performance measures are equivalent. Since some measures reflect certain aspect of tracking performance, combining those that address the same aspect introduces bias to the result. We identify a pair of complementary measures that are sensitive to two different aspects of trackers performance. We also demonstrate some side-products of our analysis, e.g., how the selected measure pair can be used to interpret sequence properties that makes the interpretation of the results easier. Our preliminary work on this topic has been published in [98, 99]. In Section 3.4 we then use the selected performance measures as a basis for a ranking methodology that can be used to compare multiple trackers. In Section 3.5 we introduce the evaluation system that implements the evaluation protocol and discuss the technical challenges that had to be taken into account during its design and development. We conclude the chapter in Section 3.6 with a summary and a brief description of the Visual Object Tracking Challenge [100–102] initiative that has been founded on the results of our research.

3.1 *Requirements for performance evaluation in visual tracking*

There are three main requirements that make up a good and comprehensive performance evaluation framework:

- Performance measures. An abundance of performance measures have been proposed for single-object tracker evaluation, however, there is no consensus on which should be preferred. Ideally, measures should clearly express certain aspects of tracking and should enable ranking of trackers. Apart from merely ranking, we also need to determine cases when two or more trackers are performing equally well due to their stochastic nature which results in different trajectories on each run on the same data. Good measures should allow easy interpretation and clear tracker comparison and support means of establishing a well defined tracker equivalence.

- Evaluation systems. For a rigorous evaluation, an evaluation system that performs the same experiment on different trackers using the same dataset is required. The wide-spread practice is to initialize the tracker in the first frame and let it run until the end of a sequence, which is a technically simple approach, however, the tracker might fail (drift from the target) right at the beginning of the sequence due to some visual degradation. For short-term tracking scenario this effectively means that the system utilized only the first few frames for evaluation of this tracker. A good system should fully use the data, which means that once the tracker fails, the system has to detect the failure and reinitialize the tracker. A certain level of interaction, that goes beyond simple running until the end of the sequence, is required in this case. Furthermore, the evaluation system has to also account for the fact that the trackers are typically coded in various programming languages and that researchers use various platforms for their research.

- Datasets. The dataset for evaluation of visual trackers includes a set of annotated video sequences. A good dataset should include video sequences with various properties like occlusion, clutter and illumination change. One approach is to construct a very large dataset, however, that does not guarantee diversity in visual attributes. A better approach is to annotate each sequence with the visual attributes occurring in that sequence. For example, a sequence is deemed to include occlusion globally even if the target is occluded only at a specific interval in the sequence. The trackers can then be compared only on the sequences corresponding to a particular attribute. However, many visual phenomena do not usually last throughout the entire sequence. For example, an occlusion might occur at the end of the sequence, while a tracker might fail due to some other effects occurring at the beginning of the sequence. In this case, the failure would be falsely attributed to occlusion. A per-frame dataset labeling is therefore required to facilitate a more precise analysis.

In this thesis we address the first two aforementioned requirements. The last requirement was addressed as a part of the Visual Object Tracking Challenge, in year 2013 [100] and additionally in year 2014 [101]. Since both of these challenges utilize our discoveries in terms of evaluation methodology as the theoretical foundation and adopted the evaluation system that we are presenting in Chapter 3.5 as the core software infrastructure component, we briefly review them at the end of this chapter.

3.2 A review of performance measures

As stated in the previous section, the first requirement for a good performance evaluation methodology is a good choice of performance measures. There are numerous performance measures that have become popular and are widely used in the literature when evaluating short-term single-target trackers, however, none of them is a *de-facto* standard. Our premise is that, while all of these measures do exhibit the properties required for tracker comparison, they can also be misused in various ways. In this section we review these measures and point out their theoretical issues.

All established performance measures for short-term single-object visual tracking assume that manual annotations are given for a sequence. Therefore we first establish a general definition of an object state description in a sequence with length N as

$$\Lambda = \{(R_t, \mathbf{x}_t)\}_{t=1}^{N}, \tag{3.1}$$

where $\mathbf{x}_t \in \mathcal{R}^2$ denotes a center of the object and R_t denotes the region of the object at time t. In practice the region is usually described by a bounding box (that is most commonly axis-aligned), however, a more complex shape could be used for a more accurate description. An example of two single-frame annotations can be seen in Figure 3.1. In some cases the annotated center can be automatically derived as a centroid of the region, but for some articulated objects, the centroid does not correspond to $\mathbf{x}_t$ (right case in Figure 3.1), therefore the center has to be annotated separately.

With established definition of object state we can now define performance measures as functions that aim at summarizing the extent to which the tracker's predicted annotation Λ_T agrees with the ground truth annotation, i.e., Λ_G.

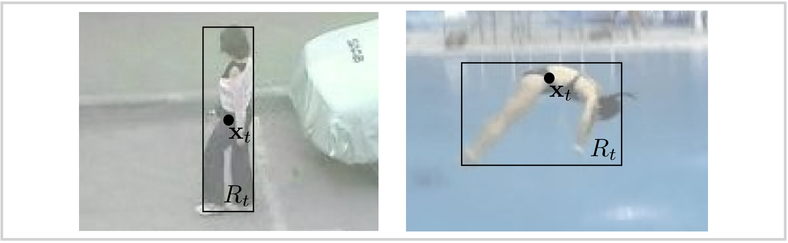


Figure 3.1

Two examples of an annotation for a single frame from the *woman* and the *driver* sequence. In the left example the center of the object can be estimated using the centroid of R_t, which is not true in the second case.

3.2.1 Center error

Perhaps the oldest means of measuring performance, which has its roots in aeronautics, is the center prediction error. It is still a popular measure in visual tracking [42, 48, 59, 70, 103, 104]. The main idea is to measure the difference between the target's predicted center from the tracker and the ground-truth center.

$$\Delta(\Lambda^G, \Lambda^T) = \{\delta_t\}_{t=1}^N, \quad \delta_t = \|\mathbf{x}_t^G - \mathbf{x}_t^T\|. \tag{3.2}$$

The popularity of center prediction measure comes from its minimal annotation effort, i.e., only a single point per frame. The results are usually shown in a plot, as in Figure 3.12 or summarized as average error (3.3), or root-mean-square-error (3.4):

$$\Delta_\mu(\Lambda^G, \Lambda^T) = \frac{1}{N}\sum_{t=1}^{N} \delta_t, \tag{3.3}$$

$$\mathrm{RMSE}(\Lambda^G, \Lambda^T) = \sqrt{\frac{1}{N}\sum_{t=1}^{N} \|\mathbf{x}_t^G - \mathbf{x}_t^T\|^2}. \tag{3.4}$$

The main drawback of this measure is its sensitivity to vague definition of the object's center. Because most trackers report center as a centroid of the region that they are tracking, this may not correspond to the actual center of the object that is provided in the groundtruth and results in systematic error. The center-error measure also completely ignores the target's size and does not reflect the apparent tracking failure [92]. To remedy this, a normalized center error $\widehat{\Delta}(\cdot, \cdot)$ is used instead, e.g. [30, 93], in which the center error at each frame is divided by the tacker-predicted visual size of the target, $size(R_t^G)$,

$$\widehat{\Delta}(\Lambda^G, \Lambda^T) = \left\{\widehat{\delta}_t\right\}_{t=1}^{N}, \quad \widehat{\delta}_t = \|\frac{\mathbf{x}_t^G - \mathbf{x}_t^T}{size(R_t^G)}\|. \tag{3.5}$$

Nevertheless, despite the normalization, this measure may give misleading results as the center error is reduced proportionally to the estimated target size which makes the measure much more sensitive for smaller targets than larger targets. Furthermore, when the tracker fails and is drifting over a background, the actual distance between the annotated and reported center, combined with the estimated size (which can be arbitrarily large) overpowers the averaged score which does not properly reflect the important information that the tracker has failed.

3.2.2 Region overlap

The normalization problem and other issues of center-error measures are rather well addressed by the overlap-based measures [49, 72, 93]. These measures require region annotations and are computed as an overlap between predicted target's region form the tracker and the ground-truth region:

$$\Phi(\Lambda^G, \Lambda^T) = \{\phi_t\}_{t=1}^{N}, \quad \phi_t = \frac{R_t^G \cap R_t^T}{R_t^G \cup R_t^T}. \tag{3.6}$$

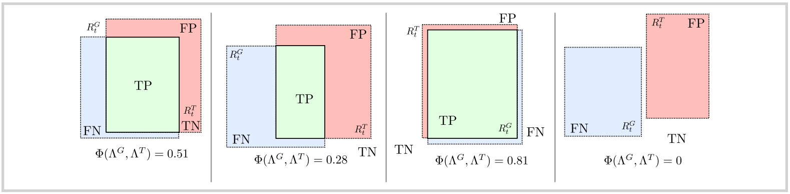

R_t^G
FP
TP
R_t^T
TN
FN
$\Phi(\Lambda^G, \Lambda^T) = 0.51$
R_t^G
FP
TP
FN
R_t^T
$\Phi(\Lambda^G, \Lambda^T) = 0.28$
TN
R_t^T
FP
TP
R_t^G
FN
TN
$\Phi(\Lambda^G, \Lambda^T) = 0.81$
R_t^T
FP
FN
R_t^G
TN
$\Phi(\Lambda^G, \Lambda^T) = 0$

Figure 3.2

An illustration of the overlap of ground-truth region with the predicted region for four different configurations.

An appealing property of region overlap measures is that they account for both position and size of the predicted and ground-truth bounding boxes simultaneously, and do not result in arbitrary large errors at tracking failures, as is the case on center-based error measures. In fact, once the tracker drifts to the background, the measure becomes zero, regardless of how far from the target the tracker is currently located. In terms of pixel classification (see Figure 3.2), the overlap can be interpreted as

$$\frac{R_t^G \cap R_t^T}{R_t^G \cup R_t^T} = \frac{TP}{TP + FN + FP}, \tag{3.7}$$

a formulation that is similar to the F-measure in information retrieval, which can be written as $F = \frac{2TP}{2TP+FN+FP}$. Another closely related measure, used in tracking to account for un-annotated object occlusions is precision [72], i.e. $\frac{TP}{TP+FP}$.

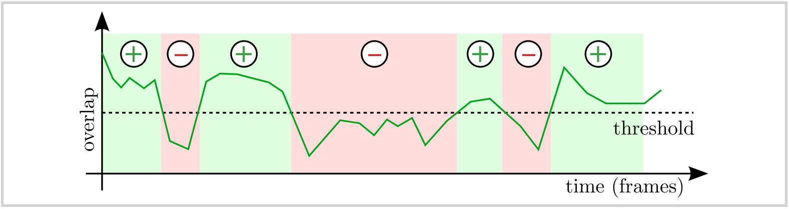


Figure 3.3

An illustration of overlap being used as a detection measure. The plus signs mark the intervals with positive detections (overlap above threshold), while minus signs mark the intervals with negative detections (interval below threshold).

The overlap measure is summarized over an entire sequence by an *average overlap* (e.g. in [91, 103]) that is defined as an average value of all region overlaps in the sequence

$$\bar{\phi} = \sum_t \frac{\phi_t}{N}. \tag{3.8}$$

Another measure based on region overlap is number of correctly tracked frames $N_\tau = \sum_t \| \{t|\phi_t > \tau\}_{t=1}^N \|$, where τ denotes a threshold on the overlap. This approach comes from the object detection community [82], where the overlap threshold for a correctly detected object is set to $\tau = 0.5$. The same threshold is often used for tracking performance evaluation, e.g. in [49] and [66], however, this number is too high for general purpose tracking evaluation. As seen in Figure 3.2 this threshold is reached even for visually well overlapping rectangles. This is especially problematic when considering non-rigid articulated objects.

To make the final score more comparable across a set of sequences of different lengths, the number of correctly tracked frames is divided by the total number of frames

$$P_\tau(\Lambda^G, \Lambda^T) = \frac{\| \{t|\phi_t > \tau\}_{t=1}^N \|}{N}, \tag{3.9}$$

where τ denotes the threshold of the overlap. The P_τ, also known as *percentage of correctly tracked frames*, is a frame-wise definition of the *true-positive* score, an interpretation that has become popular in tracking evaluation with the advent of tracking-by-detection concept. As noted in [93], the F-measure is another score that can be used in this context, however, it is worth noting that the detection based measures disregard the sequential nature of the tracking problem. As it is illustrated in Figure 3.3, these measures do not necessarily account for complete trajectory reconstruction which is an important aspect in many tracking applications.

The most popular measures for multi-target tracking performance, the Multiple Object Tracking Precision (MOTP) and Multiple Object Tracking Accuracy (MOTA) [86] can also be seen in the context of single-object short-term tracking as an extension of region overlap measures. MOTP measure is defined as average overlap over all objects on all frames, taking into account different number of objects that are visible at different frames, i.e.

$$MOTP = \frac{\sum_{i=1}^{M} \sum_{t=1}^{N} \phi_{i,t}}{\sum_{t=1}^{N} M_t}, \tag{3.10}$$

where M denotes the number of different objects in the entire sequence and M_t denotes the number of visible objects at frame t. In single-target short-term tracking $M = M_t = 1$, therefore MOTP can be simplified to an average overlap measure, defined in equation (3.8) earlier in this section. The MOTA measure, on the other hand, takes into account three components that account for accuracy of multiple-object tracking algorithm: number of misses, number of false alarms and number of identity switches, i. e.

$$MOTA = 1 - \frac{\sum_{t=1}^{N}(c_m MI_t + c_f FP_t + c_s SW_t)}{\sum_{t=1}^{N} N_t^G}, \tag{3.11}$$

where MI_t denotes the number of misses, FP_t denotes the number of wrong detections, SW_t denotes the number of identity switches, c_m, c_f, and c_s, are weighting constants and N_t^G denotes the number of annotated objects at time t. In single-target short-term tracking scenario there is only one object ($N_t^G = 1$, $SW_t = 0$) whose location can and should always be determined ($FP_t = 0$, $MI_t \in \{0, 1\}$), which means that the MOTA measure can be simplified to the percentage of correctly tracked frames, defined in equation (3.12) earlier in this section.

3.2.3 Tracking length

Another measure that has been used in the literature to compare trackers is *tracking length* [68, 104]. This measure reports the number of successfully tracked frames from tracker's initialization to its (first) failure. A failure criterion can be a manual visual inspection (e.g. [72]), which is biased and cannot be repeated reliably even by the same person. A better approach is to automate the failure criterion, e.g., by placing a threshold τ on the center or the overlap measure (see Figure 3.4). The choice of the criterion may impact the result of comparison. As the overlap based criterion is more robust with respect to size changes, we will from now on denote in the following the tracking length measure with an overlap-based failure criterion by L_τ .

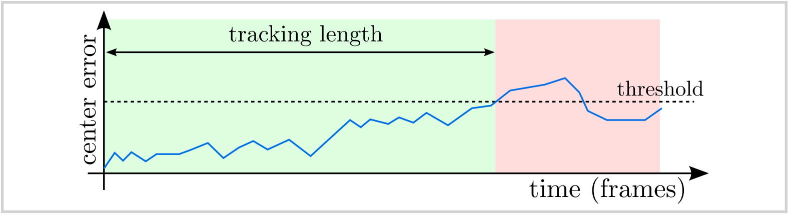


Figure 3.4

An illustration of the tracking length measure for center error.

While this measure explicitly addresses the tracker's failure cases, which the simple average center-error and overlap measures do not, it suffers from a significant drawback. Namely, it only uses the part of the video sequence up to the first tracking failure. If by some coincidence, the beginning of the video sequence contains a difficult tracking situation, or the target is not visible well, which results in a necessarily poor initialization, the tracker will fail, and the remainder of the sequence will be discarded. This means that, technically, one would require a significant amount of sequences exhibiting the various properties right at its beginning to get a good statistic on this performance measure.

3.2.4 Failure rate

A measure that largely addresses the problem of the tracking length measure is the so-called *failure rate measure* [20, 105]. The failure rate measure casts the tracking problem as a supervised system in which a human operator reinitializes the tracker once it fails. The number of required manual interventions per frame is recorded and used as a comparative score. The approach is illustrated in Figure 3.5. This measure also reflects the trackers performance in a real-world situation in which the human operator supervises the tracker and corrects its errors.

Compared to the tracking length measure, the failure rate approach has the advantage that the entire sequence is used in the evaluation process and decreases the importance of

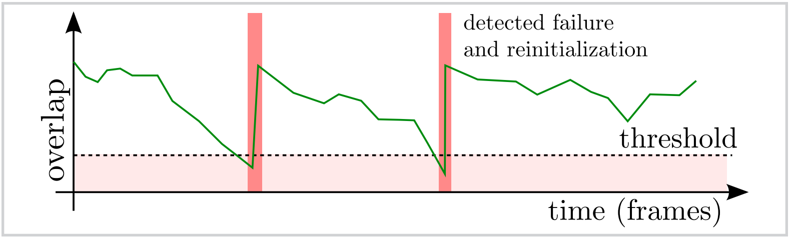


Figure 3.5

An illustration of the failure rate measure for overlap distance.

the beginning part of the sequence. The question of a failure criterion threshold is even more apparent here as each change in the criterion requires the entire experiment to be repeated. Researchers in [106, 107] consider a failure when the bounding box overlap is lower than 0.1. This lower threshold is reasonable for non-rigid objects, since these are often poorly described by the bounding-box area. An even lower threshold could be used for overlap-based failure criteria if we are interested only in the most apparent failures with no overlap between the regions. We will denote the failure rate measure with an overlap-based failure criterion with threshold τ as

$$F_\tau = |\mathcal{F}_\tau|, \quad \mathcal{F}_\tau = \{f_i\}, \tag{3.12}$$

where $\mathcal{F}_\tau$ denotes the set of all failure frame numbers f_i. A drawback of the failure rate is that it does not reflect the distribution of these failures across the sequence. A tracker may fail uniformly in approximately equal intervals or it may fail more frequently at certain events. We can analyze these different distributions by looking at the *fragmentation* of the trajectory that is caused by the failures. Using an information theoretic point of view [108], we define the following trajectory fragmentation indicator, $\mathrm{Fr}(\mathcal{F}_\tau)$,

$$\mathrm{Fr}(\mathcal{F}_\tau) = \frac{1}{\log F_\tau} \sum_{f_i \in \mathcal{F}_\tau} -\frac{\Delta f_i}{N} \log \frac{\Delta f_i}{N},$$

$$\Delta f_i = \begin{cases} f_{i+1} - f_i & \text{when } f_i < \max(\mathcal{F}_\tau) \\ f_1 + N - f_i & \text{when } f_i = \max(\mathcal{F}_\tau) \end{cases}, \tag{3.13}$$

where F denotes the number of failures and f_i denotes the position of the i-th failure. The special case for the last failure ensures that the resulting value is not distorted by the beginning and end of the sequence[1]. Fragmentation can only be computed when $|\mathcal{F}_\tau| > 1$ as we are observing the inter-failure intervals. Maximum value 1 is reached when the failures are uniformly distributed over the sequence and the value decreases when the inter-failure intervals become unevenly distributed. Note that the fragmentation can only be used as a supplementary indicator to the failure rate since it contains only limited information about the performance of a tracker, e.g. it will produce the same value for trackers that fail uniformly throughout the sequence no matter how many times they fail. However, it can be used to discriminate between trackers that fail frequently at a specific interval and those that fail uniformly over the entire sequence. As the evaluation datasets are getting larger, additional scores like fragmentation can help interpreting results on a higher level which we will demonstrate in Section 3.3.5.

3.2.5 Hybrid measures

Nawaz and Cavallaro [92] propose a threshold-independent overlap-based measure that combines the information based on tracking accuracy and tracking failure into a single score. This hybrid measure is called the *Combined Tracking Performance Score* (CoTPS) and is defined as a weighted sum of an accuracy score and a failure score. High score indicates poor tracking performance. The intuition behind CoTPS is illustrated in Figure 3.6. At a glance, an appealing property of this measure is that it orders trackers by accounting for two separate aspects of tracking. However, no justification, neither theoretical nor experimental, is given of such rather complicated fusion which makes interpretation of this measure rather difficult. It can be shown (see Appendix A.2) that the CoTPS measure can be reformulated in terms of average overlap, $\bar{\phi}$, and percentage of failure frames (where overlap is 0), λ_0, i.e.

$$CoTPS = 1 - \bar{\phi} - (1 - \lambda_0)\lambda_0. \tag{3.14}$$

The equation (3.14) conclusively states that two very different basic measures are being combined in a rather complicated manner, prohibiting a straightforward interpretation. Precisely, if one tracker is ranked higher than another one it is not clear if this is due to a higher average overlap or less failed frames. Furthermore, if equation (3.14) is reformulated as $CoTPS = (1-\lambda_0)(1-\hat{\phi}) + \lambda_0^2$, where the $\hat{\phi}$ denotes the average overlap on

[1]We interpret the sequence as a circular time-series and join the first and the last fragment. This way the value of Fr stays the same for the shifts of the same distribution of failures.

non-failure frames (where the overlap is greater than 0), multiple combinations of two values produce the same CoTPS score. In Figure 3.7 we illustrate several such equality classes, where the same CoTPS score is achieved using different combinations of the two components, which makes the interpretation of the results difficult. The combined score is also inconvenient in scenarios where a different combination of performance properties is desired.

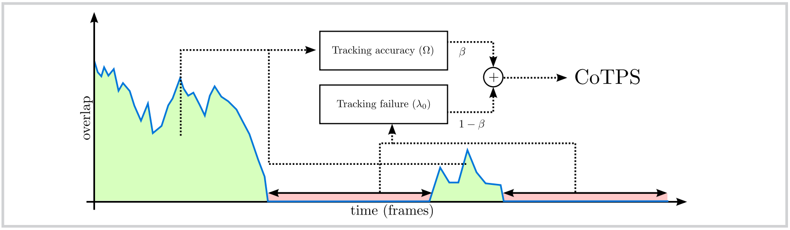


Figure 3.6
An illustration of the the CoTPS measure as described in [92].

In terms of performance score, we therefore believe that a better strategy is to focus on a few complementary performance measures with a well-defined meaning, and avoid fusing them into a single measure early on in the evaluation process.

3.2.6 Performance plots

Plots are frequently used to visualize the behavior of a tracker since they offer a clearer overview of performance when considering multiple trackers or sets of tracker parameters. The most widely-used plot is a center-error plot that shows the center-error with respect to the frame number [30, 42, 49, 70]. While this kind of plots can be useful for visualizing tracking result of a single tracker, a combined plot for multiple trackers is in many cases misused if applied without caution, because the tracker with an inferior performance diverts focus from the information that we are interested in with this type of plots, i.e. the tracker accuracy. An illustration of such a problematic plot is shown in Figure 3.8 where two trackers appear equal due to a distorted scale caused by the third tracker. A less popular but better bounded alternative approach is to plot region overlap, e.g. in [103].

In the previous section we have seen that a failure criterion plays a significant role in visual tracker performance evaluation. Choosing an appropriate value for the threshold may affect the order and can also be potentially misused to influence the results of a comparison. Generally it is better to avoid the use of a single specific threshold alto-

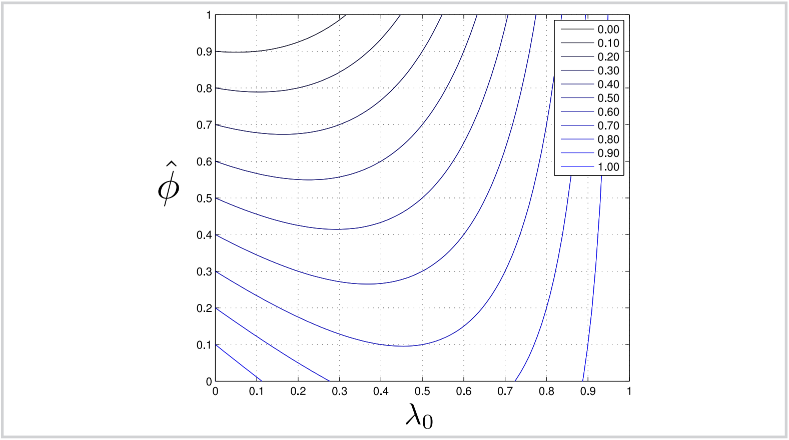


Figure 3.7

Equality classes for different values of CoTPS measure. Each line denotes pairs of average overlap on non-failed frames ($\hat{\phi}$) and percentage of failure frames (λ_0) that produce the same CoTPS score.

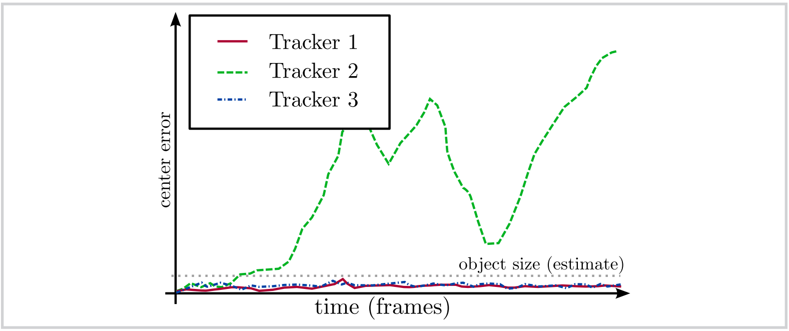


Figure 3.8

An example of center-error plot comparison for three trackers. Tracker 2 has clearly failed in the process, yet its large center errors cause the plot to expand its vertical scale, thus reducing the apparent differences of trackers 1 and 3.

gether, especially when the evaluation goal is general and a specific threshold is not a part of the target task. To avoid the choice of a specific threshold, results can be presented as a *measure-threshold* plot. This kind of plots have some resemblances to a ROC curve [109], like monotony, intuitive visual comparison, and a similar calculation algorithm. Measure-threshold plots were used in [42], where the authors used center-error as a measure as well as in [91], where both center-error and overlap are used.

The percentage of correctly tracked frames, defined in (3.12) as P_τ, is a good choice for a measure to be used in this scenario, although other measures could be used as well. The P_τ measure can be intuitively computed for multiple sequences which makes it useful

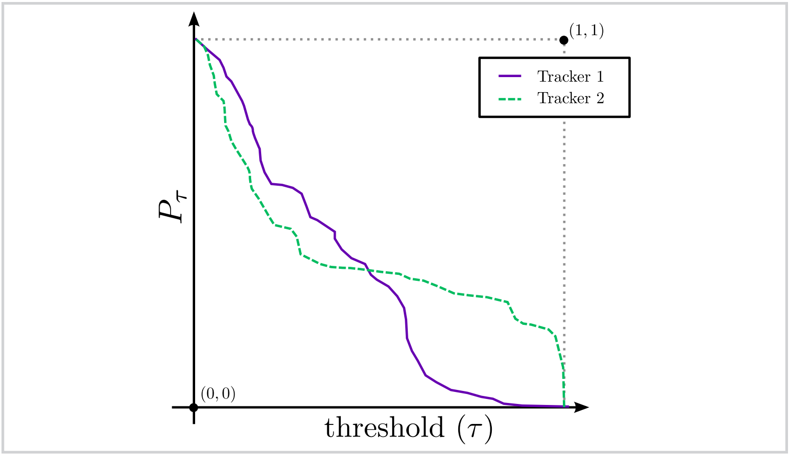


Figure 3.9

An illustration of the *measure-threshold* plot for two trackers. It is apparent that different values of the threshold would clearly yield different rankings for the trackers.

for summarizing the entire experiment (an example of P_τ plot is illustrated in Figure 3.9). Interpretations of such plots have been so far limited to their basic properties which in a way negates the information verbosity of a graphical representation. For example, similarly to ROC curves, we can compute an area-under-the-curve (AUC) summarization score, which is used in [91, 92] to reason about the performance of the trackers. However, the authors of [91, 92] do not provide an interpretation of this score. We prove in this paper (see Appendix A.1) that the AUC is in fact the average overlap, which results in two important implications: (1) the complicated computation of ROC-like curve and subsequent numerical integration for calculating AUC can be avoided by simple averaging of overlap over the sequence and (2) the AUC has a straight-forward interpretation.

A curve that is visually similar to P_τ plot is the *survival curve* [93]. In this case the curve summarizes the trackers' success (various performance measures can be used) over a dataset of sequences that are ordered from the best performance to the worst. While this approach gives a good overview of the overall success, it is not suitable for a sequence-wise comparison as the order of sequences differs from tracker to tracker. Not all sequences are equal in terms of difficulty as well as in terms of the phenomena that they contain (e.g. occlusion, illumination changes, blur) which makes it very hard to interpret the results of a survival curve on a more detailed level.

3.3 Experimental analysis of performance measures

The theoretical analysis so far shows that different measures may reflect different aspects of tracking performance, so it is impossible to simply say which the best measure is. Furthermore some measures are proven to be equal (e.g., area-under-the-curve and average overlap). We start our experimental analysis by establishing similarities and equivalence between various measures, by experimentally analyzing which measures produce consistently similar responses in tracker comparison.

In order to analyze the performance measures, we have conducted a comparative experiment. Our goal is to evaluate several existing trackers according to the selected measures on a number of typical visual tracking sequences. The selection of measures is based on our theoretical discussion in Section 3.2. We have selected the following measures:

1. average center error (Section 3.2.1),
2. average normalized center error (Section 3.2.1),
3. root-mean-square error (Section 3.2.1),
4. percent of correct frames for $\tau = 0.1$, $P_{0.1}$ (Section 3.2.2),
5. percent of correct frames for $\tau = 0.5$, $P_{0.5}$,
6. tracking length for threshold $\tau > 0.1$, $L_{0.1}$ (Section 3.2.3),
7. tracking length for threshold $\tau > 0.5$, $L_{0.5}$,
8. average overlap (Section 3.2.2),
9. hybrid CoTPS measure (Section 3.2.5),
10. average center error for F_0,
11. average normalized center error for F_0,
12. root-mean-square error for F_0,
13. percent of correct frames for $\tau = 0.1$, $P_{0.1}$ for F_0,
14. percent of correct frames for $\tau = 0.5$, $P_{0.5}$ for F_0,

15. average overlap in case of F_0,

16. failure rate F_0 (Section 3.2.4).

The first nine measures were calculated on trajectories where the tracker was initialized only at the beginning of the sequence, and the remaining seven measures were calculated on trajectories where the tracker was reinitialized if the overlap between predicted and ground-truth region became 0.

Since the goal of the experiment is not evaluation of trackers but selection of measures, the main guideline when selecting trackers for the experiment was to create a diverse set of tracking approaches that fail in different scenarios and are therefore capable of showing differences of evaluated measures on real tracking examples. We have selected a diverse set of 16 trackers, containing various detection-based trackers, holistic generative trackers, and part-based trackers, that were proposed in the recent years: A color-based particle filter (PF) [39], the On-line boosting tracker (OBT) [40], the Flock-of-features tracker (FOF) [10], the Basin-hopping Monte Carlo tracker (BHMC) [68], the Incremental visual tracker (IVT) [48], the Histograms-of-blocks tracker (BH) [62], the Multiple instance tracker (MIL) [42], the Fragment tracker (FRT) [70], the P-N tracker (TLD) [110], the Local-global tracker (LGT) [107], Hough tracker (HT) [72], the L1 Tracker Using Accelerated Proximal Gradient Approach (L1-APG) [30], the Compressive tracker (CT) [49], the Structured SVM tracker (STR) [41], the Kernelized Correlation Filter tracker (KCF) [38], and the Spatio-temporal Context tracker (STC) [55]. The source code of the trackers was provided by the authors and adapted to fit into our evaluation framework.

We have run the trackers on 25 different sequences, most of which are well-known in the visual tracking community, e.g. [48, 49, 66, 68, 70, 72, 106, 107], and several sequences were acquired additionally. Representative images from the sequences are shown in Figure 3.10. The sequences were annotated with an axis-aligned bounding-box region of the object (if the annotations were not already available), as well as the central point of the object, in cases where the center of the object did not match the center of the bounding-box region. To account for stochastic processes that are a part of many trackers, each tracker was executed on each sequence 30 times. The parameters for all trackers were set to their default values and kept constant during the experiment. A separate run was executed for the *failure rate* measure as the reinitialization influences other aspects of tracking performance.

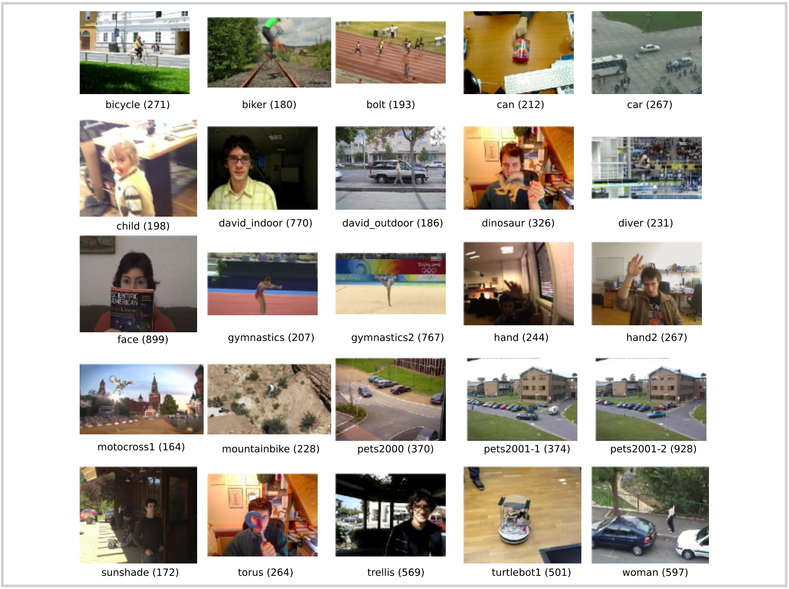


Figure 3.10

Overview of the sequences used in the experiment. The number in brackets besides the name denotes the length of a sequence in frames.

3.3.1 *The correlation analysis*

A correlation matrix was computed from all pairs of measures calculated over all tracker-sequence pairs. Note that we do not calculate the correlation on rankings to avoid handling situations where several trackers take the same place (if differences are not statistically significant). The rationale is that strongly correlated measure values will also produce similar order for trackers. Since we have run 16 trackers, each of the stochastic ones was run 30 times on every sequence, this means that every performance measure has about 10000 samples. This is more than enough for statistical evaluation of whether correlation across the measures exists. The obtained correlation matrix is shown in Figure 3.11. Using automatic cluster discovery by affinity propagation [111] we have determined five distinct clusters, one for measures 1 to 3, one for measures 4 to 9, one for measures 10 to 13, one for measures 14 and 15, and one for measure 16. All these correlations are highly statistically significant ($p < 0.001$).

The first cluster of measures consists of the three center-error-based measures. This is expected since all of these measures are based on *center-error* using different averaging

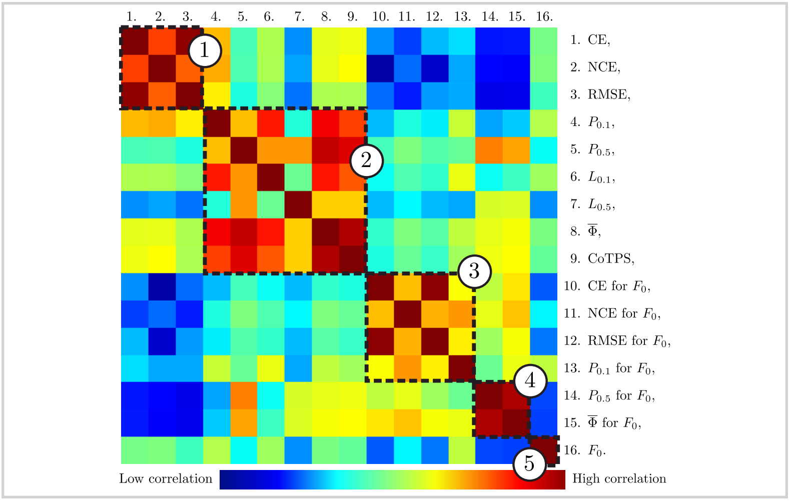


Figure 3.11

Correlation matrix for all measures visualized as a heat-map overlaid with obtained clusters. The image is best viewed in color.

methods. The second cluster of measures contains *average overlap*, *percentage of correctly tracked frames* for two threshold values ($P_{0.1}$ and $P_{0.5}$) and *tracking length* ($L_{0.1}$ and $L_{0.5}$). Measures in the second cluster assume that incorrectly tracked frames do not influence the final score based on the specific (incorrect) position of the tracker. Because of this and the insensitivity to the scale changes they are a better choice to measure tracking performance than the center-error-based measures. An illustration of this difference for *overlap* and *center-error* is shown as a graph in Figure 3.12, where we can clearly see that the center-error measure takes into account the exact center distance at frames after the failure has occurred, which depends on the movement of an already failed tracker and does not reflect its true performance.

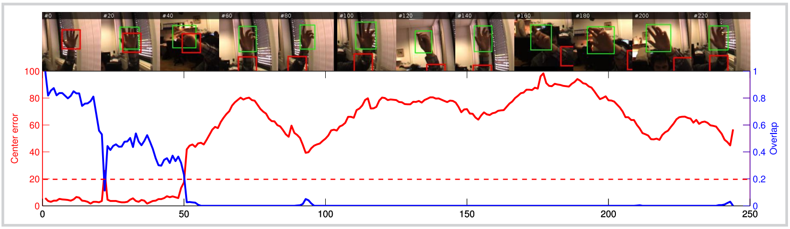


Figure 3.12

A comparison of *overlap* and *center error distance* measures for tracker CT on sequence *hand* [106]. The dashed line shows the estimated threshold above which the center error is greater than the size of the object. The tracker fails around frame 50.

The first cluster of measures in Figure 3.11 implies that the first three measures are equivalent and it does not matter which one is chosen. The second cluster requires further interpretation. Despite the apparent similarity of overlap-based measures 4 to 8 and of the CoTPS measure, the correlation is not perfect and the order of trackers differ in some cases. One example of such a difference can be seen for the TLD tracker on the *woman* sequence (Figure 3.13). We can see that the tracker loses the target early on in the sequence (during an occlusion), but manages to locate it again later because of its discriminative nature. The *average overlap* (Measure 8) and the *percentage of correct frames* (Measures 4 and 5) therefore order the tracker higher than the *tracking length* (Measures 6 and 7). On the general level we can also observe that the choice of a threshold can influence the outcome of the experiment. This can be observed for tracking length measures $L_{0.1}$ and $L_{0.5}$ and to some extent for the percentage of correct frames measures $P_{0.1}$ and $P_{0.5}$. In those cases, the scores for a higher threshold (0.5) result in a different order of trackers compared to the lower threshold (0.1). This means that care must be taken when choosing the thresholds as they may affect the outcome of the evaluation. While a certain threshold may be given for a specific application domain, it is best to avoid it in general performance evaluation. The last measure in the second cluster is the hybrid CoTPS measure [92] which turns out to be especially strongly correlated with the average overlap measure. By looking back at our theoretical analysis in Section 3.2.5 the CoTPS produces identical results for trajectories where the overlap never reaches 0 (no failure). In other cases the percent of failed frames, which can be approximated using $1 - P_{0.1}$, is also strongly correlated with average overlap. This means that the entire measure is biased towards only one aspect of tracking performance.

We can in fact observe a slight overlap between the first two clusters in the correlation matrix, implying similarity in their information content. Based on the above analysis and discussion in Section 3.2 we conclude that the *average overlap* measure is the most appropriate to be used in tracker comparison, as it is simple to compute, it is scale and threshold invariant, exploits the entire sequence, and it is easy to interpret. Note also that it is highly correlated with a more complex *percentage-of-correctly-tracked-frames* measure.

The *failure rate* measure influences the trackers' entire trajectory, because of the reinitializations. The data for measures 9 to 16 was therefore acquired as a separate experiment. The advantage of the supervised tracking scenario is that the entire sequence is used, which makes the results statistically significant at smaller number of sequences. It

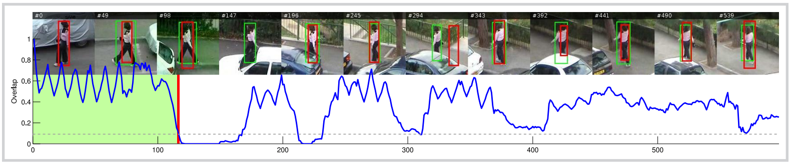


Figure 3.13

An overlap plot for tracker TLD on sequence *woman* [70]. The dashed line shows the threshold below which the tracking length detects failure (for threshold 0.1), which happens around frame 120.

does not matter that much if one tracker fails at the "difficult" beginning of the sequence, while the other one barely survives and then tracks the rest successfully. While supervised evaluation looks more complex, this is a technical issue that can be solved with standardization of evaluation process, for example using a standard communication protocol as the one that we present in Appendix B. In Figure 3.14 we can see the performance of the LGT tracker on the *bicycle* sequence. Because of a short partial occlusion near frame 175 the tracker fails, although it is clearly capable of tracking the rest of the sequence reliably if reinitialized. Measures that are computed on the trajectories with reinitialization exhibit similar correlation relations than for the trajectories without reinitialization.

According to the correlation analysis the least correlated measures are failure rate and average overlap on reinitialized trajectories. These findings are discussed in next section where we propose a conceptual framework for their joint interpretation. To further support the stability of the measurements, we have also performed the correlation analysis on different subsets of approximately half of the total 25 sequences and found that the these findings do not change.

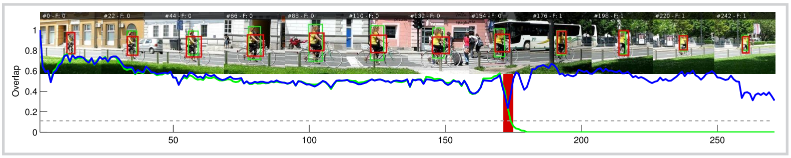


Figure 3.14

An overlap plot for tracker LGT on sequence *bicycle* [107]. The green plot shows the unsupervised overlap, and the blue plot shows the overlap for supervised tracking, where the failure is recorded and the tracker reinitialized.

3.3.2 *Accuracy vs. robustness*

An intuitive way to present tracker performance is in terms of accuracy (i.e., how accurately the tracker determines the position of the object) and robustness (i.e., how many times the tracker fails). Based on the correlation analysis in Section 3.3.1 we have selected a pair of evaluated measures that estimates the aforementioned qualities. The *average overlap* measure is the best choice for measuring the accuracy of a tracker because it takes into account the size of the object and does not require a threshold parameter. However, it does not tell us much about the robustness of the tracker, especially if the tracker fails early in the sequence. The *failure rate* measure, on the other hand, measures the number of the failures which can be interpreted as robustness of the tracker. According to correlation analysis in Section 3.3.1, if we measure average overlap on the reinitialized data, used to estimate failure rate, the measures are not correlated. This is a desired property as they should measure different aspects of tracker performance. We thus propose measuring the short-term tracking performance by the following A-R pair,

$$\text{A-R}(\Lambda^G, \Lambda^T) = \left(\overline{\Phi}(\Lambda^G, \Lambda^T), F_0(\Lambda^G, \Lambda^T)\right), \tag{3.15}$$

where $\overline{\Phi}$ denotes *average overlap* and F_0 denotes the *failure rate* for $\tau = 0$. Note that the value of failure threshold τ can influence the final results. If the value is set to a high value (i.e. close to 1) the tracker is restarted frequently even for small errors and the final score is hard to interpret. Based on our analysis, we propose to use the lowest theoretical threshold $\tau = 0$ to only measure complete failures where the regions have no overlap at all and a reinitialization is clearly justified. In theory, a tracker can also report an extremely large region as the position of the target and avoids failures, however, the accuracy will be very low in this case. This is an illustrative example of how the two measures complement each other in accurately describing the tracking performance.

It is worth noting that there are some parallels between the hybrid CoTPS measure [92], and the proposed A-R measure pair. In both cases two aspects of tracking performance are considered. The first part of the CoTPS measure is based on the AUC of the overlap plot, which, as we have shown, is equal to average overlap. The second part of the measure attempts to report tracker failure by measuring the number of frames where the tracker has failed (overlap is 0), which could also be written as P_0. Despite these apparent similarities, the A-R measure pair is better suited for visual tracker evaluation for several reasons: (1) the chosen measures are not correlated, (2) the supervised evaluation protocol uses sequences more effectively because of reinitializations, (3) different performance profiles for average overlap and failure rate produce different combinations of scores that can be interpreted, which is not true for CoTPS measure.

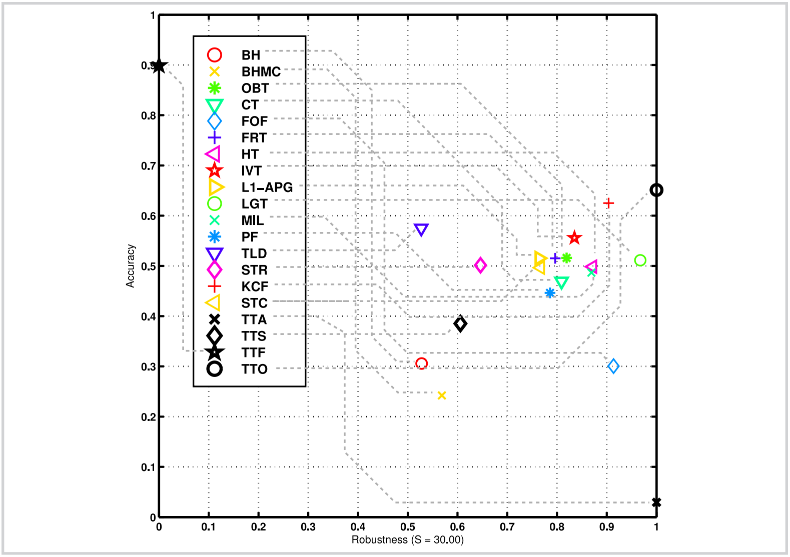


Figure 3.15

An accuracy-reliability data visualization for all trackers over all sequences.

A pair of measures is most efficiently represented via visualization. We propose to visualize the A-R pair as a 2-D scatter plot. This kind of visualization is indeed very simple, but is easy to interpret and has been used in visual tracking visualization before, e.g. [44]. An example of an A-R plot for the data from the experiment can be seen in Figure 3.15, where we show the average scores for all sequences, from which one can read the trackers performance in terms of accuracy (the tracker is more accurate if it is higher along the vertical axis) and robustness (the tracker fails fewer times if it is further to the right on the horizontal axis). Because the robustness does not have an upper bound we propose to interpret it as *reliability* for visualization purposes. The reliability of a tracker is defined as an exponential failure distribution, $R_S = e^{-SM}$. The value of M denotes mean-time-between-failures, i.e. $M = \frac{F_0}{N}$, where N is the length of the sequence. The reliability of a tracker can be interpreted as a probability that the tracker will still successfully track the object up to S frames since the last failure, assuming a uniform failure distribution, which is of course not completely true in all cases. Note that this formulation and the choice of S do not influence the order of the trackers and have the advantage that the value of S can be adjusted as a scaling factor for clearer visualization. While visualizing results this way is useful for quick interpretation of results, one should still consult the detailed raw values of average overlap and failure rate before making any final assumptions.

3.3.3 Theoretical trackers

For a better understanding of the complementing nature of the two measures we introduce four theoretical trackers denoting extreme prototypical tracker behaviors. The *first theoretical tracker*, denoted by TTA, always reports the region of the object to equal the image size of the sequence. This tracker provides regions that are too loose, but does not fail (overlap is never 0) and is therefore displayed in the bottom-right corner as it is extremely robust, but not accurate at all. The *second theoretical tracker*, denoted by TTS, reports its initial position for the entire sequence. This tracker will likely fail if the object moves, and will achieve better accuracy because of frequent manual interventions. The *third theoretical tracker*, denoted by TTF only tracks one frame and then deliberately reports a failure. This way the tracker maintains a high accuracy, however the failure rate is extremely high and the tracker is placed in top-left corner of the plot. The *fourth theoretical tracker* is denoted as TTO and represents an *oracle* tracker of fixed size. The tracker always correctly predicts the center position of the object, however, the size of the object is fixed. This tracker represents a practical performance limit for trackers that do not adapt the size of the reported bounding box which is the same as the initialization bounding box.

The performance scores for the theoretical trackers can be easily computed directly from ground-truth. The simplicity, intuitive nature, and the parameter-less design make them excellent interpretation guides in the graphical representations of results, such as A-R plot. In other words, they put the results of evaluated trackers into context by providing reference points for a given evaluation sequence.

3.3.4 Interpretation of results using the A-R plots

By establishing the selection of measures, visualization and the theoretical trackers as an interpretation guide, we can now provide several examples of results interpretation. The A-R plot in Figure 3.15 shows results, averaged over entire data-set. We can see that the LGT tracker is on average the most robust one in the set of evaluated trackers (positioned most right), but is surpassed in terms of accuracy by KCF, IVT and TLD (positioned higher). Especially the TLD tracker is positioned very low in terms of robustness, so the

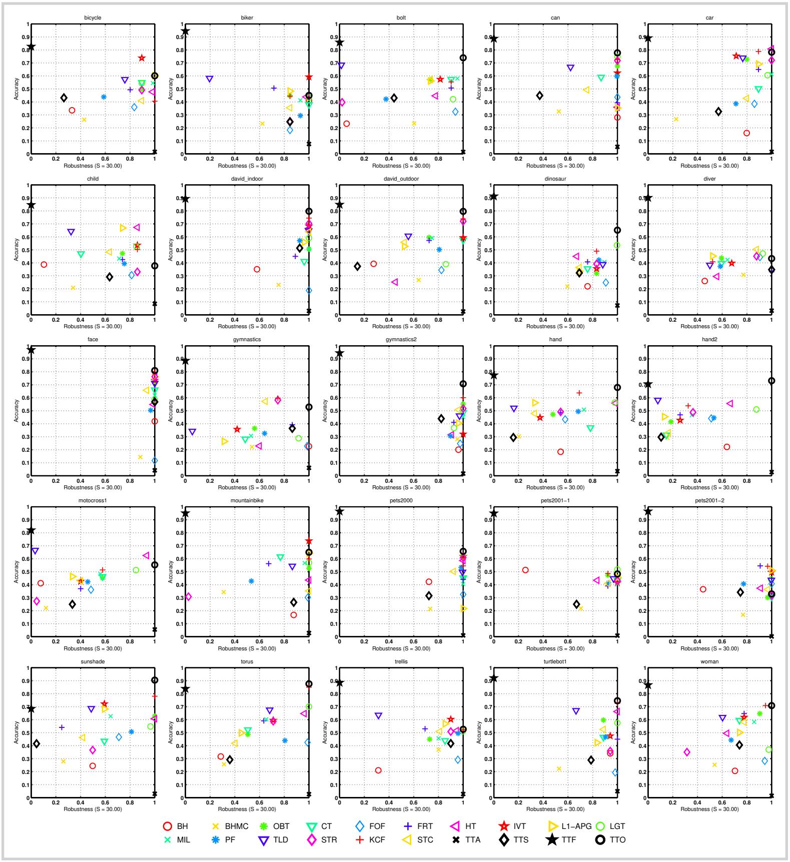


Figure 3.16

An accuracy-reliability data visualization for all trackers over all sequences.

high accuracy may in fact be a result of frequent reinitializations, a behavior that is similar to the TTF tracker. We acknowledge that this behavior of TLD is a design decision as the TLD is actually a long-term tracker that that does not report the position of the object if it is not certain about its location. The FOF tracker, on the other hand, is quite robust, but its accuracy is very low. This means that it most likely sacrifices accuracy by spreading across a large portion of the frame, much like TTA.

As the averaged results can convey only a limited amount of information, Figure 3.16 also contains per-sequence A-R plots for the trackers considered in our experiments. These plots show that the actual performance of trackers differs significantly between the sequences. Theoretical trackers TTA and TTF remain worse on their individual axes as expected, while the relative position of the other trackers changes depending on the properties of the individual sequence. In many sequences the TTO tracker achieves the best performance because of its ability to "predict" the position of the target. In cases where the size of the object changes this advantage becomes less apparent and trackers like IVT, L1-APG, HT, and LGT that account for this change can even surpass it in terms of accuracy (e.g. in *biker*, *child*, and *pets2001-2*). The sequence *diver* is interesting considering the results. Even though the object does not move a lot in the image space, which is apparent from the high robustness of the TTS tracker, the sequence has nevertheless proven to be very challenging for most of the trackers because of the large deformations of the object. The BH and BHMC trackers are on average very similar to the TTS tracker which would mean that they do not cope well with moving objects. At a closer look we can see that this is only true for some sequences (e.g. *torus*, *bicycle*, and *pets2000*). In other sequences both tracker perform either better than TTS, where the background remains static and can be well separated from the object (e.g. *sunshade*, *david_outdoor*, and *gymnastics2*), or worse, where the appearance of the background changes (e.g. *motocross1*, *child*, and *david_indoor*). Considering the good average performance of the LGT tracker we can see that the tracker performed well in sequences with articulated and non-rigid objects (e.g. *hand*, *hand2*, *dinosaur*, *can*, and *torus*), while the difference in case of more rigid objects (e.g. *face*, *pets2001-1*, and *pets2001-2*) is less apparent. In the plot for the *bolt* sequence we can see that the TLD tracker behaves similarly to the TTF tracker, i.e. fails a lot without actually drifting. On the other hand the TLD tracker works quite well in the case of *pets2000*, *pets2001-1*, and *pets2001-2* sequences where the changes in the appearance of the object are gradual.

3.3.5 *Fragmentation*

Recall that we have introduced the fragmentation indicator as a complementary indicator for the number of failures measure in Equation 3.13. Using this measure we can infer some additional properties of a tracker that would otherwise require looking at raw results. Fragmentation reflects the distribution of failures throughout the sequence. If the fragmentation is low then the failures are likely clustered together around some specific event (which can indicate a specific event that is problematic for the tracker). On the other hand, if the fragmentation is high, then the failures are uniformly distributed, independently of localized events in the sequence and can be most likely attributed to internal problems of the tracker. To demonstrate this property we have selected several cases where the number of failures is the same, but the fragmentation is different. In Figure 3.17 we can see three such cases. Several trackers, despite failing the same number of times do this for different reasons and in different intervals. On the *hand* sequence, the FRT tracker fails almost uniformly, while the BH tracker manages to hold to the target for a long time (the region is, however, estimated very poorly), but then fails to successfully initialize around frame 170 because of background clutter and motion. In *bicycle* and *bolt* sequences, the failures of PF tracker are concentrated on a specific event, most likely because of color ambiguity or small target size. The failures of the BHMC tracker are almost uniformly distributed over both sequences, most likely because of the problems of the tracker implementation (e.g. inability to cope with small target size).

3.3.6 *Sequences from the perspective of theoretical trackers*

The theoretical trackers, introduced in Section 3.3.3, provide further insights into each sequence from the perspective of the basic properties that each theoretical tracker represents. Because of their simplicity and absence of parameters, they can easily be applied to any annotated sequence and provide some insight about its properties. These properties can then be used when constructing an evaluation dataset or interpreting the results.

The TTA tracker will always achieve good robustness (no failures), but will produce high accuracy values only when the target will cover large part of the image frame. This tracker therefore measures the average relative size of the object. The TTS tracker will only achieve good robustness when the object remains stationary with respect to the image plane (e.g. the *diver* and the *face* sequence) and will also achieve good robustness when the size of the object does not change with respect to the initialization frame. The

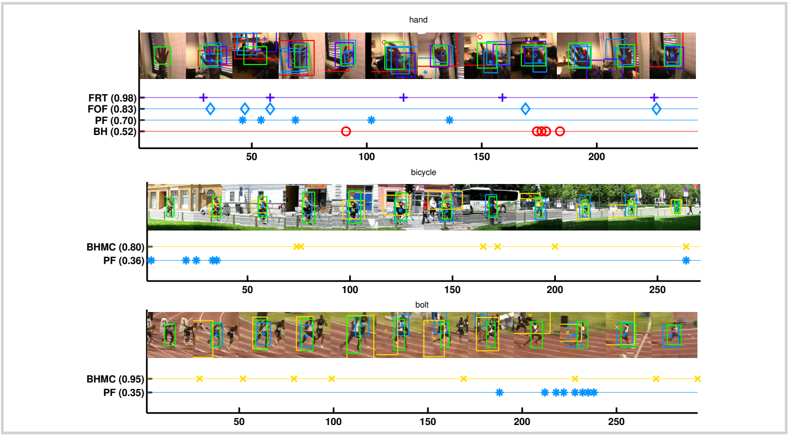


Figure 3.17

Selected results of the fragmentation analysis. Failures are marked on the time-line with symbols, the corresponding fragmentation values are shown in brackets next to tracker name.

TTF tracker will fail uniformly, however it will produce high accuracy only when there is no rapid motion predominantly present over the entire sequence. In Figure 3.16 we can observe that the assumption of no rapid motion is not true for the sequences *hand*, *hand2*, and *sunshade*. The TTO tracker will achieve good robustness (no failures), however, it will not achieve a good accuracy when the size of the object region changes a lot, e.g. in sequences *diver* and *gymnastics*. These observations can be extended to the entire set of sequences using clustering. As a demonstration we have used K-means clustering with expected number of clusters set to $K = 3$ to generate labels that are shown in Table 3.1. The labels are of course relative to the entire set, but they summarize these relative properties well, e.g., we can see that *face* sequence is similar to *diver* sequence in terms of movement, however, the *diver* sequence contains a lot of size changes. This simple approach could be in future extended to provide automated and less-biased sequence descriptions.

3.4 *Evaluating multiple trackers*

The performance measures that we have analyzed in the previous sections can tell us if a tracker performs better than another tracker on a given sequence, but in case of multiple trackers and a larger dataset, we need an extension of the methodology that would take into account the structure of the dataset and cases where some trackers are essentially equal.

Table 3.1

Sequence properties according to theoretical tracker performance.

	Size (TTA)	Motion (TTS)	Speed (TTF)	Size change (TTO)
bicycle	small	high	medium	medium
biker	large	medium	low	high
bolt	small	high	medium	medium
can	medium	high	medium	low
car	small	medium	medium	low
child	large	medium	medium	high
david_indoor	small	low	medium	low
david_outdoor	small	high	medium	low
dinosaur	large	medium	low	medium
diver	small	low	medium	high
face	medium	low	low	low
gymnastics	medium	low	medium	high
gymnastics2	small	low	low	medium
hand	small	high	high	medium
hand2	small	high	high	medium
motocross1	medium	high	medium	high
mountainbike	small	medium	low	medium
pets2000	small	medium	low	medium
pets2001-1	small	medium	low	high
pets2001-2	small	medium	low	high
sunshade	small	high	high	low
torus	small	high	medium	low
trellis	small	low	medium	high
turtlebot1	medium	medium	low	medium
woman	small	medium	medium	medium

3.4.1 Ranking trackers

Our approach to ranking multiple trackers is inspired by [82, 112, 113]. The key idea is that trackers are not merely ordered according to their performance scores, but that a group of trackers is assigned an equal rank if they perform equally well on a given sequence.

After ranking trackers on a given sequence by ordering the raw scores, the ranks are corrected as follows. For tracker i, a group of equivalent trackers E_i is determined. This group contains indices of trackers that performed equally well as the selected tracker (including the tracker i). The corrected rank of the i-th tracker is then calculated based on the ranks of trackers in the group of equivalent trackers. One way of correcting the rank is to assign tracker i an average rank of all trackers in E_i

$$\hat{R}_i = \frac{1}{\|E_i\|} \sum_{j \in \mathbb{E}_i} R_j ,$$

where $R_\cdot$ denotes uncorrected rank and $\hat{R}_\cdot$ denotes corrected rank. Note that this equality is not transitive, and should not be mistaken for a classical equivalence relation. For example, consider trackers T_1, T_2 and T_3. It may happen that a tracker T_2 performs equally well as T_1 and T_3, but this does not necessarily mean that T_1 performs equally well as both, T_2 and T_3. The equality relation between trackers should therefore be established for each tracker separately.

To determine for each tracker the group of equivalent trackers, a measure of equivalence on a given sequence is required. In case of accuracy, a per-frame overlap is available for each tracker. One way to determine equivalence in this case is to apply a paired test to determine whether the difference in accuracies is statistically significant. When the differences are distributed normally, the Student's t-test, which is often used in the aeronautic tracking research [114], is the appropriate choice. However, in a preliminary study we have applied Anderson-Darling tests of normality [115] and have observed that the accuracies in frames are not always distributed normally, which might render the t-test inappropriate. As an alternative, the Wilcoxon Signed-Rank test as in [112] can be used. In case of robustness, several measurements of the number of times the tracker failed over the entire sequence in different runs is obtained. These values cannot be paired, and the Wilcoxon Rank-Sum (also known as Mann-Whitney U-test) [112] is used instead to test the difference in the average number of failures.

We can calculate the average rank for the i-th tracker by averaging over all sequences. The averaging over sequences assumes that every sequence contributes equally to the final ranking, regardless of their length. Another thing that has to be taken into account is that while statistical equivalence is a widely accepted method, it is also very conservative. When establishing an equivalence between two trackers, we have to keep in mind that statistical significance does not directly imply a practical difference [116]. If a single rank is needed (e.g. for a competition) the most straightforward way of obtaining it is by giving an equal weight to both performance measures and simply average the two corresponding rankings for accuracy and robustness. Note, however, that this severely reduces the interpretability of the results.

3.4.2 Visualization of ranking results

Ranking results can be visualized using plots similar to the A-R plots proposed in the Section 3.3.1 (Figure 3.15). We display the rank results either for a particular sequence or averaged over the entire dataset. Since each tracker is presented in terms of its rank with respect to robustness and accuracy, we can plot it as a single point on the corresponding 2D A-R rank plot as shown in Figure 3.18. Trackers that perform well relative

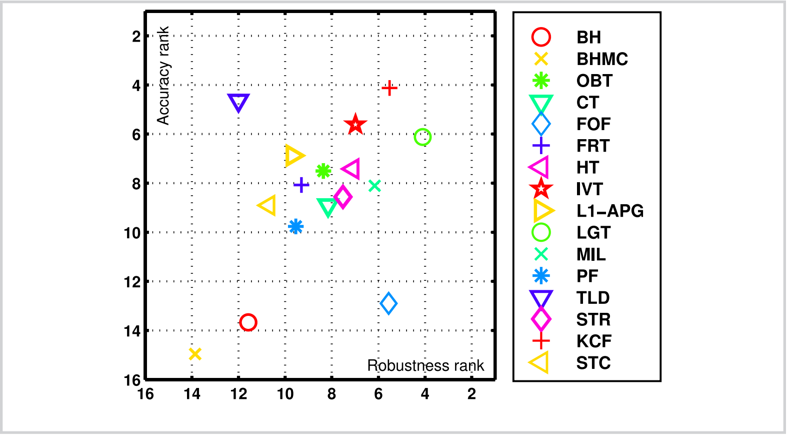


Figure 3.18

Example of an A-R rank plot for the experiment, presented in Section 3.3. The data was averaged over sequences.

to the others are positioned in the top-right part of the plot, while the, relatively speaking, poorly-performing trackers occupy the bottom-left part. Because the A-R ranking plot tends to *normalize* the distances between trackers, we recommend to always use it in tandem with the raw A-R plot on the same results when interpreting the results.

Comparing the ranking results in Figure 3.18 to the averaged raw results in Figure 3.15 shows that ranking "normalizes" the plot, but the overall structure remains similar. The LGT tracker is the most robust, while the KCF tracker achieves the best accuracy ranking and also achieves the best average ranking if both performance aspects are considered equal.

3.5 Performance evaluation systems

Comparing a set of tracking algorithms on several dozen sequences using many measures is a complex, large-scale experiment that has to be properly managed. The easiest way to perform complex experiments objectively is to automate the entire process. To address this issue we have created two visual tracking evaluation systems, both designed to perform large-scale experiments, but satisfying different evaluation scenarios. Both systems are run on multiple platforms and support the same tracker integration mechanism that enables multi-programming language compatibility. More details about the software is provided in Appendix C.

3.5.1 VOT toolkit

The *VOT toolkit* is an evaluation system that was implemented in Matlab/Octave language and was designed to perform comparative evaluation of multiple trackers, using proposed evaluation methodology, on a set of sequences and a set of experiments that can include sequence transformations such as gray-scale conversion, image noise, etc (which is performed on-the-fly by the system). The main focus of the VOT toolkit is to enable execution of multiple tracking algorithms as well as the analysis of the results and generation of informative reports. To solve the problem of integrating various trackers, written in different languages into our evaluation system, we have designed a simple protocol that uses standard input and output stream for communication between a tracker and our system. The *TraX* protocol is described in detail in Appendix B. Because of the simplicity of the protocol, existing trackers can be adapted for basic use within the system within an hour. The system is available as an open-source software as a part of the Visual Object Tracking Challenge initiative and can be used by other researchers to perform low-effort evaluation of their trackers and comparison to the state-of-the-art.

3.5.2 TraXtor

The second evaluation system is called *TraXtor* and is written in Java. Its main purpose is to support tracker development and continuous evaluation. In comparison to the VOT toolkit the TraXtor allows parallel execution of individual tracker runs, therefore shortening the required execution time for evaluation on the entire set of sequences. It also supports parameter space exploration, but does not have advanced result analysis capabilities. As it is evident from the name, it also supports the TraX protocol.

3.6 Performance evaluation in practice

The measures and protocols that were proposed in this chapter have already been adopted as the foundation of the evaluation methodology of a recently organized visual tracking challenges VOT2013 [100] and VOT2014 [101], where a rigorous analysis in terms of accuracy and robustness has provided multiple interesting insights into performance of individual trackers. In this section we briefly summarize the two challenges that were organized as a legacy of our research that was reaffirmed by the interest and acceptance of the research community.

3.6.1 The VOT2013 challenge

For this challenge, the presented tracker evaluation and comparison methodology was applied first to a practical large-scale experiment in a form of a research competition that we have co-organized in an international consortium of researchers. Instead of implementing or adapting the code of various existing state-of-the-art trackers ourselves, we invited researchers to participate by running the experiments themselves, using the first version of the evaluation system that was described in Section 3.6, and sending us the raw results of the experiments. The dataset that was used in the competition was constructed in a novel way using clustering by sequence properties which resulted in a diverse selection of a small set of sequences. In comparison to the previous trends in visual tracking evaluation that proposed using more and more sequences, this approach of dataset construction, together with the proposed performance methodology that efficiently uses the evaluation material offers a scalable alternative while providing a good estimation of the true performance. The results of the challenge were presented at a VOT2013 [100] workshop that was held in conjunction with the International Conference on Computer Vision (ICCV2013) and was well accepted.

3.6.2 The VOT2014 challenge

Based on the success from the previous year we have co-organized another challenge that extended the previous one. We constructed a new dataset with additional sequences. The sequences were annotated using rotated rectangle annotations that were better estimate the region of rotated or articulated objects. In addition we have introduced the concept of practical difference that takes into account the noise of the annotations when considering if two trackers are equivalent in terms of accuracy. From the technical side we have presented an improved evaluation toolkit that also introduced the communication protocol that we describe in Appendix B. The results of the challenge were presented at a VOT2014 [101] workshop that was held in conjunction with the European Conference on Computer Vision (ECCV2014). The VOT2014 benchmark was also adopted by the OpenCV Challenge for evaluation of promising visual tracking algorithms.

3.7 Summary

In this chapter we have addressed the problem of performance evaluation in monocular single-target short-term visual tracking. We have presented three core requirements for a comprehensive evaluation framework. We have addressed two of them, the first one being selection of the evaluation measures. Through theoretical and experimental analysis we have investigated various popular performance evaluation measures, discussed their pitfalls and showed that many of the widely used measures are equivalent. Since some measures reflect certain aspect of tracking performance, combining those that address the same aspect provides no additional information regarding the performance or even introduces bias toward a certain aspect of performance to the result. Based on the results of our experiment we have proposed to use a pair of two existing complementary measures. This pair, that we call the A-R pair, takes into account the accuracy (using *average overlap*) and the robustness (using *failure rate*) of each tracker. We have also proposed an intuitive way of visualizing the results in a 2-dimensional scatter plot, called the A-R plot. Additionally, we have introduced fragmentation as an additional indicator for distribution of failures. We have introduced several theoretical trackers that can be used to quickly review the results of the evaluated trackers in terms of basic properties that the theoretical trackers exhibit. We have also shown that the theoretical trackers can be used for automatic annotation of sequence properties from a tracker viewpoint.

The A-R measures were also extended to ranking multiple visual trackers on a given set of sequences where equivalent performance is also important and has to be accounted for. We have described the basic properties of the two performance evaluation systems that we have developed. Both systems support the same way of third-party tracker integration using a custom communication protocol that we have designed. Both systems and a reference protocol implementation are available as open-source software. This way researchers in the field of visual tracking can save time when it comes to evaluation by reusing existing evaluation tools or develop new specialized evaluation software that is immediately usable to a large number of people.

As we have shown in the last section, the work on visual tracking performance evaluation that we have described in this chapter was accepted by the community and is now integrated into and being built upon in the scope of the Visual Object Tracking Challenge that represents an ongoing effort to promote a consistent evaluation methodology, thus pushing forward the field of visual tracking.

4

Tracking with a coupled-layer appearance model

In this chapter we present a first working instance of a hierarchical appearance model, based on the concept, described in Section 1.2. The description is based on our earlier work, published in [106, 107], where the appearance model was presented as the *coupled-layer* appearance model because of the collaborative interaction of local and global appearance description that form a visual hierarchy of two layers. In Section 4.1 we present the details about the proposed appearance model. In Section 4.2 we describe the integration of the appearance model with a motion model in a tracking framework and the software implementation details. Section 4.3 contains an in-depth evaluation of the resulting tracker, both in terms of parameter analysis and as a comparison to the related work. We conclude the chapter with a summary in Section 4.4.

4.1 The coupled-layer appearance model

The main idea behind the proposed appearance model is to couple the local appearance description that is able to adapt to the geometrical deformations of non-rigid targets with a global appearance model of an object that guides the appearance model across changes in the appearance of the object. The coupled-layer appearance model contains only two layers according to the hierarchical appearance model concept and is this respect the simplest non-trivial hierarchical appearance model, however, as we show in the experimental evaluation that we will present in Section 4.3, the tracker that uses a coupled-layer model is capable of tracking in many hard cases of short-term tracking scenarios.

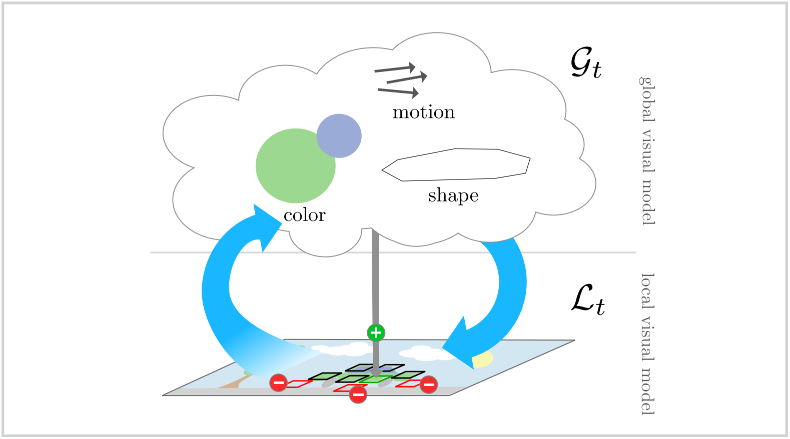


Figure 4.1

Illustration of the proposed coupled-layer appearance model. The local layer is a geometrical constellation of local parts that describe the target's local visual properties. The global layer encodes the target's global visual features in a probabilistic model.

As illustrated in Figure 4.1, the coupled-layer appearance model is organized in a *local* and a *global* layer,

$$\mathcal{V}_t = \{\mathcal{L}_t, \mathcal{G}_t\}.$$

The local layer $\mathcal{L}_t$ is a geometrical constellation of visual parts (parts) that describe the target's local visual/geometrical properties. As the target's appearance changes or a part of the object gets occluded, some of the parts in the appearance model cease to correspond to the target's visible parts. Those are identified and gradually removed from the model. The allocation of the new parts in the local layer is constrained by the global layer $\mathcal{G}_t$ that encodes the target's global visual features. The global layer maintains a probabilistic model of target's global visual features such as color, shape and apparent motion and is adapted during tracking. This adaptation is in turn constrained by focusing on the stable parts in the local layer.

4.1.1 The local layer

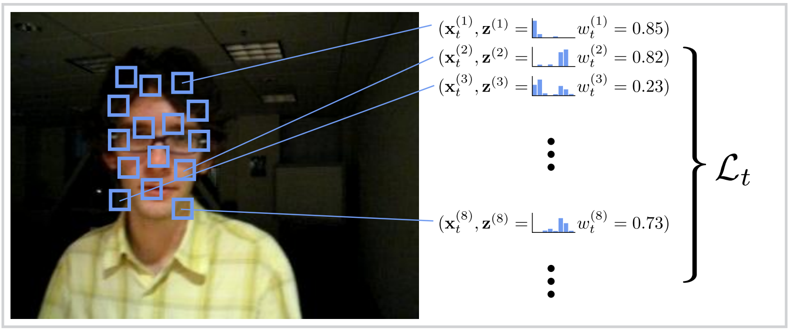
The local layer is the layer that is closest to the current appearance of the object and is updated all the time to adapt to small changes that occur during two frames in a sequence.

Definitions: The local layer is defined as a geometrically constrained constellation of local parts. Each part has its own appearance model. The state, $\mathcal{L}_t$, of the the local layer at time-step t is described by a geometrical constellation of parts:

$$\mathcal{L}_t = \{\langle \mathbf{x}_t^{(i)}, \mathbf{z}^{(i)}, w_t^{(i)} \rangle\}_{i=1:N_t}, \tag{4.1}$$

where $\mathbf{x}_t^{(i)}$ and $\mathbf{z}^{(i)}$ are the position in image space and the appearance model of the i-th part, respectively, and $w_t^{(i)}$ is a weight that reflects belief that the target is well-represented by that part, i.e. the weights sum to one across all parts. An example of such part-set is illustrated in Figure 4.2. The target's center, $\mathbf{c}_t$ is defined as a weighted average over the parts. In the definitions that follow we will denote the set of positions of all parts at time-step t by $\mathbf{X}_t = \{\mathbf{x}_t^{(i)}\}_{i=1:N_t}$.

The main idea of part-based appearance models is that appearance models of individual parts, in our case denoted by $\mathbf{z}^{(i)}$, are not too complex as the strength of the entire appearance model is not in their individual matching ability but in their joint descriptive strength.



$(\mathbf{x}_t^{(1)}, \mathbf{z}^{(1)} =$ $w_t^{(1)} = 0.85)$
$(\mathbf{x}_t^{(2)}, \mathbf{z}^{(2)} =$ $w_t^{(2)} = 0.82)$
$(\mathbf{x}_t^{(3)}, \mathbf{z}^{(3)} =$ $w_t^{(3)} = 0.23)$
$(\mathbf{x}_t^{(8)}, \mathbf{z}^{(8)} =$ $w_t^{(8)} = 0.73)$
$\mathcal{L}_t$

Figure 4.2

Describing appearance of an object using a constellation of parts. In this example simple histograms are used as appearance models of individual parts.

The local appearance model of a part is encoded by a gray-level histogram $\mathbf{z}^{(i)}$ which is extracted when a part is initialized in the constellation and remains unchanged during tracking to prevent drifting; the size of the extraction region is also constant for all the parts. We have chosen this simple appearance representation due to its simplicity and invariance to rotation. Let $\mathbf{z}_t$ be a histogram extracted at the current location of a part $\mathbf{x}_t$. We define the visual likelihood $\rho(\cdot, \cdot)$ between the appearance model of the part and the visual information at position $\mathbf{x}_t$ as the Bhattacharryya distance between the histograms [39]

$$\rho(\mathbf{z}^{(i)}, \mathbf{z}_t) = \sum_{b=1}^{B} \sqrt{\mathbf{z}^{(i)}[b]\mathbf{z}_t[b]}, \tag{4.2}$$

where B denotes the number of bins in the histogram and $[\cdot]$ represents addressing of individual bin in a histogram.

Matching: Matching of the local layer to a new image is a process of finding the optimal positions of individual parts while also taking into account the geometrical constraints between the parts from previous frames. At each frame we start from an initial estimate from the previous frame, $\mathbf{X}_{t-1}$, and the set of current image measurements $\mathbf{Y}_t$, and seek the value of $\mathbf{X}_t$ that maximizes the joint probability $p(\mathbf{Y}_t, \mathbf{X}_t | \mathbf{X}_{t-1})$. By treating the local layer appearance model $\mathcal{L}_t$ as a mixture model, in which each part competes to explain the target's appearance, we can decompose the joint distribution into

$$p(\mathbf{Y}_t, \mathbf{X}_t | \mathbf{X}_{t-1}) = \sum_{i=1}^{N_t} w_t^{(i)} p(\mathbf{Y}_t, \mathbf{X}_t | \mathbf{X}_{t-1}, \mathbf{z}_t^{(i)}), \tag{4.3}$$

where $w_t^{(i)}$ quantifies the representativeness of the i-th part for the tracked model using weights of the parts. This is a very general formalization and assumes that all parts are directly mutually dependent. This may not be the case, espectially in case of non-rigid objects some parts are only linked between themselves through their neighbors. A *neighborhood* of the i-th part is a subset of parts that are directly connected to the part; the relation is symmetrical. We can therefore write

$$p(\mathbf{Y}_t, \mathbf{x}_t^{(i)} | \mathbf{X}_{t-1}, \mathbf{z}^{(i)}) \propto p(\mathbf{Y}_t, \mathbf{x}_t^{(i)} | \varepsilon_t^{(i)}, \mathbf{z}^{(i)}), \tag{4.4}$$

where $\varepsilon_t^{(i)}$ denotes the set of the i-th part's local neighbor parts. Assuming the independence of geometrical constraints and appearance similarity the distribution in the right-hand side of (4.4) can now be further decomposed in terms of visual and geometrical aspects as

$$p(\mathbf{Y}_t, \mathbf{x}_t^{(i)} | \varepsilon_t^{(i)}, \mathbf{z}^{(i)}) = p(\mathbf{Y}_t | \mathbf{x}_t^{(i)}, \mathbf{z}^{(i)}) p(\mathbf{x}_t^{(i)} | \varepsilon_t^{(i)}), \tag{4.5}$$

where we assume that the visual likelihood measurement at the i-th part is independent from the other parts. We define the visual likelihood of the i-th part to the location as

$$p(\mathbf{Y}_t | \mathbf{x}_t^{(i)}, \mathbf{z}^{(i)}) \propto e^{-\lambda_\mathrm{v} \rho(\mathbf{z}^{(i)}, \mathbf{z}(\mathbf{x}_t^{(i)}))}, \tag{4.6}$$

where $\rho(\cdot, \cdot)$ is the visual distance between the appearance model of part ($\mathbf{z}^{(i)}$) and extracted visual information, as defined in (4.2), and λ_v is a constant factor.

In the case of non-rigid objects, the neighborhood of a part is a set of parts that constraints the movement of the part. Since non-rigid objects tend to deform quickly and because we do not know the properties of the object in advance, the estimation of the neighborhood has to be derived from the available information about the structure of the object, which is scarce in most tracking scenarios where the exact type of the tracked object is not known. In this case one has to rely on generally applicable heuristics like proximity of parts. We propose two proximity heuristics to determine the neighborhood of a part. In the first approach, the neighbors of a part are the parts that are directly connected with that part in a Delaunay triangulated mesh of an entire set of parts.

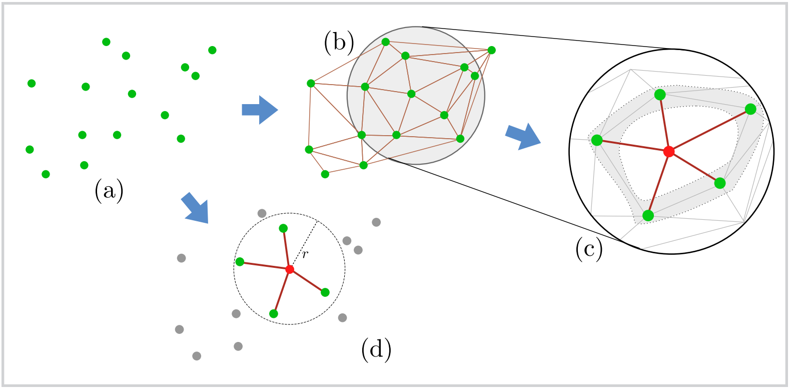


Figure 4.3 Determining the neighborhood of a part. The figure shows: positions of the parts (a), Delaunay mesh edges for the point set (b), neighborhood of a single part in a Delaunay graph (c), neighborhood of a single part according to the proximity threshold r.

The second approach is to consider Euclidean distance between parts. The neighbors of a part are the parts whose distance to that part is lower than a specific threshold. Both approaches are illustrated in Figure 4.3. The advantage of the first approach is that it ensures a small size neighborhood even if the parts are positioned far apart, but the difference in distance between closest and farthest neighbor can be big. This approach is more suitable for sparse evenly distributed part-sets, where the mesh can be computed reliably [1]. The second approach is faster and more suitable for dense part-sets, but it can be sensitive to the neighborhood threshold parameter and object size change.

The constraints enforced on the local geometry by the neighborhood are formalized using an elastic deformation model

$$p(\mathbf{x}_t^{(i)}|\varepsilon_t^{(i)}) \propto e^{-\lambda_\mathrm{g}||\mathbf{x}_t^{(i)} - \mathbf{A}(\varepsilon_t^{(i)})\mathbf{x}_{t-1}^{(i)}||}, \tag{4.7}$$

where $\mathbf{A}(\varepsilon_t^{(i)})$ is a transformation matrix computed from correspondences between the i-th part's initial and current neighborhoods and λ_g is a constant factor. Similarly to Martinez and Binefa [61], we assume that the movements of a parts in a neighborhood is constrained by an affine transformation, which means that the transformation $\mathbf{A}(\varepsilon_t^{(i)})$ in (4.7) can be calculated by estimating an affine transformation from the past (time $t-1$) and current (time t) positions of the parts in neighborhood $\varepsilon_t^{(i)}$.

Note that this geometric model assumes that the deformations of the constellation are locally approximately affine. Therefore, during adaptation of the local layer to the

[1]Delaunay triangulation is numerically unstable when points lay close together.

target's current appearance, we seek a deformation $\tilde{\mathbf{X}}_t$ of an initial set of parts from previous frame $\mathbf{X}_{t-1}$ that maximizes the joint probability in (4.3)

$$\tilde{\mathbf{X}}_t = \arg\max_{\mathbf{X}_t} p(\mathbf{Y}_t, \mathbf{X}_t | \mathbf{X}_{t-1}). \tag{4.8}$$

Determining the parameters of the unknown deformation $\tilde{\mathbf{X}}_t$ is a difficult optimization problem due to the high dimensionality and complexity of the problem space. The objective function (4.3) may contain many local maxima and the optimization may converge to the wrong one, therefore the optimization method has to be designed to take this into account. A two-stage optimization approach that is based on the idea of Graduated Non-Convexity [117] can be used in such occasions to ensure a more stable solution. An intuitive approach, based on an observation that transformation of a part-set can be split in a global rigid transformation and residual corrections, is to split the optimization into coarse global optimization that determines the optimal state approximately in a low-dimensional sub-space and refinement phase that improves this estimate further, i.e.

$$\mathbf{x}_t^{(i)} = \mathbf{A}_t \mathbf{x}_t^{(i)} + \delta_t^{(i)}, \;\; \delta_t^{(i)} \in \Delta_t. \tag{4.9}$$

Determining the parameters $\mathbf{A}_t$ and Δ_t of (4.8) is a high-dimensional optimization problem that we approach by optimizing (4.3) using the cross-entropy stochastic optimization method [118]. First we make an initial estimate by opptimizing (4.3) w.r.t. the global affine deformation. The problem is considered in a five-dimensional problem space $G = [t_x, t_y, r, s_x, s_y]$, where t_x and t_y represent the target's position, r represents rotation and s_x and s_y represent scale. The five parameters define a transformation matrix $\mathbf{A}(G)$ as

$$\mathbf{A}(G) = \begin{bmatrix} s_x \cos(r) & -\sin(r) & t_x \\ \sin(r) & s_y \cos(r) & t_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.10}$$

The cross entropy method iteratively searches for an optimal combination of parameters according to the cost function by updating the candidate probability distribution over the parameters of G. In our case we model the probability distribution as a normal distribution, defined by a mean value μ and the covariance matrix $\mathbf{\Sigma}$ as seen in the overview of the algorithm in Figure 4.4. At the beginning the parameters of the

- Input: The set of parts $\mathbf{X}_{t-1}$.
- Initialization: Set the initial mean and covariance $\mu_0 = \mu_G$ and $\mathbf{\Sigma}_0 = \mathbf{\Sigma}_G$.
- For $j = 1 \dots M_G$:
    - Sample S_G samples of parameter values from $\mathcal{N}(\mu_{j-1}, \mathbf{\Sigma}_{j-1})$.
    - For each sample construct the corresponding affine transformation and evaluate (4.3) for $\tilde{\mathbf{X}}_t = \{\mathbf{A}_t^{\mathrm{G}} \mathbf{x}_{t-1}^{(i)}\}_{i=1 \dots N_t}$.
    - Select E_G best samples according to (4.3) and use them to recalculate μ_j and $\mathbf{\Sigma}_j$.
    - Break the loop if $det(\Sigma_j)$ is smaller than convergence threshold.
- Output: The optimal global affine transformation $\mathbf{A}_t^{\mathrm{G}}$, constructed from the parameters of μ_j.

Figure 4.4

Initial global optimization using the cross-entropy method.

distribution are initialized using constant values, i.e. $\mu_G = [0, 0, 0, 1, 1]$ and $\mathbf{\Sigma}_G = diag(m_G, m_G, r_G, s_G, s_G)$. The idea of the cross-entropy method is that the parameters of the distribution are iteratively updated using only the E_G best samples of the total S_G samples sampled from the distribution $\mathcal{N}(\mu, \mathbf{\Sigma})$. This can be done by computing a weighted mean and a weighted covariance of the best samples using the cost function to determine the weights. This process is then repeated for M_G iterations or until the distribution collapses, i.e. the determinant of the covariance matrix falls bellow a convergence threshold.

Note that in the case of the global optimization step, equation (4.3) can be simplified. Because a global affine transformation constraint is assumed in the problem space it is clear that every local affine transformation $\mathbf{A}(\varepsilon_t^{(i)})$ from (4.7) equals to the global affine transformation $\mathbf{A}(G)$, therefore the value of (4.7) is 1 for every part. Therefore, (4.5) can be reduced to a weighted sum of a visual likelihood for every part

$$p(\mathbf{Y}_t, \mathbf{X}_t | \mathbf{X}_{t-1}) = \sum_{i=1}^{N_t} w_t^{(i)} e^{-\lambda_{\mathrm{v}} \rho(\mathbf{z}^{(i)}, \mathbf{z}(\mathbf{x}_t^{(i)}))}, \tag{4.11}$$

After convergence of the first stage, the value of $\mathbf{A}(G)$ is fixed and the positions of each part $\mathbf{x}_t^{(i)}$ are additionally refined by using a stochastic coordinate descent in the cross-entropy framework. The search for optimal position of every part is represented as a cross-entropy optimization problem in a two dimensional problem space using (4.5)

```
- Input: The set of parts $\mathbf{x}_t^{(i)} = \mathbf{A}_t^{\mathrm{G}} \mathbf{x}_{t-1}^{(i)}$
- Initialization: Set the initial mean vectors and covariance matrices $\mu_0^i = \mathbf{A}_t^{\mathrm{G}} \mathbf{x}_{t-1}^{(i)}$
  and $\mathbf{\Sigma}_0^i = \mathbf{\Sigma}_L$.
  Compute the neighborhood of each part.
- For $j = 1 \dots M_L$:
    - For each part $\mathbf{x}_t^{(i)}$:
        - Sample $S_L$ samples from $\mathcal{N}(\mu_{j-1}^i, \mathbf{\Sigma}_{j-1}^i)$
        - For each sample $\Delta_t^{(i)}$ evaluate (4.5) for $\mathbf{x}_t^{(i)} = \mathbf{A}_t^{\mathrm{G}} \mathbf{x}_{t-1}^{(i)} + \Delta_t^{(i)}$.
        - Select $E_L$ best samples according to (4.5) and use them to recalculate $\mu_j^i$
          and $\mathbf{\Sigma}_j^i$
- Output: The optimized set of points $\mathbf{X}_t = \{\mathbf{x}_t^{(i)} = \mathbf{A}_t^{\mathrm{G}} \mathbf{x}_{t-1}^{(i)} + \mu_j^i\}_{i=1 \dots N_t}$
```

Figure 4.5

Refinement optimization step using the cross-entropy method.

as a cost function. The initial distribution for all parts is set using constant values, i.e. $\mu_L = [0, 0]$ and $\mathbf{\Sigma}_G = diag(m_L, m_L)$. For each iteration of the algorithm we iterate through all the parts and perform an iteration of the cross-entropy method for the part, while fixing the positions of all other parts. The algorithm outline for the second step of the optimization is presented in Figure 4.5.

Updating: The parts in the set are reasonably small, focused only on a local section of the object, the appearance of which is described using a grayscale histogram. This simple appearance representation is robust to some deformations, e.g., rotation and provides good short-term tracking support, especially when using more such parts together. However, it is not sufficient for more than a short period of time, usually ten to twenty frames. In the long run some parts become outdated. Updating appearance models of individual parts would result in drifting, therefore we employ an alternative update strategy where entire parts are replaced with new ones once the old parts become outdated. Because different parts become outdated at different points in the sequence, the membership changes that occur in the part-set are gradual.

Recall from (5.1) that there is an importance weight $w_t^{(i)}$ associated with each part. It reflects the belief of the corresponding part in the mixture of parts that can change over time, depending on how reliable is a specific part. The dynamics of the weights is governed by two rules simple, but effective rules: (1) the quality of visual match for a part which is based directly on the visual likelihood, defined in (5.6), and (2) the *drift from the flock* [10], which is defined by a sigmoid function:

$$p(\mathbf{x}_t^{(i)}|\mathbf{X}_t) = \frac{1}{1 + e^{\lambda_D (mdst(\mathbf{x}_t^{(i)}, \mathbf{X}_t) - T_D)}}, \tag{4.12}$$

where $mdst(\mathbf{x}_t^{(i)}, \mathbf{X}_t)$ stands for the median of Euclidean distances between the part position and position of every other part in the set. The T_D and λ_D are constants that determine the size of the object and the influence of the consistency constraint respectively. The new proposed weight $\hat{w}_t^i$ equals to

$$\hat{w}_t^i = p(\mathbf{Y}_t|\mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)}|\mathbf{X}_t). \tag{4.13}$$

After matching the part set to the new frame, each part is analyzed and its weight is modified in an autoregressive manner using the proposed estimate $\hat{w}_t^i$

$$w_t^i = \lambda_W w_{t-1}^i + (1 - \lambda_W)\hat{w}_t^i, \tag{4.14}$$

where λ_W is defined as a persistence constant and $\hat{w}_t^i$ is defined as the estimated weight in the current time step. If a certain part does not represent the object well for a period of time, its weight becomes very low (lower than a threshold T_R) which results in the removal of the part from the set. To maintain numerical stability[2] and avoid unnecessary computations, we also merge parts that are too close to each other by only retaining part with the highest weight. We define a subset of outdated parts as

$$\begin{aligned} \mathcal{L}^{\dagger} = \{\langle \mathbf{x}^{(i)}, \mathbf{z}^{(i)}, w^{(i)} \rangle \in \mathcal{L} | w^{(i)} < T_R \vee \\ (\exists j : \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\| < T_M \wedge w^{(i)} > w^{(j)})\}. \end{aligned} \tag{4.15}$$

A quality of a good part-based representation of an object is that the object is well covered with the parts. To ensure a good coverage of the object, new parts have to be added in the local layer to replace the ones that are removed. The parts are allocated by sampling their position from a probability density function (pdf) that determines

[2] E.g., Delaunay triangulation works better if the input points are not too close to each other.

locations in the image which are likely to belong to the object. This pdf is provided by the global layer using the process that is described in the next section.

The weight $w_t^{(i)}$ of the allocated part is initialized with a value of twice the threshold for part removal, i.e., $w_0 = 2T_\mathrm{R}$. The remaining question is *how many* parts should be allocated to cover the entire object. Since we do not known the exact size of the object, we propose the following approximation. Let $\tilde{N}_t$ denote the number of parts in the local layer after removing the irrelevant parts. We define N_t^cap to be the local layer's *capacity*, i.e., the maximum number of parts allowed in the local layer at time-step t. To allow the number of allocated parts to vary with the target's size, we always try to allocate at most $N_t^\mathrm{all} \leq N_t^\mathrm{cap} - \tilde{N}_t + 1$ new parts. To prevent sudden significant changes in the estimated capacity, we adapt it using the autoregressive scheme:

$$N_{t+1}^\mathrm{cap} = \alpha_\mathrm{cap} N_t^\mathrm{cap} + (1 - \alpha_\mathrm{cap}) N_t, \tag{4.16}$$

where $N_t = N_t^\mathrm{all} + \tilde{N}_t$ and α_cap is an exponentially forgetting factor.

4.1.2 The global layer

The global layer in the coupled-layer appearance model captures different aspects of the target's global appearance. In total we have evaluated the following three visual properties: color C_t, apparent motion M_t and shape S_t, therefore the state of the global layer, $\mathcal{G}_t$, is denoted as

$$\mathcal{G}_t = \{C_t, M_t, S_t\}. \tag{4.17}$$

When required, this information can be used to allocate new parts that are added to the part-set. The position of a new part is determined by drawing samples from the following distribution

$$p(\mathbf{x}|C_t, M_t, S_t) \propto p(C_t, M_t, S_t|\mathbf{x}). \tag{4.18}$$

Assuming that the modalities are independent given a position $\mathbf{x}$, equation (4.18) factors into

$$p(\mathbf{x}|C_t, M_t, S_t) \propto p(C_t|\mathbf{x}) p(M_t|\mathbf{x}) p(S_t|\mathbf{x}). \tag{4.19}$$

Each visual modality has to be addressed in its own manner regarding storing the visual information, updating the representation as well as comparing the information to the image to generate a probability distribution. Below we describe the details about all three modalities.

Color: The global color model is encoded by two color histograms $\mathbf{h}_t^F$ and $\mathbf{h}_t^B$, the first corresponding to the object and the second to the background. This way the model can focus on the colors that separate the object from the background, which is particularly important if some of the background color enters the foreground model. Let $I(\mathbf{x})$ be a pixel value at position $\mathbf{x}$ in image I. Using the histograms, the probability that a pixel corresponds to the background or foreground is $p(x|\mathrm{F}) = \mathbf{h}_t^F(I(\mathbf{x}))$ and $p(x|\mathrm{B}) = \mathbf{h}_t^B(I(\mathbf{x}))$, respectively. The likelihood that a pixel at location $\mathbf{x}$ belongs to a target is therefore

$$p(C_t|\mathbf{x}) = \frac{p(\mathbf{x}|\mathrm{F})p(\mathrm{F})}{p(\mathrm{F})p(\mathbf{x}|\mathrm{F}) + (1 - p(\mathrm{F}))p(\mathbf{x}|\mathrm{B})}. \tag{4.20}$$

Both histograms are updated during tracking as follows. After the matching operation in the local layer is completed, a histogram $\hat{\mathbf{h}}_t^F$ is extracted in the current image from the regions that correspond to the parts of the local layer. The background histogram $\hat{\mathbf{h}}_t^B$ is extracted from a ring-shaped region defined by the convex hull of the parts in the local layer. These histograms are used to update the global color model by an autoregressive scheme

$$\begin{aligned} \mathbf{h}_{t+1}^F &= \alpha_F \mathbf{h}_t^F + (1 - \alpha_F)\hat{\mathbf{h}}_t^F \\ \mathbf{h}_{t+1}^B &= \alpha_B \mathbf{h}_t^B + (1 - \alpha_B)\hat{\mathbf{h}}_t^B, \end{aligned} \tag{4.21}$$

where α_F and α_B determine the rate of adaptation.

Motion: The apparent motion model is defined by the local-motion model from [33]. Briefly, the local motion model [33] first determines salient points $\{\mathbf{x}_i\}_{i=1}^{N_\mathrm{s}}$ with sufficient texture in the image. It then computes the motion likelihood $p(\mathbf{x}_i|M_t)$ at each salient point $\mathbf{x}_i$ by comparing the local velocity of a pixel $\mathbf{v}(\mathbf{x}_i)$ (estimated by Lucas-Kanade optical flow [119]) with the global velocity $\mathbf{v}_t$ estimated by the tracker. In our implementation we apply Harris corner detection [120] to determine the salient points. As in [33], the motion likelihood at salient point $\mathbf{x}_i$ is defined as

$$p(\mathbf{x}_i|M_t) \propto (1 - \alpha_N)e^{-\lambda_\mathrm{M}(d(\mathbf{v}(\mathbf{x}_i), \mathbf{v}_t))} + \alpha_N, \tag{4.22}$$

where $d(\mathbf{v}(\mathbf{x}_i), \mathbf{v}_t))$ is the distance between two velocities as defined in [33] and α_N is a small constant that represents uniform noise. Finally, to obtain a dense estimation, the set of salient points is convolved with a smoothing kernel. We therefore define the motion likelihood as

$$p(M_t|\mathbf{x}) \propto \frac{1}{K} \sum_{i=1}^{N_\mathrm{s}} p(\mathbf{x}_i|M_t) \Phi_{\mathbf{\Sigma}}(\mathbf{x} - \mathbf{x}_i), \tag{4.23}$$

where $\Phi_{\mathbf{\Sigma}}(\mathbf{x})$ is a Gaussian kernel with covariance $\mathbf{\Sigma}$ and N_s is the number of salient parts. The covariance is estimated automatically from the weighted set of salient points using the multivariate Kernel Density Estimation [121].

Shape: The shape model is an estimate of the object's shape. An approximate object shape at time-step t is defined as an object-centered region P_t, which is calculated by a convex envelope over the parts from the local layer. To maintain the growing capability we dilate the hull by a constant amount of D_S pixels. We define a function $s(\mathbf{x}, S_t) \equiv 1$ if $\mathbf{x} \in S_t$ and 0 otherwise and the shape likelihood model for a pixel at $\mathbf{x}$ is thus defined as

$$p(S_t|\mathbf{x}) \propto s(\mathbf{x}, S_t). \tag{4.24}$$

As mentioned before, (4.19) is used for allocating new parts in the local layer. We do not sample (4.19) directly, but rather discretize it first, by calculating its value for each pixel in the image. This discretized distribution is then used to draw positions for new parts from the potential target region. The process of probability map construction is illustrated on a real example in Figure 4.12. To make sure that parts are allocated only in regions whose likelihood of containing the target is high enough, we set to zero those regions of the discretized distribution, whose value is smaller than the third of the maximal value from $p(\mathbf{x}|C_t, M_t, S_t)$. The regions of the discretized distribution that correspond to existing parts are masked out (set to zero) in order to prevent part duplication.

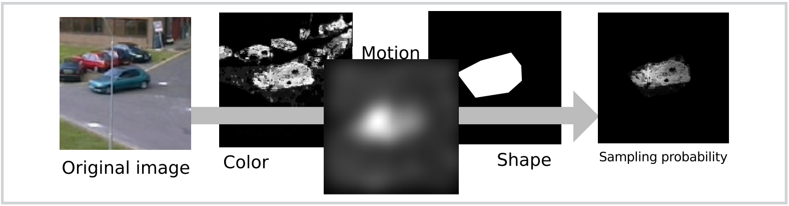


Figure 4.6

Illustration of the cumulative probability map construction.

4.2 Tracking with the coupled-layer appearance model

In this section we present the integration of the coupled-layer appearance model into a visual tracker, which is also denoted as *Local-Global* tracker or LGT tracker in the following text due to the interleaved combination of local and global appearance description. The section describes how the appearance model is initialized in the short-term tracking context and how it interacts with a motion model of a tracker. At the end we present our software implementation of the tracker together with some technical details.

4.2.1 Appearance model initialization

In the first frame, the tracker has to initialize the appearance model using the information it receives as an input. Based on common practice in computer vision, we assume that a rectangular region encompassing the target will be provided, although more detailed information about the region of the object can be used efficiently by the LGT in contrast to many other trackers. Since no other structural information about the object is given in advance, the the local layer is initialized by uniformly placing the parts in a grid pattern within the given rectangular region, as illustrated in Figure 4.7. The weights of the parts are initialized to the same value. The global information in the middle layer is initialized based on the information from the parts from the local layer.

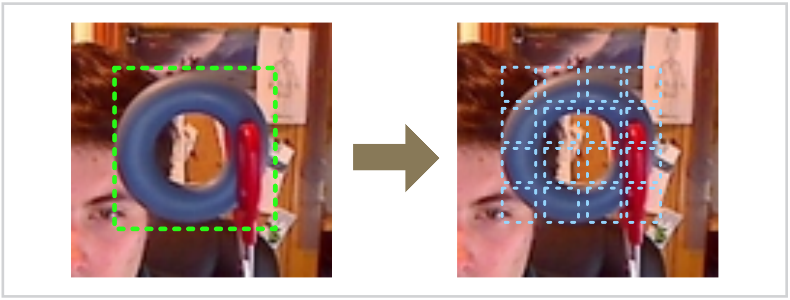


Figure 4.7

An illustration of the local layer initialization for a given initialization region.

4.2.2 Motion model

The proposed tracker also utilizes a motion model that predicts the motion of the object, which is used to provide a better initial estimate when matching the local layer. The coupled-layer appearance model starts from an initial estimate of the target's position and then refines its estimate by adapting to the current image as described in Section 4.1.1. The center of the target can then be identified as a weighted average $\mathbf{c}_t$ of the part's positions. During tracking we can use prediction of the motion of the parts to initialize the matching of the local layer, the better the prediction the faster the part set will converge. For this we utilize a motion model that predicts the motion of the object. We apply a Kalman filter [122] with a nearly-constant velocity (NCV) dynamic model [123] to filter the estimates of the target's center $\mathbf{c}_t$. Thus, at time-step t, the target's velocity $\hat{\mathbf{v}}_t$ estimated by the Kalman filter is used as an initial estimate for the local layer parts. A better initial position does normally (when the motion model correctly predicts the position of the object) result in faster convergence of the optimization algorithm and a more robust solution. The translation transformation also does not change the neighborhood properties of the set.

4.2.3 The tracking loop

We now overview the LGT tracker by iterating through the steps of matching and updating the appearance model during tracking. An list of steps is shown in Figure 4.8. Using a new image from the image sequence we start from an initial position, predicted by the motion model. The object is located by matching the local layer to the new frame to maximally explain the visual data. The global layer is used to identify and remove the parts from the local layer that do not correspond to the target. The good parts parts are used to update the middle layer of the appearance model that is then is used to allocate new parts in the local layer if necessary. Finally, a new position of the object is calculated and used to update the motion model. Additionally, we summarize the sequence of relevant steps of our tracking algorithm in Figure 4.9.

4.2.4 Implementation

The reference implementation of the LGT tracker that was used in the experiments in Section 4.3 was written in a mixture of Matlab and C++ code using the OpenCV library. This implementation is based on the implementation of the LGT tracker that was written for [107] also used in the evaluation and comparison in the next section. Additionally, we have also implemented the LGT tracker purely in C++ within the Legit tracker library that is mentioned in Appendix C.

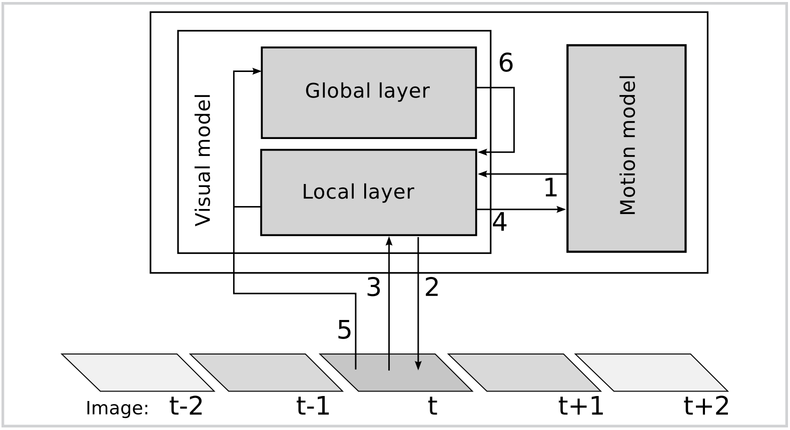


Figure 4.8

A schematic overview of the main steps in processing of a single frame. 1 – prediction from the motion model, 2 – matching the local layer, 3 – updating weights and removing parts, 4 – updating the motion model, 5 – updating the global layer, 6 – adding new parts.

4.3 *Experimental analysis*

In this section we describe the experiments that we have performed in order to evaluate the proposed coupled-layer appearance model and discuss the results. Throughout the experiments we extensively use the evaluation methodology that we have proposed in Chapter 3, predominantly the A-R measure pairs and the A-R plots for visualization. Using the A-R methodology we evaluate different properties of the appearance model on the local and global layer using the sequences from VOT2013 benchmark [100]. We compare these changes to the reference configuration, that was derived from [107] and that is summarized in Table 4.1.

4.3.1 *Resource requirements analysis*

In terms of memory requirements of the coupled-layer appearance model, the size of the bottom layer is directly correlated with the number of parts. Each part stores a fixed size histogram, a position and a weight. The size of the global layer is fixed during tracking and depends on the number of bins in the color histograms and the size of probability maps.

To obtain a realistic distribution of computational requirements of individual parts of the tracker, we have run the tracker on several sequences from VOT2013 dataset and recorded the times for individual parts of the algorithm. From this we have observed that

```
■ Input: A rectangular region that encompasses the object.
■ Initialization: Distribute parts in a regular grid in the region and assign uniform weights,
  initialize global model.
■ For $t = 1, 2, 3 \dots$
    ■ Predict the target's velocity $\hat{\mathbf{v}}_t$ using the Kalman filter and initialize the local-layer
      parts with the NCV model.
    ■ Adapt the local layer parts by maximizing $p(\mathbf{Y}_t, \mathbf{X}_t | \mathbf{X}_{t-1})$ as described in algorithms in Figure 4.4 and Figure 4.5.
    ■ Recalculate the target's center $\mathbf{c}_t$ and update the Kalman filter estimate.
    ■ Identify and remove irrelevant parts.
    ■ Update the global model using the remaining parts.
    ■ Sample new parts from the distribution $p(\mathbf{x} | C_t, M_t, S_t)$.
```

Figure 4.9

The integration of the LGT appearance model in a tracking framework.

the computational complexity of the algorithm depends on the number of parts in the part set, which on average takes about 80% of the total tracker update time for a set of 30 to 40 parts, which is a typical set size on most of the evaluated sequences. Processing of the top layer (updating the representations and generating a joint probability distribution) takes about 15% of the time. The most expensive global modality is motion modality that requires detection of corner features and estimation of optical flow.

On average, the tracker performed at about 2 to 2.5 frames per second on VOT2013 dataset on a AMD Opteron 6238 processor. No explicit parallelization was used in our implementation, although we have observed that more than a single core of a processor was utilized during tracking, most likely because of Matlab implicit parallelization of matrix operations. We also acknowledge that some segments of the algorithm could run in parallel, for example evaluating visual similarity function for parts in each iteration of the optimization algorithm as well as image processing algorithms (histogram backprojection, morphological operations, optical flow calculation, etc.) in the global layer.

4.3.2 *Parameter analysis*

Parameter analysis can provide valuable information about the behavior of a given model for same input data, but with varying configurations of model parameters. Since the pa-

Table 4.1

Reference parameters, according to [107].

Section	Parameters
parts	patch size: 6×6 pixels, histogram bins: 16.
matching	cost function parameters: $\lambda_{\text{v}} = 1, \lambda_{\text{g}} = 0.015$. global optimization: $M_G = 10, S_G = 300, E_G = 10, m_G = 20, r_G = 0.08, s_G = 0.001$. refinment step: $M_L = 5, S_L = 50, E_L = 5, m_L = 5$.
guiding	$\lambda_D = 3, T_D = 40, T_R = 0.1, \alpha_{\text{cap}} = 0.8$.
modalities	color: 16 bin HSV histogram, updating: $\alpha_F = 0.95, \alpha_B = 0.5$. motion: $\lambda_M = 1$. shape: $D_S = 10$.

rameters of a model can be numerous, as it is the case for the proposed coupled-layer appearance model, the combinatorial explosion prohibits detailed study of mutual interactions. However, individual parameters can still be analyzed and can provide a lot of valuable insights. For the analysis of individual parameters of the appearance model we have chosen a subset of VOT2013 dataset that presents a sufficient challenge for the LGT tracker in various aspects. We have selected sequences *bicycle*, *bolt*, *diving*, *gymnastics*, *hand*, *iceskater*, *sunshade*, and *woman*.

All the experiments were performed in the following manner: the tracker was evaluated on a specific sequence for a specific parameter 30 times to account for stochastic processes in cross-entropy optimization and positioning of new parts. The performance scores obtained on individual trials were then averaged together. When averaging the scores between sequences, the length of individual sequences was taken into account. In total the algorithm was run more than 20000 times for the parameter analysis alone, a feat that can only be achieved consistently with good software automation approach. We have used the *TraXtor* tracker development environment that we have already mentioned in Chapter 3 and is described in more details in Appendix C.

Local layer: The first aspect of the appearance model that we have investigated is the matching operation of the local layer. The operation specifies several parameters, related to the cost function and optimization method. The first parameters are the matching cost function constants that regulate the influence of visual similarity and geometrical constraints, i.e. λ_v and λ_g. Since we are only interested in the maximum value of the cost function (4.4), the actual value of the function is not important. The analysis can be therefore focused on the ratio between λ_v and λ_g. We can assume that one of the constants, e.g. λ_v is fixed to 1 and we only observe changes of λ_g, which we call the *rigidity* factor. The higher the λ_g, the more the geometry constraints in the neighborhood of parts are enforced. An A-R plot visualizing performance for various values of λ_g for a set of sequences is shown in the left A-R plot in Figure 4.10. We can see that the performance on many sequences is improved for lower values of parameter λ_g, especially when it comes to non-rigid objects, like *diving*, *hand*, and *iceskater*. On the other hand, the performance is not increased constantly, for extremely low values of λ_g it starts decreasing again, e.g. for sequences *bolt* and *gymnastics*, which means that some degree of geometrical constraints is still required. Another sequence, which is apparently heavily influenced by the rigidity parameter is *sunshade*, where the object (head) is more rigid. A more rigid model would in theory be beneficial for this kind of object, however, the rapid transitions from sun to shade and back, that occur in the sequence, present a rigid constellation with a problem. A lot of parts become unreliable outliers. In a weaker constellation such parts simply drift, but if the constraints are too strong, they also corrupt the overall solution, much like in the case of least-square optimization.

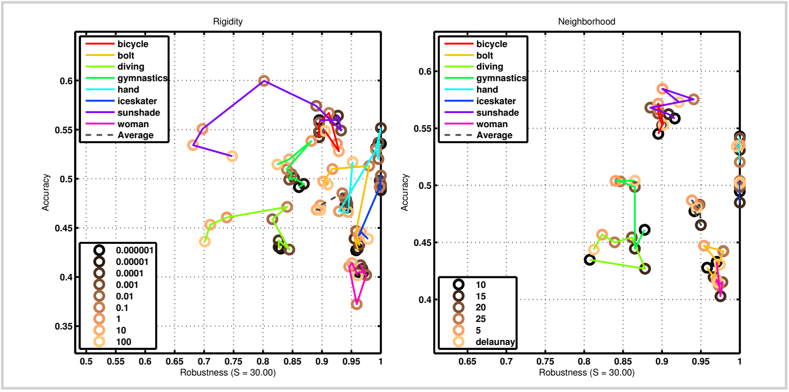


Figure 4.10

Influence of rigidity parameter λ_g (left) and the neighborhood selection technique (right) on tracking performance.

Both stages of optimization are influenced by the choice of initialization parameters m_G, r_G, r_G, and, m_L that set the scope of the search region, and the cross-entropy parameters M_G, S_G, E_G for the global step and M_L, S_L, and E_L for the local step. The performance of the LGT tracker for various values of initialization parameters on test sequences are visualized in A-R plots in Figure 4.13. We can see that there is not a lot of change in performance even if the search space is increased with respect to the reference setup, which means that the cross-entropy algorithm robustly converges even in case of more sparse sampling (the number of samples is kept constant).

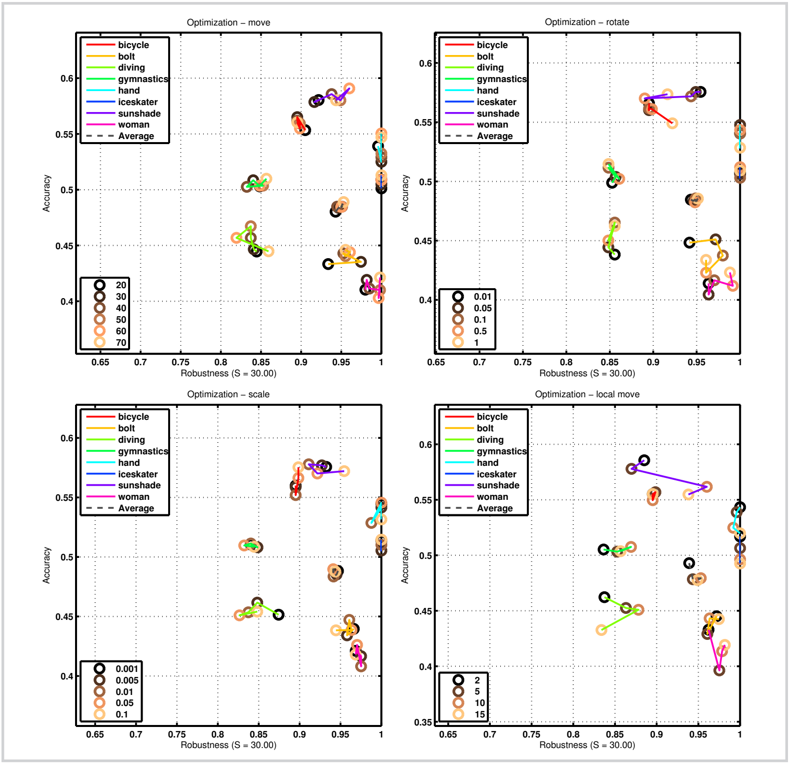


Figure 4.11

Influence of parameters m_G. r_G, s_G and m_L on tracking performance.

Choice of part neighborhood is another aspect that can influence the tracking performance. In Section 4.1 we have proposed two approaches, the first is to use a Delaunay graph, the second is to use the parts within a fixed radius. The results for Delaunay graph and several fixed radii is presented in the right A-R plot in Figure 4.10. We can see that both techniques perform approximately equally well on average. Larger differences can be observed only in some sequences, e.g. *gymnastics* and *diving*, where the object is elongated most of the time and the differences in neighborhoods are more clear. The main advantage of Delaunay triangulation for neighborhood selection is the absence of an explicit parameter, that can be beneficial as in case of the *gymnastics* sequence, or not, as in case of the *diving* sequence.

We have also investigated the type of local appearance models and compared several simple descriptions: 8-bit, 16-bit, 32-bit gray-scale histograms matched using Bhattacharyya distance and gray-scale templates matched using sum-of-square-distances (SSD) and normalized cross-correlation (NCC). Despite our expectations that more discriminative template-based matching could give a better result than histogram-based approaches, the results in Table 4.2 show that the template-based approaches are in fact performing worse. This is likely due to rotation invariance of histogram representations and the fact that an appearance model of a single part does not have to be very discriminative on its own. However, we also acknowledge the fact that the performance of individual descriptions could be improved by adjusting other related parameters, like λ_V.

Table 4.2

Average overlap and average number of failures for different types of local appearance models. Arrows indicate sorting direction.

	8-bit hist	16-bit hist	32-bit hist	SSD	NCC
Overlap ↑	0.48	0.48	0.47	0.46	0.50
Failures ↓	0.58	0.54	0.61	2.95	1.09

Modalities selection: In the global layer, the most important question regarding configuration of coupled-layer appearance model is how do the three global modalities (color, motion and shape) influence the tracking performance. We have performed evaluation on selected sequence with all modalities, only subset of two and subset of one modality. For completeness, we have also added a tracker configuration, where the global layer is completely absent. In this case the tracking is performed by matching of a fixed set of parts that are initialized at the first frame.

The results of the experiment are shown in Figure 4.12 as a series of A-R plot for selected test sequences. We can observe several trends in the plots. The fully inclusive tracker setup is in most cases the best, which confirms that a multi-modal fusion is beneficial for the performance of the appearance model. In many cases this first position is shared with at least one other setup that excludes one modality. In case of sequences *bolt*, *diving*, and *hand* where removing the motion modality results in no apparent degradation, or even in improvement in case of the *hand* sequence. All three aforementioned sequences do not contain a uniform global motion of the object because of the non-rigid properties of the object. In case of sequences *woman*, *bicycle*, and *iceskater*, removing the shape modality does not change the performance, as the object can be clearly separated from the background by the color modality. Removing the color modality from the global layer results in impaired performance on most sequences, which makes it the most important (and most complex) modality. The sequence that stands out in terms of modalities selection is the *sunshade* sequence. Removing any one of the three modalities actually even slightly improves the performance, while removing any two of them decreases it. This shows that the modalities can be used to complement each other. Clearly not all three modalities are beneficial in all tracking scenarios, the usefulness of a certain modality can even change during sequence. A modality selection technique could therefore improve the overall performance.

The last observation of results in Figure 4.12 is related to the case, where the global layer is not used altogether and only a fixed set of parts is used. This is clearly not a sustainable tracking approach as the geometry of the part set can only accommodate a certain level of appearance change and cannot account for the outdated parts. This is confirmed by the results. The only sequence, where the local layer alone is successful is the *gymnastics* sequence, because the object is being followed by the camera all the time.

Color modality: The color modality has some parameters that influence its generalization and adaptation. One of the parameters is the choice of color space. The coupled-layer appearance model in [107] uses a HSV color model that is considered suitable for tracking for its robustness to illumination changes. We have compared the HSV model to a RGB model and found that they both work similarly well. Furthermore we have also evaluated the adaptation rate for the foreground and background histograms and the number of bins. As seen in A-R plots in Figure 4.13, the performance decreases nearly always when the adaptation parameter of either histograms is set to 1, which means that the histogram is acquired at the beginning and does not change after that. If at least some adaptation is allowed, the performance improves significantly in all test sequences, except in the case of the foreground histogram for *diving* sequence. The reason for that

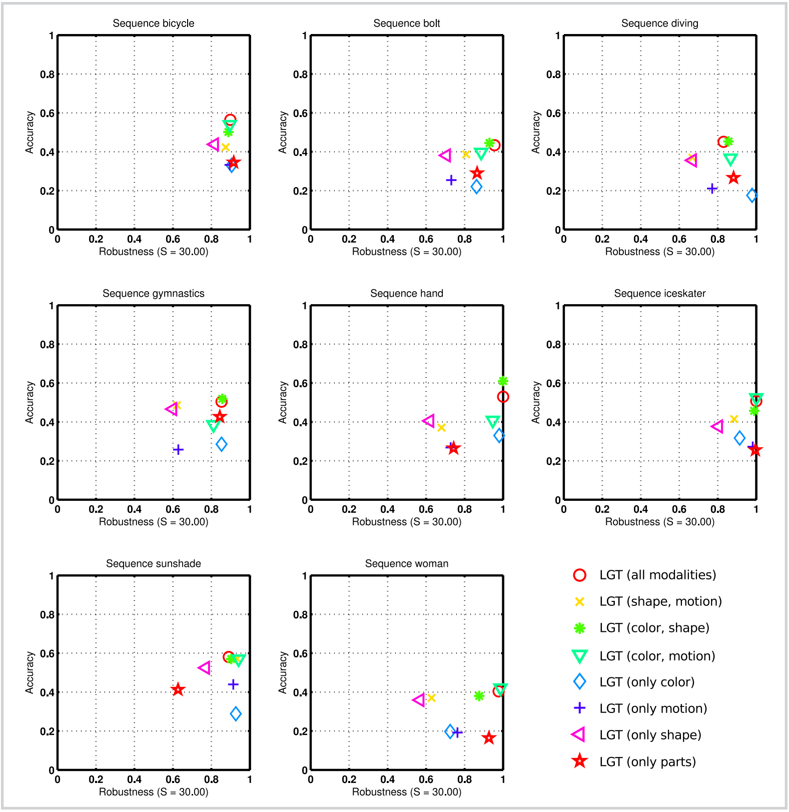


Figure 4.12

Influence of global modality selection on tracking performance.

is that the color of the object does not change in this sequence, while the deformations of the object make it very hard to maintain the correct representation of that color. If the histogram is static it is therefore more robust. The other notable exception is the *sunshade* sequence where the change of the background adaptation parameter does not influence the performance, even if the adaptation is disabled, most likely because the background is static and can be estimated from the first frame well enough.

Motion modality: The only notable parameter in motion modality is λ_M that defines the contribution of individual points of local motion estimation on the sampling proba-

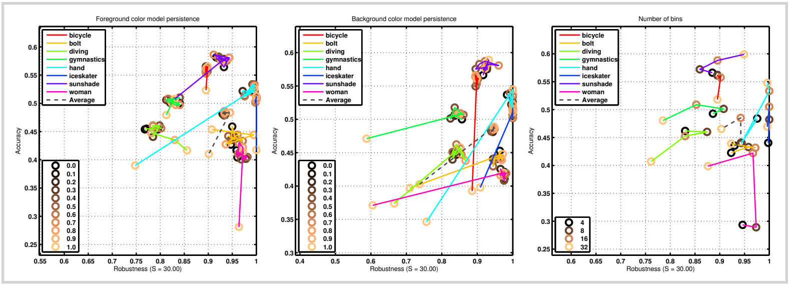


Figure 4.13

Influence of parameters α_F and α_B and the number of bins in the model's histograms on tracking performance.

bility map, as described in (4.22). The results, presented in the left A-R plot in Figure 4.14 show that the parameter does not have a uniform effect over the sequence. Sequences *hand* and *bolt* benefit from lower influence of individual points, while sequences *bicycle* and *sunshade* benefit from higher influence. In the first two sequences the optical flow estimates are likely unreliable due to lack of texture in case of *hand* and small size and frequent appearance changes in case of *bolt*. In case of the second two sequences the motion is actually beneficial because the objects are textured and does not deform a lot.

Shape modality: The parameter D_S controls the expansion of the shape estimation, which directly influences the region where new samples can be positioned. The results, presented in the right A-R plot in Figure 4.14 show that the expansion is beneficial: in case of low expansion the average accuracy of the tracker is reduced. Interestingly, the two sequences that seem least affected by this are *bicycle* and *gymnastics*. While the nature of those sequences is in many aspects different, the visual size of the object in both sequences is reduced. This means that only a few parts have to be added to the local layer to account for size change.

4.3.3 Comparative evaluation

To put the performance of the LGT tracker into perspective, we have conducted a comparative performance evaluation to a set of baseline, as well as current state-of-the-art trackers. Our tracker was evaluated on a recent VOT2013 [100] benchmark, which provides a fully annotated dataset, evaluation protocol and the evaluation toolkit along with the results of a large number of state-of-the-art trackers. The large number of available tested trackers makes the benchmarks one of the largest short-term tracking benchmarks

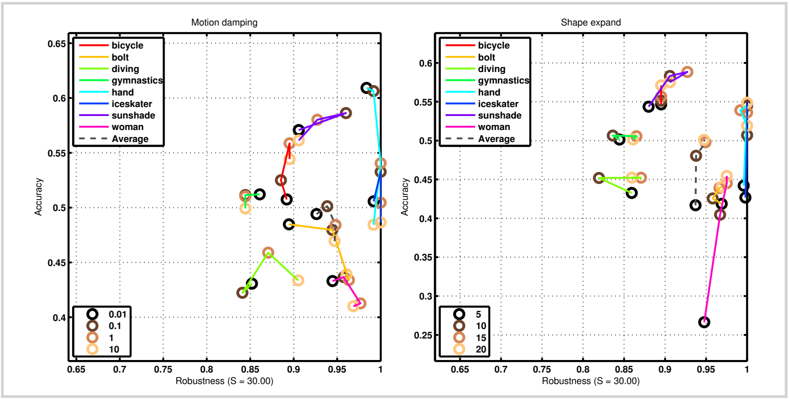


Figure 4.14
Influence of parameters λ_M (left) and D_S (right) on tracking performance.

to date. In the evaluation we follow the official protocol of VOT challenges, the results are summarized in terms of accuracy (average region overlap) and robustness (number of re-initializations). The three experiments in the benchmark (normal sequences, initialization region perturbation, and conversion to gray-scale) were performed using the official VOT toolkit which also provides ranking analysis that takes into account statistical difference on accuracy and robustness performance measures to ensure a fair comparison. The analysis can be performed on per-attribute or per-sequence basis. The details about the methodology are available in [100, 101].

Based on our parameter analysis, presented in Section 4.3.2, we have determined a parameter configuration that improves the original LGT parameter configuration, used in [107]. Note that we have not performed intensive fine-tunning of the algorithm, but only used a subset of sequences to determine very general observations that some constraints, like the scope of global optimization and shape modality expansion parameter, can be relaxed. The new parameter configuration is listed in Table 4.3 The tracker that uses the improved configuration is denoted as LGTi.

The overall VOT2013 benchmark results are visualized in terms A-R ranking plot and A-R plot for the baseline experiment in Figure 4.15, which is also the focus of our discussion. For completeness we also provide ranking results for all three experiments in Table 4.4. From the A-R plots we can see that the LGT tracker family (LGT [107] and LGTi) are one of the most robust trackers in the experiment. In terms of accuracy they

Table 4.3

Improved parameters of the LGTi tracker.

Section	Parameters
parts	patch size: 6×6 pixels, histogram bins: 16.
matching	cost function parameters: $\lambda_\text{v} = 1, \lambda_\text{g} = 0.01$. global optimization: $M_G = 10, S_G = 300, E_G = 10, m_G = 40, r_G = 0.1, s_G = 0.01$. refinment step: $M_L = 5, S_L = 50, E_L = 5, m_L = 10$.
guiding	$\lambda_D = 3, T_D = 40, T_R = 0.1, \alpha_\text{cap} = 0.8$.
modalities	color: 16 bin HSV histogram, updating: $\alpha_F = 0.95, \alpha_B = 0.5$. motion: $\lambda_M = 1$. shape: $D_S = 15$.

perform below average, however, the improved parameter configuration significantly improves the tracker in this respect. In comparison to LGT++ [124] tracker, that is based on the original LGT algorithm, but introduces several modifications that address explicit problems of LGT on VOT2013, the results of LGTi show that an even better performance may be achieved by a good parameter analysis. This also clearly shows the importance of tools that enable consistent automation of such analysis.

The LGTi tracker is outperformed by two trackers, the FoT and PLT. The FoT tracker [64] is a clear example of a tracker that is indeed very accurate, but also fails in many cases. Because of this the tracker gets even more accurate as every re-initialization corrects the scale estimate. On the other hand, the PLT tracker excels in both aspects, most apparently in terms of robustness. The PLT tracker results are hard to comment in detail as its algorithm was never described in any kind of publication. It is known that conceptually the tracker is an extension of the Struck tracker [41] and that it uses color information to weight features used by the SSVM classifier. Interestingly the PLT tracker does not adapt scale, but still achieves good-enough accuracy. This means that scale adaptation may not be required to successfully track an object in many scenarios, which makes the problem easier due to reduced number of parameters that have to be estimated.

The results for two additional experiments in Table 4.4 show that the LGT tracker is very robust even in case of initialization noise. Additionally the LGT tracker performs reasonably well even in case of gray-scale sequences, which is especially interesting, since

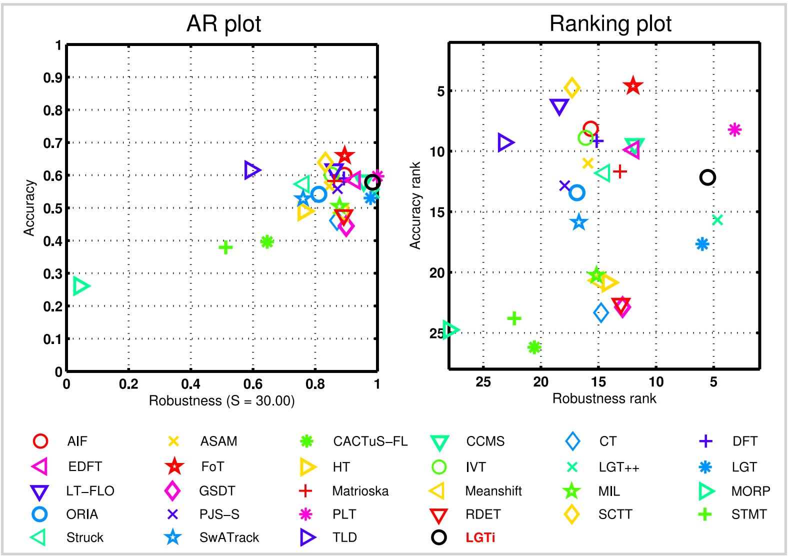


Figure 4.15

Results for VOT2013 benchmark presented as per-attribute ranking A-R plot (left) and per-attribute raw A-R plot (right).

the coupled-layer appearance model utilizes color information on the global layer as the most important global modality, as we have demonstrated in Section 4.3.2.

More detailed results in term of individual per-frame attributes for baseline experiment are presented in Table 4.5. The LGT tracker is very robust in case of illumination change. Taking into account the fact that attributes are not equally represented in the dataset, the hardest attribute is occlusion (there are not many occlusions in the dataset). This is expected as the coupled-layer does not have a mechanism to explicitly handle occlusions and is at the same time prone to adaptations that maximally explain the visual information available.

Looking at the results from per-sequence perspective we can identify sequences that are the most problematic for the coupled-layer appearance model - the results are presented in Table 4.6. In terms of robustness the most problematic sequences are *bicycle* and *diving*, the first one because of the occlusion, the second one because of rapid aspect-ratio changes and foreground-background color similarity. Nevertheless, the tracker excels at many sequences with non-rigid objects like *iceskater*and *hand*, but the main advantage is that the tracker achieves a reasonably low failure count over the entire sequence

Table 4.4

Ranking results for all three experiments in the VOT2013 benchmark. The table shows accuracy rank (A) and robustness rank (R) for all three experiments, as well as average accuracy and robustness rank and the average rank over all three experiments (final column). First, *second and third best values are highlighted. Trackers that did not participate in all three experiments were omitted.*

	baseline		region_noise		grayscale				
	A	R	A	R	A	R	A	R	Rank
PLT	8.21	3.17	5.19	3.67	*5.95*	3.17	6.45	3.33	4.89
FoT [64]	4.61	11.99	*5.42*	11.68	3.25	10.02	4.43	11.23	*7.83*
LGTi	12.16	5.51	8.40	*4.08*	14.69	*5.10*	11.75	*4.90*	8.32
LGT++ [124]	15.67	*4.67*	12.44	4.72	14.97	7.77	14.36	5.72	10.04
EDFT [125]	10.04	11.96	9.63	14.52	6.17	11.58	8.61	12.69	10.65
LT-FLO [77]	6.17	18.40	6.96	15.18	6.83	13.54	6.65	15.71	11.18
CCMS	9.43	11.86	6.96	9.70	11.35	19.27	9.25	13.61	11.43
SCTT	*4.75*	17.30	7.21	17.40	5.97	17.40	*5.98*	17.37	11.67
LGT [107]	17.66	5.97	14.75	5.78	18.40	8.05	16.94	6.60	11.77
Struck [41]	11.78	14.49	12.85	13.55	9.55	11.58	11.39	13.21	12.30
MTR [126]	11.62	13.13	11.56	15.38	8.71	14.03	10.63	14.18	12.40
IVT [48]	8.89	16.12	10.74	16.16	8.80	14.93	9.48	15.74	12.61
AIF [127]	8.15	15.67	10.42	16.25	6.54	19.64	8.37	17.19	12.78
DFT [128]	9.09	15.16	11.64	16.49	12.21	12.24	10.98	14.63	12.80
PJS-S	12.84	17.93	13.33	15.84	10.61	15.05	12.26	16.28	14.27
ORIA	13.42	16.87	15.02	16.84	11.25	14.18	13.23	15.96	14.60
TLD [110]	9.28	23.21	6.74	20.67	9.31	19.52	8.44	21.13	14.79
MIL [42]	20.39	15.18	19.27	13.93	15.66	12.09	18.44	13.73	16.08
RDET [129]	22.57	13.05	20.42	11.80	18.25	10.80	20.41	11.89	16.15
GSDT	22.88	12.91	23.33	13.47	18.10	10.52	21.44	12.30	16.87
HT [130]	20.85	14.19	19.80	13.53	20.75	13.82	20.46	13.85	17.16
CT [49]	23.33	14.77	22.08	13.85	19.50	13.60	21.64	14.07	17.86
Meanshift [32]	20.67	15.03	17.77	17.86	23.00	16.88	20.48	16.59	18.54
STMT	23.81	22.31	22.60	20.50	21.31	17.85	22.57	20.22	21.40
CACTuS-FL	26.19	20.57	25.08	16.38	23.75	19.33	25.01	18.76	21.88

set, while trackers like EDFT and FoT fail several times in sequences where their individual assumptions are violated. Qualitative examples of tracking on VOT2013 sequences of LGT tracker in comparison to two reference tracker can be seen in Figure 4.16.

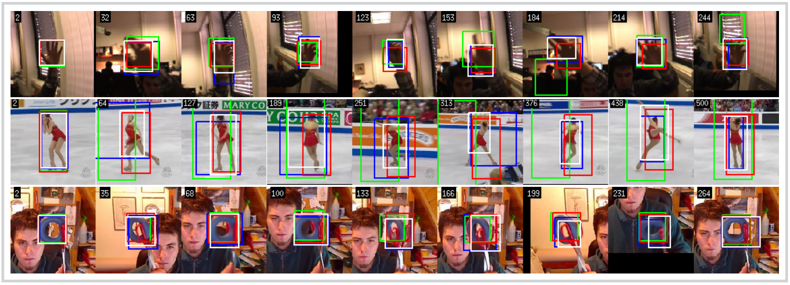


Figure 4.16

Visual comparison of trackers PLT (red), FoT (green), LGT (blue), and LGTi (white) on sequences *hand*, *iceskater*, and *torus* from VOT2013 dataset.

Table 4.5

Results of CCMS, EDFT, FoT, PLT, LGT, and LGTi trackers for *individual attributes* in VOT2013 dataset in terms of average overlap (O) and number of failures (F). First, *second and third best values are highlighted. Arrows indicate sorting direction.*

	CCMS		EDFT [125]		FoT [64]		PLT		LGT [107]		LGTi	
	O ↑	F ↓	O ↑	F ↓	O ↑	F ↓	O ↑	F ↓	O ↑	F ↓	O ↑	F ↓
cam. mot.	*0.59*	6.00	0.58	13.00	0.65	17.00	0.59	0.00	0.53	4.20	0.58	*3.33*
ill. ch.	0.50	2.00	0.64	*1.00*	0.78	0.00	*0.65*	0.00	0.52	0.00	0.61	0.00
occlusion	0.64	2.00	*0.67*	3.00	0.58	7.00	0.72	0.00	0.45	1.13	0.52	*0.73*
size	*0.50*	1.00	0.40	3.00	0.57	7.00	0.44	0.00	0.43	1.07	0.46	*0.40*
motion	0.60	4.00	0.62	5.00	0.71	6.00	0.62	0.00	0.60	1.00	*0.63*	*0.73*
empty	0.72	0.00	0.70	0.00	0.65	0.00	*0.71*	0.00	0.57	0.00	0.59	0.00
Overall	0.58	4.09	0.58	7.61	0.66	10.29	*0.60*	0.00	0.53	2.29	0.58	*1.74*

The main differences between LGT and LGTi in terms of improvement occur in sequences *woman*, *sunshade*, *hand* and *gymnastics*, most likely due to relaxed optimization parameters that account for quicker movement of the object and the increased shape expansion parameter that enables faster recovery. On the other hand the performance is decreased in case of *bolt*, *car*, and *diving* sequences, because of poor initial color histogram estimation which, in combination with larger part sampling region resulted in many parts being initialized on the background.

Table 4.6

Results of CCMS, EDFT, FoT, PLT, LGT, and LGTi trackers for *individual sequences* in VOT2013 dataset in terms of average overlap (O) and number of failures (F). First, *second and third best values are highlighted. Arrows indicate sorting direction.*

	CCMS		EDFT [125]		FoT [64]		PLT		LGT [107]		LGTi	
	O ↑	F ↓	O ↑	F ↓	O ↑	F ↓	O ↑	F ↓	O ↑	F ↓	O ↑	F ↓
bicycle	0.49	*1.00*	0.44	0.00	0.74	*1.00*	0.43	0.00	0.51	*1.00*	*0.56*	*1.00*
bolt	*0.81*	0.00	0.84	1.00	0.43	3.00	0.65	0.00	0.44	*0.13*	0.45	0.53
car	*0.47*	1.00	0.45	1.00	0.56	1.00	0.44	0.00	0.45	0.00	0.42	*0.27*
cup	0.72	0.00	*0.77*	0.00	0.76	0.00	0.81	0.00	0.71	0.00	0.75	0.00
david	0.53	*1.00*	0.68	0.00	0.80	0.00	*0.74*	0.00	0.61	0.00	0.70	0.00
diving	0.39	0.00	0.33	4.00	0.25	3.00	0.35	0.00	*0.43*	*0.87*	0.47	1.13
face	0.74	0.00	0.88	0.00	0.66	*1.00*	*0.87*	0.00	0.59	0.00	0.61	0.00
gymnastics	0.59	0.00	0.56	1.00	0.53	2.00	*0.56*	0.00	0.51	0.80	0.50	*0.40*
hand	*0.65*	0.00	0.51	0.00	0.56	3.00	0.57	0.00	0.63	*0.20*	0.68	0.00
iceskater	0.63	0.00	0.35	*3.00*	0.35	0.00	0.51	0.00	0.58	0.00	*0.61*	0.00
juice	0.63	0.00	0.64	0.00	0.89	0.00	0.64	0.00	*0.80*	0.00	0.78	0.00
jump	0.53	*1.00*	0.60	0.00	0.74	0.00	0.34	0.00	*0.63*	0.00	0.60	0.00
singer	*0.39*	*1.00*	0.37	0.00	0.81	0.00	0.35	0.00	0.21	0.00	0.21	0.00
sunshade	0.72	0.00	*0.65*	3.00	0.62	0.00	0.59	0.00	0.57	0.27	0.64	*0.07*
torus	0.83	0.00	*0.82*	0.00	0.79	0.00	0.79	0.00	0.69	0.00	0.70	0.00
woman	*0.66*	2.00	0.60	1.00	0.62	8.00	0.69	0.00	0.36	1.13	0.51	*0.20*
Overall	*0.61*	0.56	0.60	0.79	0.64	1.54	0.61	0.00	0.54	0.26	0.58	*0.18*

4.3.4 *Non-rigid objects*

To analyze the behavior of the LGT tracker specifically on non-rigid objects we perform have selected five sequences from VOT2013 dataset that include non-rigid objects, i.e. *bolt*, *diving*, *gymnastics*, *hand*, and *iceskater* and performed ranking analysis on this subset. The results are summarized in Figure 4.17, where we can observe that the LGT tracker retains its relative performance in comparison to other trackers. While the global accuracy for all trackers is decreased and many reference trackers, the performance of LGT and LGTi remains similar to the results on the entire VOT2013 dataset, which means that the proposed appearance model can be used to track non-rigid as well as rigid objects. More globally, the performance of less geometrically constrained trackers, like CCMS and SwATrack, becomes apparent. It is also interesting to observe that the difference between LGT and LGTi has decreased on a subset, meaning that the parameter changes in LGTi were more beneficial for tracking more rigid objects.

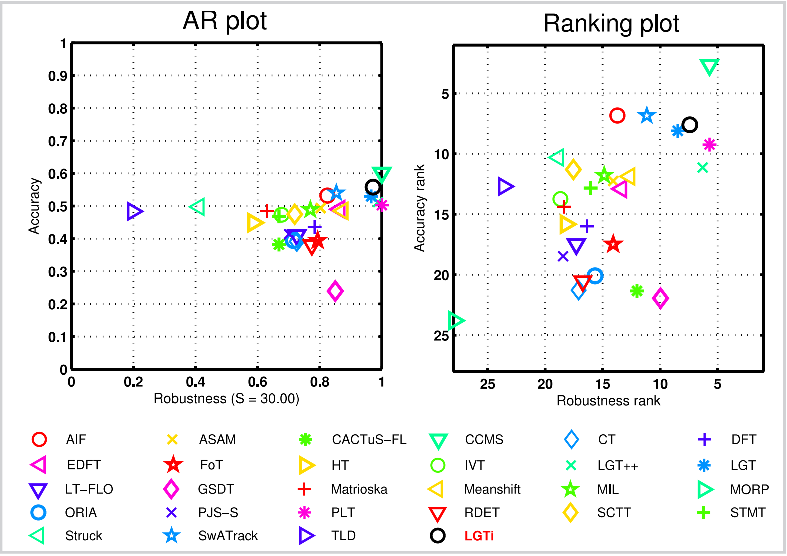


Figure 4.17

Results for subset of VOT2013 benchmark sequences that contain non-rigid objects presented as per-attribute ranking A-R plot (left) and per-attribute raw A-R plot (right).

4.3.5 *Failure cases*

Overall, the proposed visual tracker achieves competitive performance, even in comparison to many visual tracking approached that have been presented recently. The coupled-layer appearance model especially excels in robustness, however, there is plenty of room for improvement. In the previous section we have already mentioned some sequences from the VOT2013 dataset, where the coupled-layer appearance model is unable to adapt correctly and the tracking fails.

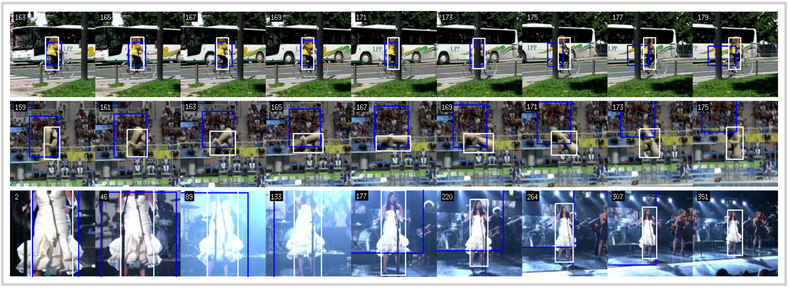


Figure 4.18

Failure cases for LGT tracker (blue), demonstrated on sequences *bicycle*, *diving*, and *singer*. The ground-truth region is shown with white color. In first two cases the strip focuses on behavior before the failure.

In Figure 4.18 we can see the failure that occurs due to occlusion in sequence *bicycle*. The problem of part-based appearance models is that constellation geometry adapts to gradual occlusion and the tracker gets "swept away" by the occluding semaphore pole. In contrast, in sequence *diving* the failure originates from the global layer. In this case the color model gets corrupted because of rapid deformation of the object. Additionally the motion modality is compromised because it assumes globally uniform motion, which is not true in case of rotating target. All this results in global layer slowly corrupting the local layer by positioning new parts on the background. The last failure case in sequence *singer* is not detected, because the target still lies withing the region, predicted by the tracker and the failure criterion is not fulfilled, however, the LGT is clearly not tracking it anymore. This is caused by slow shrinking of the target combined with rapid illumination change that causes the appearance model to start adapting to background.

4.4 *Summary*

In this chapter we have presented a working instance of a hierarchical appearance model that is also called a coupled-layer appearance model because of the collaborative interaction of local and global appearance description. We have presented the required details for the implementation of the proposed appearance model: the visual description of parts that describe the local appearance of the object, an efficient optimization for the matching operation for the set of parts, a set of heuristics for part weight dynamics, and the structure of a multi-modal global appearance model. We have described how the appearance model is integrated in a visual tracker and how it interacts with a motion model of the tracker. In the second part of the chapter we have presented the results of an in-depth analysis of the reference implementation of the tracker, both in terms of parameter analysis and as a comparison to the related work. The experimental analysis has shown that the proposed appearance model is indeed capable of tracking objects robustly with no a-priori knowledge. As predicted, the model excels in many situations with non-rigid deformations. The analysis has also shown some weak-spots of the proposed concept, some of which we will address in the next chapter.

5

Tracking with template anchors

In this chapter we present another appearance model, based on the concept of a hierarchical appearance model. The work is based on the experimental analysis of the coupled-layer appearance model, presented in Chapter 4, which shows that deformable well-designed part-based models exhibit excellent performance in tracking non-rigidly deforming targets, but are usually outperformed by holistic models when the target does not deform or in the presence of uncertain visual data. The reason for that is that part-based models deform freely to match the visual data and account for outdated or occluded parts. The uncertainty of the visual data thus introduces potentially small errors in the large number of parameters that have to be estimated in comparison to holistic models, which leads to poor position estimate. Even if the target is deforming non-rigidly, a low-parameter holistic model might lead to a smaller position error in a short run than the part-based models that would over-fit the uncertain visual data. Still, in presence of low visual uncertainty, the deformable part models typically outperform the holistic models.

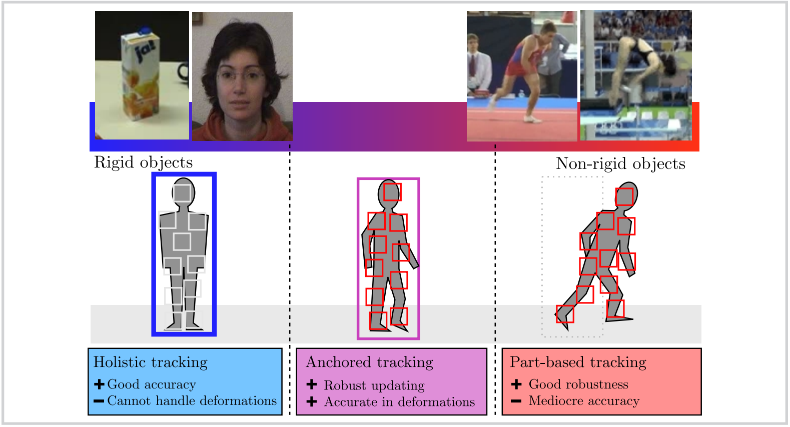


Figure 5.1
An illustration of the proposed idea of combining holistic appearance model and a part-based appearance model by switching between them.

We address the problem of self-supervised estimation of a large number of parameters by proposing a new hierarchical appearance model composed of three layers, each describing the target at a different level of detail. The level of detail varies in the type of features used, the number of parameters estimated in the layer and the aggressiveness of the adaptation. In particular, the three layers used are: part-based, holistic coarse and holistic detailed. Conceptually, the function of the bottom two layers is similar to the local and global layer in the coupled-layer appearance model. The top layer is designed as a memory system of holistic templates that constrains the lower layers if a memorized template is detected reliably. All three layers interact during target localization and, depending on the visual uncertainty, can be used for mutually supervised updating by accounting for the potential uncertainty of the visual information. This makes the appearance model shift between purely holistic and part-based behavior, depending on the visual uncertainty, as illustrated in Figure 5.1. A new tracker is proposed based on this model which exhibits the qualities of part-based as well as holistic models.

In Section 5.1 we present the details about the proposed appearance model. In Section 5.2 we describe the integration of the appearance model in a tracking framework. Section 5.3 contains an in-depth evaluation of the resulting tracker, both in terms of parameter analysis and as a comparison to the related work that confirms that the the constraints of the third layer indeed improve performance of the tracker. We conclude the chapter with a summary of our work in Section 5.4.

5.1 *The anchored appearance model*

The proposed appearance model is formalized as a hierarchical three-level set of layers in which the state at time-step t is specified by a state of the bottom part-based layer $\mathcal{P}_t$ (Section 5.1.1), middle color-based segmentation layer $\mathcal{S}_t$ (Section 5.1.2) and the top template-based layer $\mathcal{T}_t$ (Section 5.1.3), i.e.,

$$\mathcal{V}_t = \{\mathcal{P}_t, \mathcal{S}_t, \mathcal{T}_t\}. \tag{5.1}$$

The appearance model is illustrated in Figure 5.2. The bottom layer is a geometrical constellation of parts that describe the target's local visual properties. The middle layer provides mask estimates for the entire object by using color segmentation. The top layer is a long-term memory that stores static instances of object appearance and uses them to help lower layers recover. Since the top-layer templates act as anchors that strongly influence the lower layers, we refer to the presented model as an *anchored appearance model.*

5.1.1 *The part-based layer*

Definitions: The representation of the bottom layer is a geometrically constrained constellation of local parts. We use the the same formalization as in the local layer for the

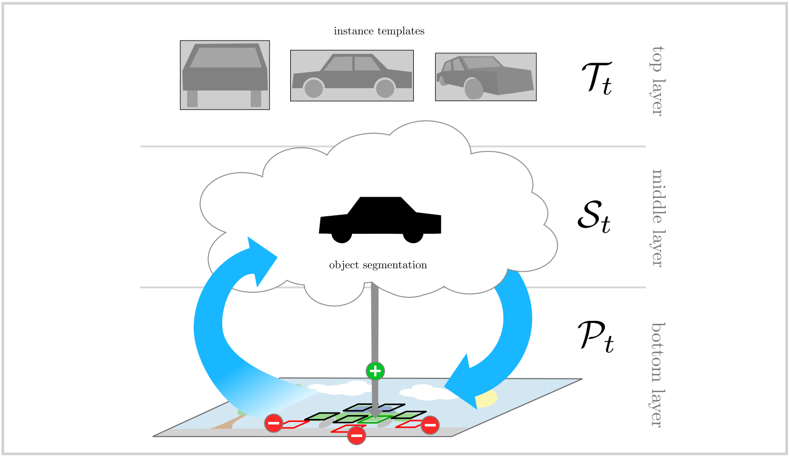


Figure 5.2

Illustration of the anchored appearance model.

coupled-layer appearance model, described in Section 4.1.1, however, the matching and update process are different. The state, $\mathcal{P}_t$, of the the layer at time-step t is described as:

$$\mathcal{P}_t = \{\langle \mathbf{x}_t^{(i)}, \mathbf{z}^{(i)}, w_t^{(i)} \rangle\}_{i=1:N_t}, \tag{5.2}$$

where $\mathbf{x}_t^{(i)}$ and $\mathbf{z}^{(i)}$ are coordinates and appearance model of the i-th part, respectively, and $w_t^{(i)}$ is a weight that reflects belief that the target is well-represented by that part.

Matching: In contrast to the coupled-layer appearance model from Section 4.1.1 that uses stochastic optimization to optimize a joint cost function over all the positions of parts, the anchored appearance model uses a three-stage deterministic matching approach that exploits the fact that the position of some parts can be determined reliably using optical flow. Using these estimates the matching of parts, for which optical flow cannot be used, is more constrained and can be quickly refined using iterative optimization. An added benefit of this multi-stage approach is that that the computationally demanding evaluation of visual similarity function can be avoided in many cases which results in lower computational requirements.

At each frame we start from an initial estimate from the previous frame, $\mathbf{X}_{t-1}$, and the set of current image measurements $\mathbf{Y}_t$, and seek the value of $\mathbf{X}_t$ that maximizes the joint probability $p(\mathbf{X}_t|\mathbf{Y}_t, \mathbf{X}_{t-1})$. The final state $\mathbf{X}_t$ is estimated by optimizing the probability of constellation state $\mathbf{X}_t$ conditioned on the measurements $\mathbf{Y}_t$, estimation from the previous time-step $\mathbf{X}_{t-1}$, and the displacement-prior certainties $\mathbf{C}_t = \{c_t^{(i)}\}_{i=1:N}$, i.e.,

$$p(\mathbf{X}_t|\mathbf{Y}_t, \mathbf{X}_{t-1}, \mathbf{C}_t) \propto \prod_{i=1}^{N_t} p(\mathbf{X}_t|\mathbf{Y}_t, \mathbf{X}_{t-1}, c_t^{(i)}, \mathbf{z}_t^{(i)}). \tag{5.3}$$

Assuming that parts are only linked between themselves through their neighbors, we again introduce a concept of *neighborhood* as a subset of parts that are directly connected to a certain part and simplify right-hand side of (5.3) into

$$p(\mathbf{x}_t^{(i)}|\mathbf{Y}_t, \mathbf{X}_{t-1}, c_t^{(i)}, \mathbf{z}^{(i)}) \propto p(\mathbf{x}_t^{(i)}|\mathbf{Y}_t, \varepsilon_t^{(i)}, \mathbf{z}^{(i)}), \tag{5.4}$$

where $\varepsilon_t^{(i)}$ denotes the set of the i-th part's local neighbor parts. Assuming the independence of geometrical constraints of individual parts and appearance similarity the distribution in the right-hand side of (5.4) can be further decomposed as

$$p(\mathbf{x}_t^{(i)}|\mathbf{Y}_t, \varepsilon_t^{(i)}, c_t^{(i)}, \mathbf{z}^{(i)}) = \begin{cases} p(\mathbf{Y}_t|\mathbf{x}_t^{(i)}, \mathbf{z}^{(i)})p(\mathbf{x}_t^{(i)}|\varepsilon_t^{(i)}) & \text{when } c_t^{(i)} = 0 \\ p(\mathbf{x}_t^{(i)}|\mathbf{v}_t^{(i)}) & \text{when } c_t^{(i)} = 1 \end{cases}, \tag{5.5}$$

where $p(\mathbf{x}_t^{(i)}|\mathbf{v}_t^{(i)}) = \delta(\mathbf{x}_t^{(i)}|\mathbf{v}_t^{(i)})$ is a Dirac-delta positioned at the flow displacement $\mathbf{v}_t^{(i)}$. In case the optical flow was not estimated reliably for a part ($c_t^{(i)} = 0$), the probability of position of individual part is defined as a product of visual similarity and geometric constraint functions. The visual similarity is defined as

$$p(\mathbf{Y}_t|\mathbf{x}_t^{(i)}, \mathbf{z}^{(i)}) \propto e^{-\lambda_v \rho(\mathbf{z}^{(i)}, \mathbf{z}(\mathbf{x}_t^{(i)}))}, \tag{5.6}$$

where $\rho(\cdot, \cdot)$ is the visual distance between the appearance model of part ($\mathbf{z}^{(i)}$) and extracted visual information, as defined in (4.2), and λ_v is a constant. The geometric constraint is defined as

$$p(\mathbf{x}_t^{(i)}|\epsilon_t^{(i)}) = e^{-\lambda_c ||\mathbf{x}_t^{(i)} - \tilde{\mathbf{x}}_t^{(i)}||^2}, \tag{5.7}$$

where $\tilde{\mathbf{x}}_t^{(i)}$ is a position predicted by the neighboring parts and λ_c is a constant. This prediction is obtained by computing a similarity transform between the neighbors $\epsilon_t^{(i)}$ from $\mathbf{X}_{t-1}$ and the current positions of the neighbors.

Similarly to the two-stage stochastic optimization, presented in Section 4.1.1, we split the the optimization into three steps: (a) optical flow calculation, (b) global alignment step, and (c) a refinement step. As shown in Figure 5.3, the first step of our matching approach calculates the candidate displacement of each part $\mathbf{v}_t^{(i)}$ using the Lucas-Kanade optical flow [119]. The new position of some parts can be estimated reliably this way, but this may not be true for all parts. Therefore the displacement-prior certainty $c_t^{(i)} \in [0,1]$ of this estimation is calculated by forward-backward optical flow criterion based on [64]. The aim of a second stage of the optimization is to find good initial position for remaining parts that cannot be updated using optical flow. For this we employ a Generalized Hough Transform [131] voting scheme, where the voting function of individual parts for a global translation vector is defined as

$$p(\mathbf{x}_t^{(i)}|\mathbf{Y}_t, c_t^{(i)}, \mathbf{z}^{(i)}) = \begin{cases} p(\mathbf{Y}_t|\mathbf{x}_t^{(i)}, \mathbf{z}^{(i)}) & \text{when } c_t^{(i)} = 0 \\ \mathcal{N}(\mathbf{x}_t^{(i)}|\mathbf{v}_t^{(i)}, \sigma_V) & \text{when } c_t^{(i)} = 1 \end{cases}, \tag{5.8}$$

where $\mathcal{N}(\mathbf{x}_t^{(i)}|\mathbf{v}_t^{(i)}, \sigma_V)$ denotes a normal distribution with mean in $\mathbf{v}_t^{(i)}$ and standard deviation σ_V. The global displacement $\mathbf{T}_t$ is then determined as maximization of sum of (5.8), i.e.

$$\mathbf{T}_t = \arg\max_{\mathbf{T}} \sum p(\mathbf{x}_t^{(i)} + \mathbf{T}|\mathbf{Y}_t, c_t^{(i)}, \mathbf{z}^{(i)}).$$

The MAP estimate of (4.4) is then refined by the Iterated Conditional Modes (ICM) algorithm [132], which iterates over the parts and for each part computes a new position as the expected position under the conditional $p(\mathbf{Y}_t, \mathbf{x}_t^{(i)}|\mathbf{X}_{t-1})$. Note that the algorithm only has to iterate over the parts whose displacement certainty $c_t^{(i)}$ is zero. The new part position at each iteration is computed by expected position over $p(\mathbf{Y}_t, \mathbf{x}_t^{(i)}|\mathbf{X}_{t-1}, \mathbf{z}^{(i)})$. At each iteration of the ICM, only the Gaussian predicted by the neighbors $p(\mathbf{x}_t^{(i)}|\epsilon_t^{(i)})$ is re-computed and multiplied with the visual likelihood. The optimization typically converges in less than ten iterations. This makes the MAP optimization of (4.4) gracefully shifting between flock-of-features estimation during confident period and fully constellation-constrained optimization in the other extreme. The object region can be estimated according to bottom layer as the smallest axis-aligned rectangle containing all parts.

Update: The parts in the bottom layer are added and removed to faithfully reflect the appearance changes. A region with high part-density is estimated and parts outside the

```
■ Input: The set of parts $\mathbf{X}_{t-1}$.
■ Initialization: For each part estimate optical flow $\mathbf{v}_t^{(i)}$ and determine its suitability $c_t^{(i)} \in \{0,1\}$.
■ Find the global translation $\mathbf{T}_t$ using Generalized Hough Transform [131].
■ Apply $\mathbf{T}_t$ to all parts where $c_t^{(i)} = 0$.
■ For $j = 1 \dots M_L$:
    ■ For each part $\mathbf{x}_t^{(i)}$ where $c_t^{(i)} = 0$:
        ■ Determine similarity transform based on $\epsilon_t^{(i)})$ and use it to determine $\tilde{\mathbf{x}}_t^{(i)}$.
        ■ Calculate new $\mathbf{x}_t^{(i)}$ as the expected position of $p(\mathbf{x}_t^{(i)}|\mathbf{Y}_t, \mathbf{X}_{t-1}, \mathbf{z}^{(i)})$.
■ Output: The refined set of parts $\mathbf{X}_t$.
```

Figure 5.3

Part-set optimization algorithm.

region are removed. The region is estimated by applying a mean shift mode detection on a kernel density estimate with a uniform kernel on the part locations. In particular, the target bounding box estimated at the previous time step is used for the kernel. For illustration of the algorithm, see Figure 5.4.

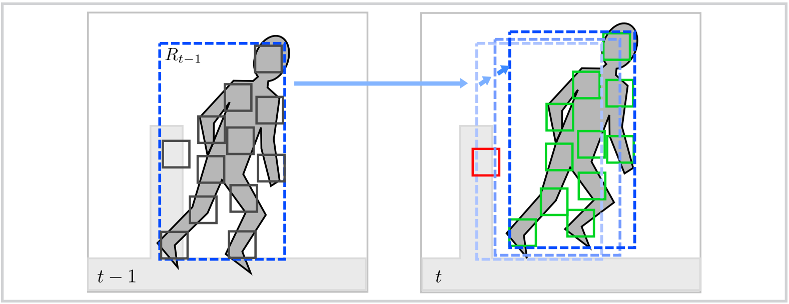


Figure 5.4

Illustration of outlier detection algorithm. The region from frame $t - 1$ is used in frame t to detect an outlier (red color) by converging to inliers with higher density (green color).

New parts are added to the set to increase the object coverage by using the target segmentation mask $S_t(\mathbf{x})$ provided by the middle layer (see Section 5.1.2). To balance between uniform coverage and placing the patches at positions with a high likelihood of certain flow estimation, the following quality score is used for part sampling:

$$q(\mathbf{x}) = H(\mathbf{x}) + \alpha_U U(\mathbf{x}), \tag{5.9}$$

where $H(\mathbf{x})$ is the Harris corner score at position $\mathbf{x}$, $U(\mathbf{x})$ is a periodic function[1] that enforces uniform coverage of sampling points in homogeneous regions and α_U is a mixing constant. A non-maxima suppression is applied to $q(\mathbf{x})$ and the local maxima at locations of existing parts or outside the segmentation mask $S_t(\mathbf{x})$ are removed. The remaining maxima are ordered according to the color similarity likelihood (middle layer) and at most N_U new parts are added at every time-step to enforce a gradual adaptation.

5.1.2 The segmentation layer

In comparison to the global layer of the coupled-layer appearance model, the middle layer of the anchored visual maintains only a global color model of the target and the immediate neighborhood described by a foreground and background RGB histograms, i.e., $\mathcal{S}_t = (\mathbf{h}_t^F, \mathbf{h}_t^B)$. These models are used to construct a coarse segmentation mask of a current region of interest. This mask can then be used for two purposes: (a) for sampling new parts at the bottom layer and (b) proposing the object region for top layer updates.

Given an estimate of the target bounding box, R_S, the segmentation mask $S_t(\mathbf{x})$ is estimated as follows. The initial region R_S is expanded by α_S to account for the scale uncertainty. Foreground and background histograms are backprojected into the expanded region resulting in two backprojection maps, which are further smoothed by a Gaussian kernel to enforce spatial coherence, resulting in foreground and background probability maps, $p(\mathbf{x}|\mathrm{F}_t)$ and $p(\mathbf{x}|\mathrm{B}_t)$, respectively. The foreground posterior is calculated at each pixel using the Bayes rule

$$p(\mathcal{F}_t|\mathbf{x}) = \frac{p(\mathcal{F})p(\mathbf{x}|\mathrm{F}_t)}{p(\mathcal{F})p(\mathbf{x}|\mathrm{F}_t) + (1 - p(\mathcal{F}))p(\mathbf{x}|\mathrm{B}_t)}, \tag{5.10}$$

where $p(\mathcal{F})$ denotes the object prior. A likelihood threshold is estimated such that the ratio between the number of pixels exceeding this threshold within the region R_S and the expanded region is greater than a high ratio λ_S. If such a threshold cannot be set, the discrimination between foreground and background cannot be determined a sufficiently high confidence and the values in the color-based map are set to zero. To reduce

[1]In our case we use a two-dimensional cosine signal that produces a grid pattern.

the probability of initializing new parts on the background, this map is further post-processed by removing all connected components fully outside R_S. The segmentation process is illustrated in Figure 5.5.

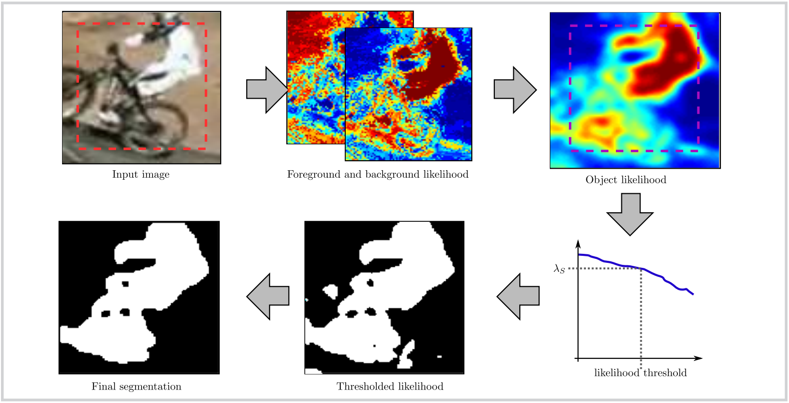


Figure 5.5

Segmentation mask $S_t(\mathbf{x})$ construction. Object likelihood at each pixels is estimated by histogram backprojection, followed by an automatic threshold selection using cumulative likelihood histograms and morphological post-processing.

For updating both histograms we use the same approach as in the coupled-layer appearance model. After the matching operation in the bottom layer is completed, a histogram $\hat{\mathbf{h}}_t^F$ is extracted in the current image from the regions that correspond to the parts of the local layer. The background histogram $\hat{\mathbf{h}}_t^B$ is extracted from a ring-shaped region defined by the convex hull of the parts in the local layer. These histograms are used to update the global color model by an autoregressive scheme

$$\begin{aligned} \mathbf{h}_{t+1}^F &= \alpha_F \mathbf{h}_t^F + (1-\alpha_F)\hat{\mathbf{h}}_t^F \\ \mathbf{h}_{t+1}^B &= \alpha_B \mathbf{h}_t^B + (1-\alpha_B)\hat{\mathbf{h}}_t^B, \end{aligned} \tag{5.11}$$

where α_F and α_B determine the rate of adaptation.

5.1.3 The memory system layer

In case of longer sequences the interaction between a part-based bottom layer and a generative middle layer is not enough. The global appearance information in the middle layer can be slowly corrupted and the local layer is unable to recover. Another layer, even more conservative in adaptation, is needed that is able to correct the lower layers when needed. This layer is inspired by a *memory system* concept that is used successfully in long-term tracking. The long-term tracking field considers objects that can leave the field of view for indefinite amount of time and have to be re-detected upon re-entering the scene. Such scenarios are extremely drift-prone and *instance-based memory systems* have been largely used to mitigate this problem [21, 22]. The idea of a memory system is that multiple snap-shot instances of the object are stored during tracking. Such instances are not adapted, but rather removed from the set when the memory saturates. New instances are added to the memory very conservatively to prevent corrupting the representation and keeping low false-positive rate at detection. Kalal et al. [22] combine a short-term tracker based on optical-flow features [64] with a template memory system. A set of foreground and background templates is kept and updated depending on agreement of the short-term tracker and detector. Similar approach is applied by Pernici et al. [21] who represent the object by a redundant set of keypoints and maintain a short-term representation of the object surrounding context to improve the accuracy of detection.

In the anchored appearance model we use a very simple memory system, presented as a set of static appearance templates of object's appearance acquired at different points in time

$$\mathcal{T}_t = \{T_1, T_2, \ldots\}.$$

Instances can be of any form of an appearance representation that can be used to find a response in an image. They can be as simple as an image patch representation of an object, but also much more complex, e.g. a discriminative detector, or a set of keypoints. The fact that we have multiple instances implies that a single instance does not have to account for all the variability of the object's appearance and can instead focus on a particular object pose and viewpoint. On the other hand, some appearance generalization is still required, otherwise the memory is too specific and cannot guide the lower layers. In Section 5.3.2 we discuss two simple representations that fit this description, a grayscale template, matched using normalized cross-correlation (NCC) and kernelized correlation filters with HOG (histogram of gradients) features [38]. At each frame, all templates are matched within a search region and a candidate with the maximal response is taken as the output.

The set of templates is updated in two stages conservatively by adding new templates only after they have been verified to consistently match the output of the lower and middle layers. A candidate template is sampled from a region proposed by the middle layer only if this region overlaps significantly with the region predicted by the bottom layer.

This template is added to the list of potential templates. A potential template is promoted into the set of active templates only after the overlap with the output region exceeds a predefined threshold λ_T for Ω_T frames.

5.2 *Tracking with the anchored appearance model*

In this section we present the integration of the anchored appearance model into a visual tracker, which is also denoted as *Anchored Templates* tracker or ANT tracker in the following text due to the important role of instance templates in the appearance model in comparison to our previous work. The section describes how the appearance model is initialized in the short-term tracking context and how it interacts with a motion model of a tracker. At the end we present our software implementation of the tracker together with some technical details.

5.2.1 *Appearance model initialization*

The only supervised example of the target is provided by the initialization bounding box. A top layer is initialized by extracting a template from the region, the middle layer is initialized by extracting the foreground and background histograms from within and outside the region. The segmentation from the middle layer is used to sample parts at the bottom layer and automatically determining the number of parts for sufficient object coverage.

5.2.2 *The tracking loop*

We now overview the ANT tracker by iterating through the steps of matching and updating the appearance model during tracking. Figure 5.6 overviews interaction of layers of the appearance model during tracking: A tracking iteration starts by initializing the tracker at a location predicted by a motion model. The part-based model is deformed to match the detailed appearance change and the holistic templates are matched to the image (a). Depending on the strength of the match, these templates can either provide a complete detection of the object or merely focus and guide the update process of the middle and bottom layers (b). The middle layer generates the object segmentation mask (c). The bottom layer is updated by removing and adding patches using the outputs from the top and middle layers (d). The middle layer is updated next, and if the middle and bottom layer outputs agree, a new template is considered for addition to the top-layer template set (d). Finally, the motion model is updated using the estimated bounding box (e).

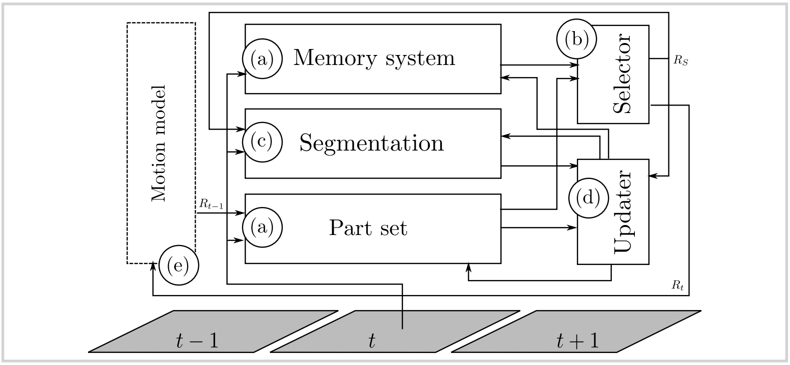


Figure 5.6

The overview of the ANT tracker and its update process.

The confidence of layers varies during tracking, which is reflected in the three modes of tracking operation: (i) detection, (ii) guiding, (iii) free. These modes primarily depend on the matching score of the top-layer template, i.e. the anchor. If the template score exceeds λ_G, the tracker enters a *detection* mode. The region provided by the template is considered as the target output region and is used for updating the middle and the bottom layer. If the template score is lower than λ_G, but exceeds a lower threshold λ_D, then the tracker enters a *guided* mode. The region provided from the template is used in combination with the middle layer only for adding new parts, but the bottom layer is used to estimate the output bounding box. If the template score is below the low threshold λ_D, the tracker enters a *free* mode. In this mode, the bottom layer estimates the output bounding box, which is also used for updating the middle layer and part allocation in the bottom layer.

5.2.3 Motion model

The proposed tracker also utilizes a motion model that predicts the bounding box of the object, which is used to provide a better initial estimate when matching the appearance model to a new image. At each time-step, the initial bounding box is estimated by a Kalman filter [122] with a nearly-constant-velocity (NCV) motion model for position and random-walk (RW) motion model for size. The appearance model is fitted to the image. We can use prediction of the object's position to provide a prior for the matching of the part set. The output region from the multi-layer appearance model is then used to update the Kalman filter estimate.

5.2.4 Implementation

A prototype of our tracker is implemented in a combination of Matlab and C++. Despite the fact that some calculations are performed several times for implementation clarity the implementation performs at about five frames per second. Based on our previous experience of porting the LGT tracker Matlab implementation to native code, we believe that a native implementation of the ANT tracker would run in real-time on an average modern computer as well.

5.3 Experimental analysis

In this section we describe the experiments that we have performed in order to evaluate the anchored appearance model and discuss the results. Throughout the experiments we extensively use the evaluation methodology that we have proposed in Chapter 3, predominantly the A-R measure pairs and the A-R plots for visualization. Using the A-R methodology we evaluate different properties of the appearance model using the sequences from VOT2013 benchmark [100].

5.3.1 Resource requirements analysis

In terms of memory requirements, the bottom two layers of the anchored appearance model have approximately the same memory footprint as the coupled-layer appearance model. The growing of the top layer is unconstrained, therefore a removal scheme is required in practice to maintain a finite memory consumption as well as low computational time. A random selection removal scheme is commonly employed in memory systems if the set is large enough. In case of smaller memory systems a strategy like least-recently-used (LRU) or first-in-first-out (FIFO) may also be employed.

In terms of computational requirements, the two critical parts are the matching of the part set and the matching of the anchor templates which are both dependent on the number of elements. The time required to match a set of parts is reduced by using optical flow to quickly determine the new position of individual parts, however, the number of confident estimations depends on the local properties in the image, so the time of the estimation may vary between the time of optical flow calculation for N parts and the time for optical flow calculation plus the time for evaluation of visual similarity function for N parts. To obtain a realistic distribution of computational requirements of individual parts of the tracker, we have run the tracker on several sequences from VOT2013 dataset and recorded the times for individual parts of the algorithm. On average, optical flow reduces the number of number of visual similarity function evaluations for about 80%. This accounts for about 8% of the entire time to process a frame, in comparison to matching of anchor templates that accounts for about 75% of time. The remaining time is distributed between color segmentation and other tasks.

The ANT tracker performed at about 5.5 to 6 frames per second on VOT2014 dataset on a AMD Opteron 6238 processor, which means that the implementation of the ANT tracker is about twice as fast as the implementation of the LGT tracker. This can be attributed mostly to using optical-flow in the part-set matching process that reduces the number of expensive visual similarity function evaluations. No explicit parallelization was used in our implementation, although we have observed that more than a single core of a processor was utilized during tracking, most likely because of Matlab implicit parallelization of matrix operations. We believe that a large part of the algorithm could be explicitly parallelized, for example evaluation of visual similarity function and matching of anchor templates, which contribute significantly to the overall computational requirements.

5.3.2 Parameter analysis

We have performed several experiments in order to evaluate the properties of the anchored appearance model. Again, we extensively employ the evaluation methodology that we have proposed in Chapter 3, predominantly the A-R measure pairs and the A-R plots for visualization. Using the A-R methodology we evaluate different parts and parameters of the appearance model using the sequences from VOT2014 benchmark [101]. We compare these changes to the reference tracker configuration, whose parameters are summarized in Table 5.1. Since the tracker is fully deterministic it was only evaluated once on a specific sequence for a specific parameter combination. For the analysis of individual parameters of the appearance model we have chosen a subset of VOT2014 dataset that presents a sufficient challenge for the ANT tracker in various aspects. We have selected sequences *ball*, *car*, *drunk*, *fernando*, *fish2*, *hand2*, *motocross*, and *tunnel*. In our preliminary experiments we have determined that a lot of findings about the coupled-layer appearance model, presented in Section 4.3.2 (e.g. the color histogram update rate)

Table 5.1

Reference parameters for the ANT tracker.

Section	Parameters
parts	patch size: 6×6 pixels, histogram bins: 16.
matching	$\lambda_\mathrm{v} = 3, \lambda_\mathrm{c} = 3, \sigma_V = 0.001$.
guiding	$\alpha_U = 10^{-4}, N_U = 2$.
segmentation	32 bin RGB histogram, $p(\mathcal{F}) = 0.4$ updating: $\alpha_F = 0.95, \alpha_B = 0.5, \lambda_S = 0.9$.
memory	$\Omega_T = 7, \lambda_T = 0.8$
mode	$\lambda_D = 0.85, \lambda_G = 0.5$

also apply to the anchored appearance model. In this section we therefore focus on the properties of the top layer and the general properties of the appearance model.

Template type: We have evaluated two types of templates, used in top layer to model instance appearance representation of the object - a gray-scale template, matched using normalized cross-correlation (NCC) and kernelized correlation filters (KCF) with gray-scale and HOG features [38]. In case of KCF templates we have used the same sets of parameters as proposed in [38], but note that the correlation filter templates are used in different way than in case of the tracker, proposed in [38], where a single template is updated over time. From the results for five different configurations that are presented in Table 5.2 we can observe that correlation filters using HOG features are best suited for our use. Normal gray-scale templates are less much less robust because they do not generalize the appearance and become redundant even at minimal changes in the image. The HOG features are sensitive only to image gradients at a lower spatial resolution (topically blocks of multiple pixels are used to construct a histogram), which goes in accordance with our requirement that the instance templates should be robust to some level of image variability.

Mode switching: Parameters that have the largest impact on model's behavior are λ_D and λ_G. This is expected as they directly influence the selection of the tracking mode. Because of this importance to the entire idea of the anchored appearance model, we have analyzed the joint effect of both parameters by measuring the performance on different

Table 5.2

Average overlap and average number of failures for different types of instance templates. Arrows indicate sorting direction.

	NCC	KCF-gray	KCF-HOG linear	KCF-HOG polyinomial	KCF-HOG gaussian
Overlap ↑	0.47	0.59	0.46	0.43	0.46
Failures ↓	1.25	1.45	0.77	0.57	0.57

combinations of both parameters. In Figure 5.7 we show average failure rate (left) and average overlap (right) over all eight sequences with respect to λ_D and λ_G.

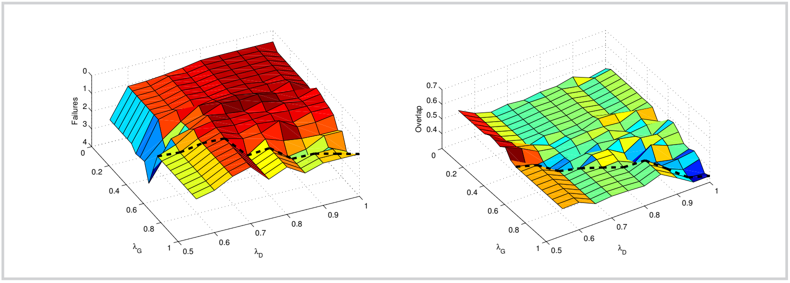


Figure 5.7

Average failure rate (left) and overlap (right) over all test sequences for different combinations of parameters λ_D and λ_G.

We omit the values for $\lambda_D < 0.5$ because the performance of the tracker drastically decreases for lower detection threshold. More generally we can see that lowering the λ_D parameter first increases robustness as good templates can contribute to tracking. However, further decreasing of the threshold rapidly decreases tracker robustness as it allows entering detection mode with less reliable template matching scores. Raising λ_G decreases the effects of guide mode which lowers the accuracy of the tracker. The black line denotes the condition where $\lambda_D = \lambda_G$, above this line the guide mode is disabled. We can see that best performance in terms of failure rate can be achieved if λ_G is set to lower values than λ_D, which means that templates with less reliable matching scores can still contribute to the appearance model in the context of the guided mode. Again, if the λ_G is decreased too much, the performance again decreases, however, the performance drop is not large, due to limited influence of the template in the guided mode.

In terms of average overlap the effect of λ_D and λ_G is not as clear. In case of low λ_D the accuracy is increased due to frequent re-initializations. The average overlap lowers in case of high values of both thresholds because the tracking is left to often unreliable part-set. The average overlap also fluctuates around the line $\lambda_D = \lambda_G$, but stabilizes for other threshold combinations, which means that the best performance may be achieved with an appearance model that supports all three tracking modes. The findings are validated again in the next section on the entire VOT2013 dataset.

5.3.3 Comparative evaluation

In the following section we present the comparative performance evaluation of the ANT tracker to a large set of state-of-the-art visual trackers. In order to continue the story from Section 4.3.3 we perform the initial evaluation using the VOT2013 benchmark [100]. The final evaluation was performed on a more recent and even more challenging VOT2014 [101] benchmark.

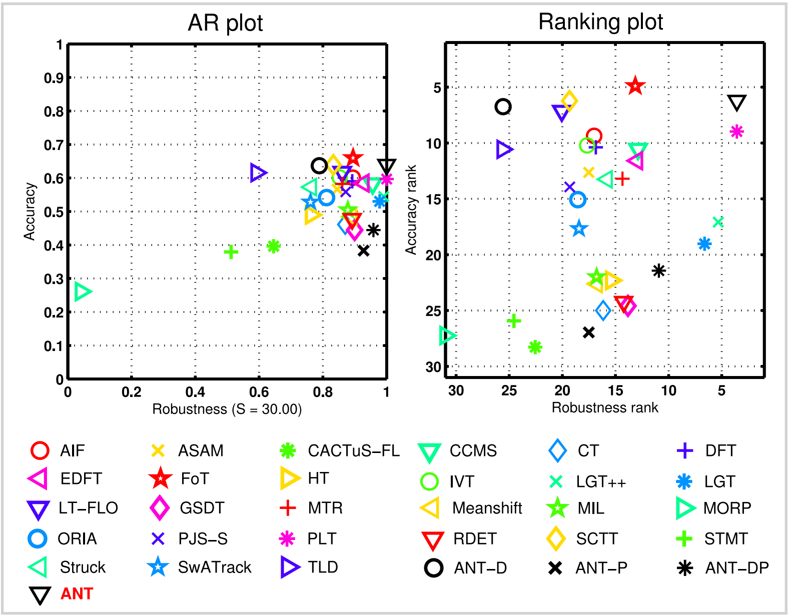


Figure 5.8

Results for VOT2013 benchmark presented as per-attribute ranking A-R plot (left) and per-attribute raw A-R plot (right).

VOT2013: The details about the VOT2013 benchmark were already described in Section 4.3.3. The overall VOT2013 benchmark results are visualized in terms of A-R ranking plot and A-R plot, both shown in Figure 5.8. We can see that the ANT tracker outperforms all reference trackers by achieving the best combination of accuracy and robustness. It is important to note that some trackers, like FoT [64] and LT-FLO [77], achieve a higher accuracy due to many re-initializations, which is not the case with ANT. The proposed tracker shares the first place with the winner of VOT2013 challenge, PLT, in robustness. Both trackers do not fail on any sequence, but PLT achieves a lower accuracy in case of deformations and scale changes due to its holistic appearance model. As seen in Table 5.3, most of the other holistic trackers, like IVT [48], Struck, and EDFT are less robust in tracking non-rigid objects, but achieve a higher accuracy in comparison to the part-based LGT and LGT++, which are related to ANT. On the other hand, the ANT achieves an accuracy comparable to the holistic models and simultaneously outperforms related part-based trackers in robustness. This further confirms our hypothesis about retaining the best properties from holistic and part-based trackers.

Table 5.3

Results for VOT2013 sequences in terms of average overlap (O) and number of failures (F). First, *second and third best values are highlighted. Arrows indicate sorting direction.*

	CCMS		EDFT [125]		FoT [64]		PLT		LGT [107]		ANT	
	O↑	F↓	O↑	F↓	O↑	F↓	O↑	F↓	O↑	F↓	O↑	F↓
bicycle	0.49	*1.00*	0.44	0.00	0.74	*1.00*	0.43	0.00	0.51	*1.00*	*0.61*	0.00
bolt	*0.81*	0.00	0.84	1.00	0.43	3.00	0.65	0.00	0.44	*0.13*	0.66	0.00
car	0.47	*1.00*	0.45	*1.00*	*0.56*	*1.00*	0.44	0.00	0.45	0.00	0.66	0.00
cup	0.72	0.00	0.77	0.00	0.76	0.00	0.81	0.00	0.71	0.00	*0.78*	0.00
david	0.53	*1.00*	0.68	0.00	0.80	0.00	*0.74*	0.00	0.61	0.00	0.72	0.00
diving	*0.39*	0.00	0.33	4.00	0.25	3.00	0.35	0.00	0.43	*0.87*	0.37	0.00
face	0.74	0.00	0.88	0.00	0.66	*1.00*	*0.87*	0.00	0.59	0.00	0.50	0.00
gymnastics	0.59	0.00	0.56	1.00	0.53	2.00	*0.56*	0.00	0.51	*0.80*	0.52	0.00
hand	0.65	0.00	0.51	0.00	0.56	3.00	0.57	0.00	*0.63*	*0.20*	0.52	0.00
iceskater	*0.63*	0.00	0.35	*3.00*	0.35	0.00	0.51	0.00	0.58	0.00	0.69	0.00
juice	0.63	0.00	0.64	0.00	0.89	0.00	0.64	0.00	0.80	0.00	*0.81*	0.00
jump	0.53	*1.00*	0.60	0.00	0.74	0.00	0.34	0.00	0.63	0.00	*0.65*	0.00
singer	0.39	*1.00*	0.37	0.00	0.81	0.00	0.35	0.00	0.21	0.00	*0.53*	0.00
sunshade	0.72	0.00	*0.65*	3.00	0.62	0.00	0.59	0.00	0.57	*0.27*	0.60	0.00
torus	*0.83*	0.00	0.82	0.00	0.79	0.00	0.79	0.00	0.69	0.00	0.83	0.00
woman	*0.66*	2.00	0.60	*1.00*	0.62	8.00	0.69	0.00	0.36	1.13	0.61	0.00
Overall	0.61	0.56	0.60	0.79	*0.64*	1.54	0.61	0.00	0.54	*0.26*	0.64	0.00

Qualitative examples of tracking on VOT2013 sequences that show stable scale adaptation of ANT tracker in comparison to several reference tracker can be seen in Figure 5.10. We can see that the ANT tracker successfully adapts to geometrical and scale changes. In comparison to LGT the tracker manages to survive short occlusion in *bicylce* sequence thanks to the template instance memory system. Another two sequences that are very hard for LGT are *car* and *singer* sequences. In *car* sequence the aspect ratio of the object's region changes together with viewing angle. The sequence is hard for

LGT because of a long static period from frames 90 to 180, where the coupled-layer appearance model slowly spreads to the background. The anchored appearance model, on the other hand, constraints the adaptation because of a good match with a template which provides a strong spatial cue in such cases. The *singer* sequence is also challenging due to simultaneous shrinking of the target and illumination changes. In this case the spatial cue again ensures a more constrained update of the bottom layers. On the other hand, the bottom layers ensure that the memory system is gradually updated with new template instances of the object on multiple scales, as visualized in Figure 5.9.

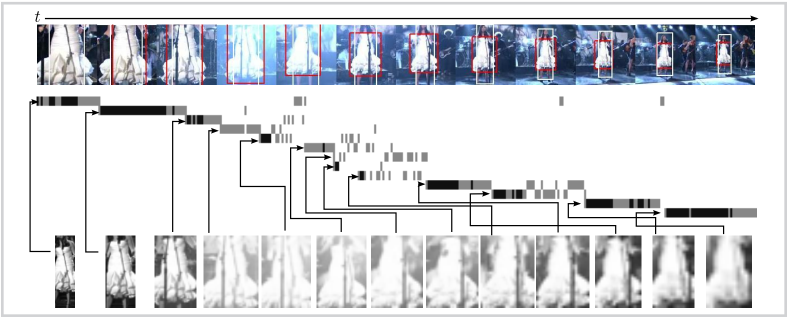


Figure 5.9

Visualization of anchor templates function in tracking with anchored appearance model on sequence *singer* from the VOT2013 dataset. The set of templates on the bottom illustrates the set of templates acquired over the sequence, their introduction into the set is illustrated with an arrow and the their function is illustrated with timeline segments - black denotes detection and gray denotes guiding role.

Benefits of the proposed guiding mechanism: In Figure 5.8 we have included the results for three special derivations of the ANT tracker to demonstrate the contributions of individual tracking approaches and the proposed interaction: The ANT-D tracker is a tracker that only uses the top-layer static template, ANT-P used only parts and segmentation (bottom and middle layer), and ANT-DP uses part-set together with the memory system (bottom and top layer), but the system only acts as a detection mechanism, it does not guide the update of the part set. Since these three trackers can be obtained using special combinations of parameters λ_D and λ_G this analysis can be seen as another re-evaluation of the mode switching parameter analysis, presented in the previous section.

The results are also summarized in Table 5.4. The ANT-D tracker achieves good accuracy, mainly at the expense of robustness since a single static template cannot properly address the appearance changes. On the other hand ANT-P achieves good robust-

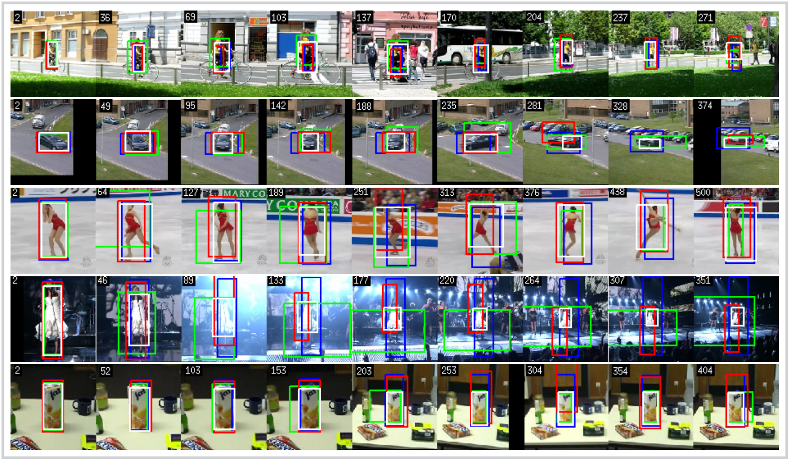


Figure 5.10

Visual comparison of trackers CCMS (red), LGT (green), PLT (blue), and ANT (white) on sequences *bicycle*, *car iceskater*, *singer*, and *juice* from VOT2013 dataset.

Table 5.4

Average overlap and average number of failures on VOT2013 benchmark for ANT tracker derivations. Arrows indicate sorting direction.

	ANT-D	ANT-P	ANT-DP	ANT
Overlap ↑	0.64	0.39	0.46	0.64
Failures ↓	2.39	0.88	0.39	0.00

ness, but the accuracy is low since the part-based model applies self-supervised updating without external supervision and recovery capability from the top-layer memory system. The ANT-DP combines the traits of ANT-D and ANT-P trackers, and benefits from switching between the detection and part-based tracking. The complete ANT tracker improves performance in terms of accuracy and robustness by using anchor templates not only to detect the object, but also to guide the update process of the bottom layer even if the template detection is not reliable enough for a full detection. In particular, ANT improves over the variations ANT-D, ANT-P and ANT-DP in accuracy as well as robustness. The results thus clearly support our hypothesis that the proposed combination of part-based appearance model and holistic appearance model improves the overall tracking performance.

VOT2014: Since the VOT2013 benchmark is saturated in terms of tracking robustness,

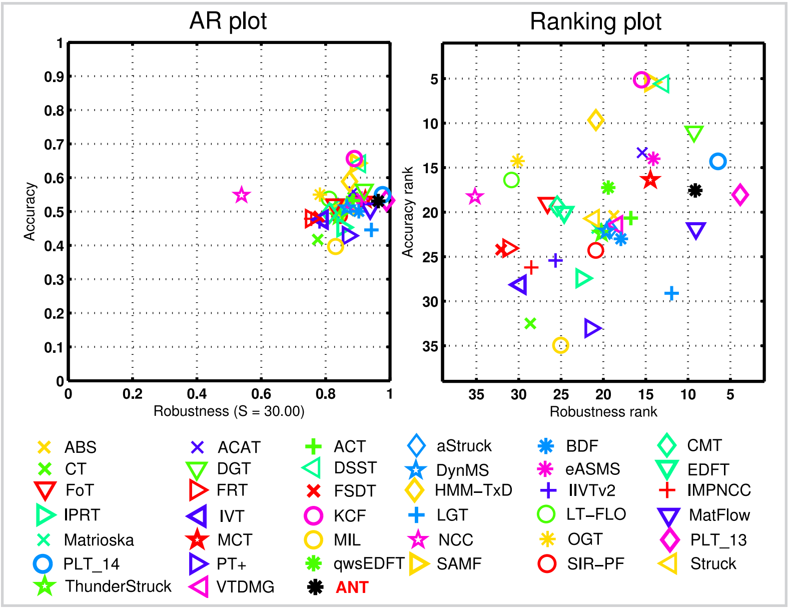


Figure 5.11
Results for VOT2014 benchmark presented as per-attribute ranking A-R plot (left) and per-attribute raw A-R plot (right).

we have also performed a comparison using the more challenging VOT2014 dataset. The VOT2014 introduces several methodological improvements: more accurate annotations and practical difference in case of accuracy. The benchmark also provides results for 38 state-of-the-art visual trackers.

The overall VOT2014 benchmark results are visualized in terms of A-R ranking plot and A-R plot for the baseline experiment, both shown in Figure 5.11 and in Table 5.5, where we show both experiments of VOT2014 challenge, the baseline experiment and the experiment with initialization region perturbation. While the tracker does not achieve the best performance in this case, the results have to be observed in more detail.

As seen in Figure 5.11, the ANT is ranked lower than two variants of PLT tracker and similarly as the part-based DGT in terms of robustness. The raw results in Table 5.6 and Table 5.7 reveal that the PLT_13 and PLT_14 are indeed a bit better in term of failure rate, but the DGT tracker in fact fails approximately four times more often, but only on certain sequences. The DGT failures occur in sequences where the assumptions required for efficient color segmentation are violated. The hybrid nature of the anchored

Table 5.5

Ranking results for both experiments in the VOT2014 benchmark. The table shows accuracy rank (A) and robustness rank (R) for both experiments, as well as average accuracy and robustness rank and the average rank over both experiments (final column). First, *second and third best values are highlighted.*

	baseline		region_noise				
	A	R	A	R	A	R	Rank
DSST [54]	5.58	12.91	5.60	13.10	5.59	13.00	9.30
SAMF [104]	*5.39*	14.43	*5.36*	13.07	*5.37*	13.75	*9.56*
DGT [133]	10.97	9.30	8.51	10.28	9.74	9.79	9.76
KCF [38]	5.13	15.49	5.29	13.24	5.21	14.36	9.79
PLT_14	14.30	*6.45*	13.52	*5.01*	13.91	*5.73*	9.82
PLT_13	18.05	3.83	16.90	4.83	17.48	4.33	10.90
eASMS [134]	14.00	14.10	11.23	14.36	12.61	14.23	13.42
ANT	17.55	9.13	18.36	9.67	17.96	9.40	13.68
HMM-TxD	9.65	20.88	9.45	19.74	9.55	20.31	14.93
ACAT	13.32	15.41	17.30	15.04	15.31	15.22	15.27
MCT [135]	16.37	14.44	17.23	13.12	16.80	13.78	15.29
MatFlow	21.83	9.06	18.82	14.84	20.33	11.95	16.14
ABS	20.39	18.77	15.18	15.49	17.78	17.13	17.46
ACT [136]	20.66	16.76	22.02	15.27	21.34	16.01	18.68
qwsEDFT [137]	17.23	19.42	18.55	21.14	17.89	20.28	19.09
LGT [107]	29.11	11.92	25.85	9.55	27.48	10.74	19.11
VTDMG	21.43	18.39	20.39	17.17	20.91	17.78	19.34
BDF [138]	23.00	17.93	21.57	18.26	22.28	18.09	20.19
Struck [41]	20.72	21.13	21.17	18.88	20.94	20.00	20.47
DynMS	22.12	19.63	21.17	19.66	21.64	19.65	20.64
ThunderStruck [41]	22.39	20.20	21.91	18.72	22.15	19.46	20.81
aStruck	21.90	19.26	20.55	22.01	21.22	20.63	20.93
Matrioska [126]	21.81	20.77	21.75	24.29	21.78	22.53	22.15
SIR-PF	24.29	20.89	22.15	22.54	23.22	21.72	22.47
EDFT [125]	20.01	24.63	21.96	24.27	20.99	24.45	22.72
OGT [139]	14.25	30.13	16.67	30.10	15.46	30.11	22.79
CMT [140]	19.33	25.45	22.03	25.13	20.68	25.29	22.99
FoT [64]	18.97	26.59	21.41	27.15	20.19	26.87	23.53
LT-FLO [77]	16.39	30.85	20.09	31.20	18.24	31.02	24.63
IPRT	27.41	22.55	26.25	23.63	26.83	23.09	24.96
IIVTv2	25.42	25.64	25.14	23.88	25.28	24.76	25.02
PT+	33.04	21.51	30.14	20.20	31.59	20.86	26.22
FSDT	24.21	32.08	24.11	29.14	24.16	30.61	27.38
IMPNCC	26.20	28.51	29.03	29.13	27.62	28.82	28.22
IVT [48]	28.14	29.81	27.39	28.12	27.76	28.97	28.37
NCC [141]	18.27	35.17	23.06	37.08	20.66	36.12	28.39
FRT [70]	24.05	31.22	27.05	31.78	25.55	31.50	28.52
CT [49]	32.49	28.62	30.43	27.87	31.46	28.25	29.85
MIL [42]	34.95	25.03	35.61	25.72	35.28	25.37	30.33

appearance model in ANT is robust to a wider array of visual degradations, hence the lower failure count. This is also noticeable in per-attribute results in Table 5.6 - the ANT tracker is the most vulnerable to occlusion considering the number of failures and the fact that this attribute is only present in a small number of frames in the entire sequence dataset.

In terms of accuracy, the proposed tracker performs less well than on VOT2013 bench-

Table 5.6

Results of SAMF, PLT_14, DSST, KCF, DGT, and ANT trackers for *individual attributes* in VOT2014 dataset in terms of average overlap (O) and number of failures (F). First, *second and third best values are highlighted. Arrows indicate sorting direction.*

	SAMF [104]		PLT_14		DSST [54]		KCF [38]		DGT [133]		ANT	
	O↑	F↓	O↑	F↓	O↑	F↓	O↑	F↓	O↑	F↓	O↑	F↓
cam. mot.	0.66	24.00	0.56	4.00	*0.66*	20.00	0.67	24.00	0.56	19.00	0.52	*7.00*
ill. ch.	0.67	*1.00*	0.50	*1.00*	0.75	*1.00*	*0.74*	*1.00*	0.47	14.00	0.54	0.00
occlusion	0.61	4.00	0.59	*2.00*	*0.63*	3.00	0.64	5.00	0.48	1.00	0.51	4.00
size	0.56	18.00	0.51	4.00	0.52	15.00	0.58	20.00	*0.58*	*6.00*	0.50	*6.00*
motion	*0.67*	25.00	0.57	4.00	0.65	24.00	0.67	26.00	0.58	14.00	0.55	*7.00*
empty	*0.57*	0.00	0.53	0.00	0.56	0.00	0.54	0.00	0.68	0.00	0.57	0.00
Overall	*0.64*	19.23	0.55	3.41	0.64	16.90	0.66	19.79	0.56	13.78	0.53	*5.69*

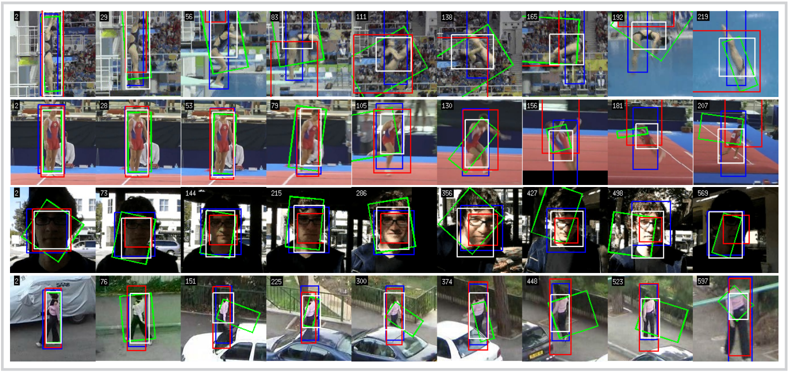


Figure 5.12

Visual comparison of trackers KCF (red), LGT (green), PLT_14 (blue), and ANT (white) on sequences *diving*, *gymnastics trellis*, and *woman* from VOT2014 dataset.

mark. This can be at least to some degree attributed to different annotation format, that uses rotated bounding boxes, which reduces region overlap with axis-aligned bounding boxes, reported by ANT. Trackers like DGT achieve better accuracy by utilizing computationally expensive segmentation. Holistic trackers, like DSST, KCF and SAMF perform better in accuracy, at a relative difference of about 10%, but fail approximately four times more often. This means that they are also more often reinitialized which consequently corrects the region estimate, resulting in artificially improved final accuracy. On the other hand, both PLT trackers perform comparably in terms of accuracy. The PLT_14 tracker, which extends the VOT2013 challenge winner, denoted with PLT_13, also provides size adaptation. This ensures the tracker with marginally higher accuracy, however, the relaxation of constraints also results in lower robustness. With this we can see the importance of looking at both aspects of tracking performance, as we have already emphasized in Chapter 3, where we have proposed the performance evaluation methodology. Considering the fact that the anchored appearance model is even more relaxed in its free mode (tracking only with part-set), it is not surprising that it is also most vulnerable in this case. In fact most failures in VOT2014 dataset can be traced back to longer periods of appearance model working in free mode due to fast visual changes that prevented guidance from the top layer.

Another important observation is that ANT tracker outperforms the LGT tracker in accuracy at relative increase of approximately 10% and significantly outperforms it in robustness. This means that improved accuracy is not due to re-initializations, but more robust and accurate tracking. In the second experiment (random perturbations of initialization region) both trackers are ranked approximately similar in terms of robustness, however, the ANT tracker is much more accurate.

Table 5.7

Results of SAMF, PLT_14, DSST, KCF, DGT, and ANT trackers for *individual sequences* in VOT2014 dataset in terms of average overlap (O) and number of failures (F). First, *second and third best values are highlighted. Arrows indicate sorting direction.*

	SAMF [104]		PLT_14		DSST [54]		KCF [38]		DGT [133]		ANT	
	O↑	F↓	O↑	F↓	O↑	F↓	O↑	F↓	O↑	F↓	O↑	F↓
ball	*0.77*	*1.00*	0.70	0.00	0.56	*1.00*	0.75	*1.00*	0.81	0.00	0.42	*1.00*
basketball	0.75	0.00	*0.74*	0.00	0.64	*1.00*	0.64	0.00	0.50	0.00	0.55	0.00
bicycle	0.61	0.00	0.57	0.00	0.58	0.00	*0.62*	0.00	0.63	0.00	0.61	0.00
bolt	0.56	2.00	0.47	0.00	*0.56*	*1.00*	0.49	3.00	0.49	0.00	0.47	0.00
car	0.51	0.00	0.38	0.00	0.73	0.00	*0.70*	0.00	0.57	0.00	0.61	*1.00*
david	*0.82*	0.00	0.65	0.00	0.80	0.00	0.82	0.00	0.53	*1.00*	0.72	0.00
diving	0.24	4.00	0.39	0.00	*0.44*	*1.00*	0.25	4.00	0.34	0.00	0.47	0.00
drunk	*0.57*	0.00	0.52	0.00	0.55	0.00	0.53	0.00	0.67	0.00	0.57	0.00
fernando	0.39	*1.00*	0.40	*1.00*	0.34	*1.00*	0.41	*1.00*	0.61	0.00	*0.44*	*1.00*
fish1	*0.49*	3.00	0.41	0.00	0.32	*1.00*	0.42	3.00	0.56	0.00	0.33	0.00
fish2	0.30	5.00	0.25	0.00	0.35	4.00	0.26	6.00	0.48	*2.00*	*0.38*	*2.00*
gymnastics	0.54	2.00	0.59	0.00	0.63	5.00	0.53	*1.00*	0.58	0.00	*0.61*	0.00
hand1	0.54	3.00	0.66	0.00	0.21	2.00	0.56	3.00	*0.63*	*1.00*	0.55	0.00
hand2	0.46	5.00	0.61	0.00	0.52	6.00	0.49	6.00	0.52	5.00	*0.56*	*2.00*
jogging	0.82	*1.00*	0.70	*1.00*	0.79	*1.00*	*0.79*	*1.00*	0.66	0.00	0.65	*1.00*
motocross	0.40	4.00	*0.51*	1.00	0.42	4.00	0.36	*2.00*	0.49	1.00	0.56	1.00
polarbear	0.71	0.00	0.63	0.00	0.63	0.00	*0.78*	0.00	0.81	0.00	0.65	0.00
skating	0.45	0.00	0.49	0.00	*0.59*	0.00	0.68	*1.00*	0.39	7.00	0.42	0.00
sphere	0.88	0.00	0.67	0.00	0.92	0.00	*0.90*	0.00	0.84	0.00	0.66	0.00
sunshade	0.76	0.00	0.73	0.00	0.78	0.00	*0.76*	0.00	0.51	0.00	0.61	0.00
surfing	*0.80*	0.00	0.80	0.00	0.90	0.00	0.79	0.00	0.63	0.00	0.75	0.00
torus	*0.84*	0.00	0.71	0.00	0.81	0.00	0.85	0.00	0.83	0.00	0.83	0.00
trellis	0.81	0.00	0.50	0.00	*0.80*	0.00	0.79	0.00	0.48	0.00	0.57	0.00
tunnel	0.54	0.00	0.27	0.00	0.80	0.00	*0.68*	0.00	0.44	*8.00*	0.23	0.00
woman	*0.76*	*1.00*	0.76	*1.00*	0.79	*1.00*	0.74	*1.00*	0.54	0.00	0.62	0.00
Overall	*0.64*	0.92	0.56	0.13	0.63	0.84	0.64	0.99	0.58	1.15	0.54	*0.27*

Looking at the results for individual sequences of the VOT2014 dataset in Table 5.7, the most problematic ones for the ANT tracker are *fish2*, *hand2*, and *fernando* due to frequent target deformations and ambiguous color palette. Other notable failure cases will be analyzed in detail in Section 5.3.5. On the other hand, the ANT tracker excels on many other sequences, some examples are shown as time-strips in Figure 5.12. Sequences *diving*, *gymnastics*, and *woman* were also part of the VOT2013 dataset (with different annotations). We can see that the ANT tracker performs much better than the LGT tracker. It is also much better in estimating the size of the object than two top-ranking trackers.

5.3.4 Non-rigid objects

To further analyze the behavior of the proposed tracker on non-rigid objects we perform a similar analysis than in Section 4.3.4. We have selected nine sequences from VOT2014 dataset that include non-rigid objects, i.e. *bolt*, *diving*, *fernando*, *fish1*, *fish2*, *gymnastics*, *hand1*, *hand2*, and *skating* and performed ranking analysis on this subset. The results are summarized in Figure 5.13. The three correlation filter trackers that performed really well on account of high accuracy on the entire VOT2014 dataset (DSST, SAMF, and KCF), are performing much worse on the subset of non-rigid objects, as are many other holistic trackers, e.g. IVT, MIL, and CT. The most robust trackers are still PLT_13 and PLT_14, as well as LGT and ANT. The difference between the LGT and ANT is actually smaller on this subset, which could indicate that the LGT and ANT perform similarly on non-rigid sequences, but the ANT tracker performs better on the remaining subset because of its template memory system.

5.3.5 Failure cases

Overall, the proposed visual tracker achieves state-of-the-art performance on VOT2013 benchmark, where it achieves both state-of-the-art robustness and accuracy. In case of the more challenging VOT2014 benchmark, the tracker is still competitive, but the dataset also reveals some of its weak spots. In the previous section we have already mentioned some sequences from the VOT2014 dataset, where the anchored appearance model is unable to track the object successfully. In this section we will analyze some notable failure cases of the proposed model and discuss how they could be avoided.

Occlusion is the main cause of failure in case of *jogger* (most of the reference trackers fail in this case) and *car*. The *car* sequence occlusion is especially interesting as the occlusion is not complete, as seen in Figure 5.14, the car is partially occluded by tree branches. This still causes a lot of problems for the part-based model, which causes the failure. Se-

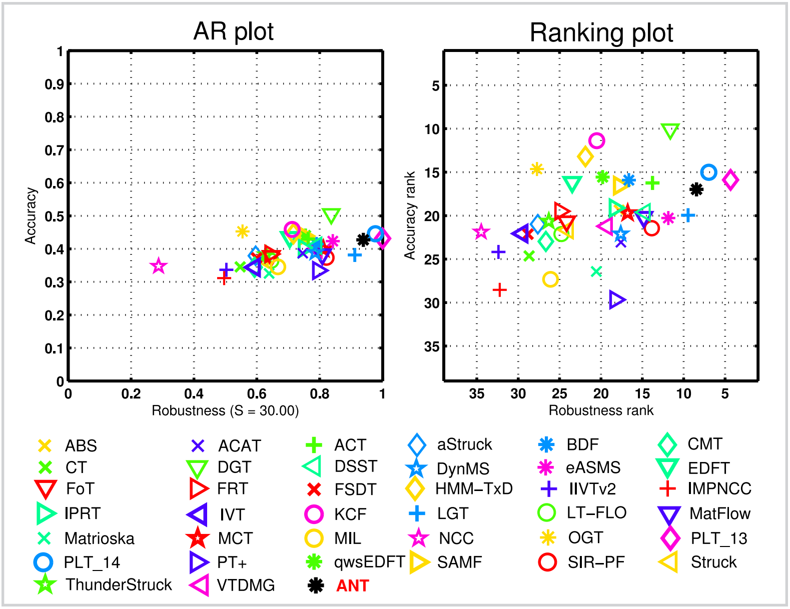


Figure 5.13

Results for subset of VOT2014 benchmark sequences that contain non-rigid objects presented as per-attribute ranking A-R plot (left) and per-attribute raw A-R plot (right).

quence *motocross* is challenging due to rotation of the object, which is not accounted for by the template representation that we are using in the evaluated implementation of the top layer of the anchored appearance model. A feature-based representation of template instances would probably be better suited in such cases. A very interesting that tells a lot about the appearance model occurs in sequence *ball*. The ball has a very distinct texture, that would be easy to localize in case of object translation. But since the ball starts to roll, such object becomes hard to match for a template method. On the other side, the part-based bottom layer also adapts to the changes - this means that all the parts move towards the edge of the ball because of the rotation. Both phenomena result in poor accuracy and eventual failure of the tracker.

Another case of poor accuracy can be seen in sequence *tunnel*. The ANT tracker did not fail this time, however, it achieved its lower average overlap on the entire dataset. Only one part of a motorcycle is annotated as the target, which is good for holistic trackers, however, the sequence contains a lot of illumination changes and saturated colors. Despite surviving these changes and recovering from the accuracy problems in the beginning of the sequence, the tracker then starts tracking the other part of the motorcycle, reducing the overlap even further.

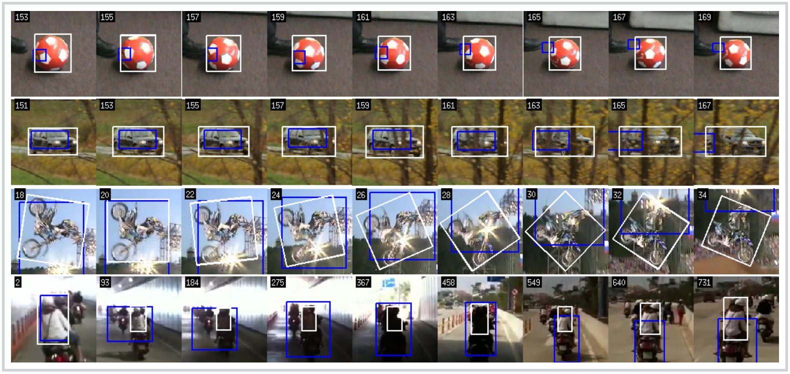


Figure 5.14

Failure cases for ANT tracker (blue), demonstrated on sequences *ball*, *car*, *motocross*, and *tunnel*. The ground-truth region is shown with white color. In first three cases the strip focuses on behavior before the failure.

5.4 *Summary*

In this chapter we have presented a new hierarchical appearance model that addresses some of the problems of the coupled-layer appearance model, presented in Chapter 4. The model is composed of three layers that differ in the level of detail by which they describe the target. We use holistic detailed, holistic coarse and part-based layers that mutually interact in localization and updates by accounting for the potential uncertainty of the visual information. This makes the appearance model shift between purely holistic and part-based behavior, depending on the visual uncertainty.

A tracker that uses the anchored appearance model was evaluated on two recent benchmarks - VOT2013 [100] and VOT2014 [101]. Analysis of the influence of different layers showed that all layers contribute to improved performance and that the significant improvement comes from the mutual interaction between them. The tracker was also compared to a large set of state-of-the-art trackers in each benchmark. Results confirm the hypothesis that the proposed tracker outperforms related part-based trackers (like LGT) both in accuracy and robustness and can compete with the top-performing trackers. Furthermore, the top-layer anchors in our tracker are very general and can easily be replaced by object-class-specific detectors like face detectors to aid tracking in specific applications like face tracking. Another direction of improvement that we will also pursue is better segmentation in the middle layer that could provide better object region estimate and consequentially improve tracking accuracy.

6

Conclusion

This doctoral thesis we have addressed the field of visual tracking, one of the major research topics in computer vision. As stated in the introduction in Chapter 1, where we have defined our research scope, we are dealing with a subtopic of on-line single-target short-term tracking using a monocular camera system. We have described a novel formalization of appearance model structure, called a hierarchical appearance model. The concept was designed to address some of the problems with existing work that we highlight in the review of the related work in Chapter 2, most importantly non-rigidness of tracked objects with unknown geometrical properties as well as integration of higher-level appearance cues into the appearance model. We have also presented a new methodology that we have developed for unbiased evaluation of visual trackers and used it evaluate our titular work in this thesis - the two visual trackers that implement the hierarchical appearance model concept. The body of the thesis can be summarized from the perspective of the claimed research contributions, first listed in Section 1.3:

A novel visual tracking algorithm formalization that integrates local and global appearance information. The idea of a hierarchical appearance model was introduced in Section 1.2. In Chapter 4 we have presented the first working instance of a hierarchical appearance model, the coupled-layer appearance model. This appearance model combines local and global appearance description. We have presented the required details for the implementation of the proposed appearance model: the visual description of parts that describe the local appearance of the object, an efficient optimization for the matching operation for the set of parts, a set of heuristics for part weight dynamics, and the structure of a multi-modal global appearance model. We have described how the appearance model is integrated in a visual tracker and how it interacts with a motion model of the tracker. The tracker was then evaluated in terms of parameter configuration and in comparison to the related work. The experimental analysis shown that the proposed appearance model is indeed capable of tracking objects robustly with no a-priori object-specific knowledge and that the model excels in many situations with non-rigid deformations. Our work on the coupled-layer appearance model for visual tracking has already been published in [106, 107].

The analysis of the coupled-layer appearance model has also shown some weak-spots of the proposed concept, like reduced accuracy. Some of these problems were addressed by our second hierarchical appearance model, described in Chapter 5. In this appearance model the appearance information is structured in three layers. The bottom two are conceptually similar to the local and global layer of the coupled-layer appearance model. The main advantage of the new appearance model is the added third layer that guides the lower two layers by constraining their adaptation. This layer contains a memory system that accumulates snap-shots of objects appearance over time, enabling the appearance model to recover if the object is re-detected. The experimental analysis confirms that the the constraints of the third layer indeed improve performance of the tracker.

A new evaluation methodology for visual tracking. In terms of performance evaluation visual trackers we have focused on the problem of performance evaluation in single-target short-term visual tracking. In Chapter 3 we have first presented three core requirements for a comprehensive evaluation framework. In the scope of this thesis we have addressed two of these requirements - the first one is the selection of the evaluation measures. Through theoretical and experimental analysis we have investigated various popular performance evaluation measures, discussed their pitfalls and showed that many of the widely used measures are equivalent. Since some measures reflect certain aspect of tracking performance, combining those that address the same aspect provides no additional information regarding the performance or even introduces bias toward a certain aspect of performance to the result. Based on the results of our experiment we have proposed to use a pair of two existing complementary measures. This pair, that we call the accuracy-robustness, or A-R pair, measures the accuracy and the robustness of each tracker. We have also proposed an intuitive way of visualizing the results as an A-R scatter plot. Our preliminary work on this topic has been published in [98, 99]. We have also introduced several simple theoretical trackers that can be used to quickly review the results of the evaluated trackers in terms of basic properties. The A-R measures were also extended to the case of ranking multiple visual trackers on a given set of sequences. We have introduced appropriate statistical tests to determine performance equivalence that as a first step towards determining tracker equivalence classes. We have described two performance evaluation systems that we have developed, both supporting the same third-party tracker integration approach using a custom communication protocol that we have designed to make performance evaluation and analysis of tracking algorithms easy and extensible.

6.1 Future research directions

In the end it is important to remember that visual tracking is in fact an ill-posed problem if it is considered as a solitary task, especially if we compare the tracking capability of a visual tracker to a human one. Being able to follow appearance changes and movement of objects in video sequence at the level of people requires complex prior knowledge and understanding of the scene, which supersedes the task of object tracking alone. In order to completely solve visual object tracking, an algorithm would have to address a large part of the computer vision field.

At the same time, many applicative tracking scenarios provide prior constraints that make tracking much more realistic problem, even when it is considered as a separate problem. The exciting aspect of a hierarchical appearance model concept, when thinking about it as a formal way of structuring appearance information for visual tracking, is that it offers a great deal of flexibility. It offers a way of introducing prior knowledge about the object in the appearance model with pre-trained higher layers in the visual hierarchy. At the same time it also provides an opportunity to build a temporal-invariant model of an object on-line by ensuring high-confidence learning samples. This presents an opportunity to connect tracking to other fields of computer vision, like detection and categorization, which is one of the research directions that we plan to investigate in the future. Having a more temporal-invariant representation of an object can also be applied to multi-camera tracking. An object that is observed from multiple viewpoints can be represented by a common higher layer representation, while the lower layer representation remains view-dependent.

Visual tracking research has to be supported by interpretable empirical analysis. In this context we would like to emphasize the fact that our work on performance evaluation is already gaining momentum in the research community. As we have shown at the end of Chapter 3, the performance measures described in this thesis are now an integral part of the Visual Object Tracking Challenge that represents an ongoing effort to promote a consistent evaluation methodology, thus pushing forward the field of visual tracking. The biggest test of the VOT Challenge and similar initiatives in the next years will be to withstand the temptation of reducing visual tracking performance evaluation to a mindless competition that is only interested in minor improvements of some measure of performance on a finite sequence set, but instead acknowledge the diversity of the research problem and offer meaningful interpretations of strengths and weaknesses of various tracking approaches. Hopefully, the adoption of thorough performance analysis will also be easier due to evaluation software that we have presented as a part of this thesis as well as a reference communication protocol implementation, all of which are available as open-source software. This way researchers in the field of visual tracking can save time when it comes to evaluation by reusing and contributing to existing evaluation tools. Good evaluation systems can also help researchers understand the behavior of a visual tracker through parameters analysis, which can lead to new discoveries and improvements, thus pushing forward the entire field of visual tracking.

Proofs and derivations

A

A.1 Proof that AUC equals to average overlap

Problem: Let $\phi_1, \phi_2, \ldots, \phi_N$ be frame overlaps for a sequence of length N. We assume that the frame overlaps are *ordered* by scale from minimal to maximal value and $\phi_0 = 0$, i.e.

$$0 = \phi_0 \leq \phi_1 \leq \cdots \leq \phi_N.$$

Let $P(\tau) = |\{j : \phi_j \geq \tau\}|$ be the number of overlaps greater than τ. The AUC measure is an integral of $\frac{P(\tau)}{N}$ from 0 to 1. We want to prove that the average overlap, $\bar{\phi}$, for the sequence $\phi_1, \phi_2, \ldots, \phi_N$ equals to the computed AUC, i.e.

$$\frac{1}{N}\sum_{i=1}^{N}\phi_i = \frac{1}{N}\int_0^1 P(\tau)d\tau.$$

Proof: Function P is a step function (constant between ϕ_i and ϕ_{i+1}). Therefore its integral I is

$$I = \sum_{i=0}^{N-1} P(\phi_i)(\phi_{i+1} - \phi_i).$$

$$\begin{aligned}
I &= P(\phi_0)(\phi_1 - \phi_0) + P(\phi_1)(\phi_2 - \phi_1) + P(\phi_2)(\phi_3 - \phi_2) + \ldots \\
&= \phi_1 P(\phi_0) - \phi_0 P(\phi_0) + \phi_2 P(\phi_2) - \phi_1 P(\phi_1) + \phi_3 P(\phi_3) - \phi_4 P(\phi_4) + \ldots \\
&= -\phi_0 P(\phi_0) + \phi_1(P(\phi_0) - P(\phi_1)) + \phi_2(P(\phi_1) - P(\phi_2)) + \cdots \\
&= 0 \cdot P(\phi_0) + \phi_1 \cdot 1 + \phi_2 \cdot 1 + \cdots && \text{(A.1)} \\
&= \phi_0 + \phi_1 + \phi_2 + \cdots && \text{(A.2)} \\
&= \sum_{i=1}^{N} \phi_i
\end{aligned}$$

In (A.1) we have assumed that the shift between the two consequential values of $P(\tau)$, i.e. $P(\phi_i) - P(\phi_{i+1})$ equals to 1, that is true if all ϕ_i are different. If k consequential ϕ_i are equal then the corresponding $k-1$ shifts are 0, while the last one is k. However, in (A.1) we add $(\phi_i \cdot 1)$ k times. ■

A.2 Reformulation of CoTPS measure

Let $\phi_1, \phi_2, \ldots, \phi_N$ be frame overlaps for a sequence of length N. In [92], the CoTPS measure is defined as a weighted average of two factors, that the authors define as *tracking accuracy*, Ω, and *tracking failure*, λ_0, that are combined using a dynamically computed factor, β, as

$$CoTPS = \beta\Omega + (1-\beta)\lambda_0. \tag{A.3}$$

The tracking failure factor λ_0 is computed as the percentage of frames where the tracker failed, i.e. $\lambda_0 = \frac{N_0}{N}$, where N_0 is a number of frames where the overlap between ground-truth region and the predicted region is 0. The weight factor is defined as $\beta = \frac{\hat{N}}{N}$, where $\hat{N}$ denotes the number of frames where the overlap is higher than 0, therefore $\beta = 1 - \lambda_0$. The definition for tracking accuracy part Ω is

$$\Omega = \sum_{\tau \in (0,1]} \frac{\hat{N}(\tau)}{\hat{N}}, \tag{A.4}$$

where $N(\tau) = |\{j : \phi_j \geq 0 \wedge \phi_j \leq \tau\}|$ denotes the number of frames that is higher than 0, but lower than τ. We observe that (A.4) is actually an approximation of the integral with respect to threshold τ, that can also be reformulated as

$$\Omega = \int_0^1 \frac{\hat{N}_\tau}{\hat{N}} d\tau = 1 - \int_0^1 \frac{\hat{P}_\tau}{\hat{N}} d\tau, \tag{A.5}$$

where $P(\tau) = |\{j : \phi_j \geq \tau\}|$. According to the proof in Appendix A.1, the integral results in average overlap over a set of frames, therefore $\Omega = 1 - \hat{\phi}$, where $\hat{\phi}$ is the average overlap over $\{\phi_j : \phi_j \geq 0\}$. Therefore, the CoTPS measure can be rewritten as

$$CoTPS = (1-\lambda_0)(1-\hat{\phi}) + \lambda_0^2. \tag{A.6}$$

Considering that average overlap over the entire sequence can be written as $\phi = (1 - \lambda_0)\hat{\phi}$, we can further derive

$$CoTPS = 1 - \bar{\phi} - (1-\lambda_0)\lambda_0, \tag{A.7}$$

meaning that the CoTPS measure is a function of average overlap as well as the percentage of frames where the overlap is 0.

TraX Protocol

B

One of the problems in visual tracking research is fragmentation of evaluation methodology. While authors of new tracking methods provide comparative experimental results for their methods and the state-of-the-art in the papers, the evaluation procedures and datasets differ from one papers to another. Besides that the results are usually trimmed down to some summarizing performance scores due to paper length limitations. This makes it difficult for other researchers to simply reuse these results in their own evaluation. One way to overcome this problem is to share the tracker implementation. In the past years researchers tend to provide a binary form or even a source code of their implementation more frequently as a supplementary material to their papers. However, while these resources are indeed valuable as they encourage other people to repeat the experiments or perform new tests, the process of preparing such a tracker for such evaluation is still time consuming. In case of binary versions it can be even impossible to properly run the tracker on arbitrary testing image sequence and obtain results that can be then compared with other trackers.

A common challenge a computer vision researcher faces when designing a new visual tracking algorithm is how to perform comparative experiments without spending too much time on technical details of reference trackers. In this appendix we present a simple stateful communication protocol [142] that allows researchers to quickly set up a second-party code in an evaluation environment or enable their own trackers to be integrated across a variety of different evaluation or visualization tools. The protocol is called TraX which stands for *Visual Tracking eXchange* protocol and was first officially published as a technical report [143] with the VOT2014 challenge.

The rest of the chapter is organized as follows: Section B.1 provides the basic foundations of the protocol. Section B.2 describes the message structure structure in Section B.3 defines protocol states and Section B.4 defines the supported data formats. In Section B.5 we provide tips for protocol implementation and integration and we draw concluding remarks in Section B.6, where we also describe the future plans for the protocol.

B.1 Overview and definitions

The TraX protocol is designed with simplicity of integration in mind as well as flexibility that allows extensions and custom use-cases. The protocol is based on the a mechanism that all modern operating systems provide: standard input and output streams of a process. The main idea is that we embed the communication between the tracker process and the control process in these streams. The communication is divided into line-based messages. Each message can be identified by a prefix that allows us to filter out tracker custom output from the protocol communication.

First we define the basic terminology of the protocol:

1. Server: We adopt the standard client-server terminology when describing the interaction. A server is a tracker process that is providing tracking information to the client that is supplying the server with requests – a sequence of images. Unlike traditional servers that are persistent processes that communicate with multiple clients, the server in our case is started by a single client and is only communicating with it.

2. Client: A client is a process that is initiating tracking requests as well as controlling the process. In most cases this would be an evaluation software that would aggregate tracking data for performance analysis, however, additional use-cases can be determined.

3. Message: Server and client communicate with each other using messages. Each message begins in new line, is prefixed by an unique string and ends with the end of the line. Types of messages are defined in Section B.3 and the exact structure of a message is defined in Section B.2.

B.2 Message format

Individual message in the protocol is a line, which means that it is separated from the past and future stream content by the new line (EOL) character. The format of all client or server messages is the same. To distinguish between arbitrary program outputs and embedded TraX messages a prefix "`@@TRAX:`" is used. The prefix is followed immediately (without white space character) by the name of the message, which is then followed by space-separated arguments. The format is illustrated in Figure B.1. The message header is followed by a number of mandatory message arguments. This number depends on the type of the message and on the runtime configuration. The mandatory arguments are then followed by a variable number of optional named arguments that can be used to communicate additional data.

All the arguments can contain spaces, however, they have to be enclosed by double-quote symbols. To use the same symbol inside the argument, it has to be prefixed by back-slash symbol. To use newline symbol inside the argument, it has to be replaced using `\n` symbol sequence.

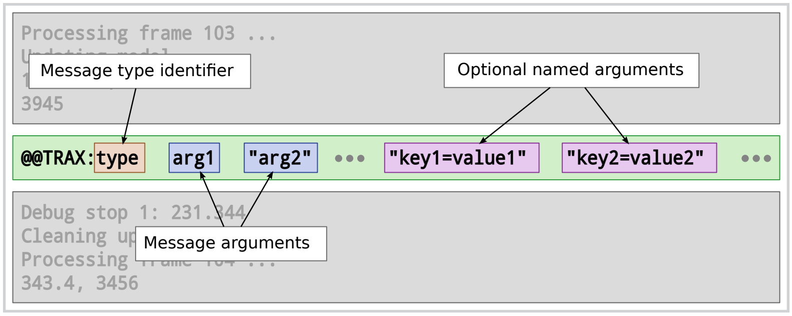


Figure B.1

An illustration of a typical protocol message (green box) embedded within the process output stream (gray boxes).

B.3 Protocol messages and states

Below we list the valid messages of the protocol as well as the states of the client and server. Despite the apparent simplicity of the protocol its execution should be strict. An inappropriate or indecipherable message should result in immediate termination of connection in case of both parties.

- `hello` (server): The message is sent by the server to introduce itself and list its capabilities. This message specifies no mandatory arguments, however, the server can report the capabilities using the optional named arguments. The official arguments, recognized by the first version of the protocol are:
  - trax.version (integer): Specifies the supported version of the protocol. If not present, version 1 is assumed.
  - trax.name (string): Specifies the name of the tracker. The name can be used by the client to verify that the correct algorithm is executed.
  - trax.identifier (string): Specifies the identifier of the current implementation. The identifier can be used to determine the version of the tracker.
  - trax.image (string): Specifies the supported image format. See Section B.4 for more details.
  - trax.region (string): Specifies the supported region format. See Section B.4 for more details.
- `initialize` (client): This message is sent by the client to initialize the tracker. The message contains the image data and the region of the object. The actual

format of the required arguments is determined by the image and region formats specified by the server.

- `frame` (client): This message is sent by the client to request processing of a new image. The message contains the image data. The actual format of the required argument is determined by the image format specified by the server.

- `state` (server): This message is used by the server to send the new region to the client. The message contains region data in arbitrary supported format (most commonly the same format that the server proposed in the introduction message).

- `quit` (client, server): This message can be sent by both parties to terminate the session. The server process should exit after the message is sent or received. This message specifies no mandatory arguments.

The state diagram of server and client is defined by a simple automata, shown in Figure B.2. The state changes upon receiving appropriate messages from the opposite party. The client state automata consists of the following states:

1. Introduction: The client waits for `hello` message from the server. In this message the server describes its capabilities that the client can accept and continue the conversation by moving to *initialization* state, or reject it and terminate the session by sending the `quit` message.

2. Initialization: The client sends a `initialize` message with the image and the object region data. Then the client moves to *observing* state.

3. Observing: The client waits for a message from the server. If the received message is `state` then the client processes the incoming state data and either moves to *initialization*, *termination* or stays in *observing* state. If the received message is `quit` then the client moves to *termination* state.

4. Termination: If initiated internally, the client sends the `quit` message. If the server does not terminate in a certain amount of time, the client can terminate the server process.

The server state automata consists of the following states:

1. Introduction: The server sends an introductory `hello` message where it optionally specifies its capabilities.

2. Initialization: The server waits for the `initialize` or `quit` message. In case of `initialize` message a tracker is initialized with the given data and the server moves to *reporting* state. The new state is reported back to the client with a `state` message. In case of the `quit` message the server moves to *termination* state.

3. Reporting: The server waits for the `frame`, `initialize`, or `quit` message. In case of `frame` message the tracker is updated with the new image information and the new state is reported back to the client with a `state` message. In case of `initialize` message a tracker is initialized with the given data and the new state is reported back to the client with a `state` message. In case of the `quit` message the server moves to *termination* state.

4. Termination: If initiated internally, the server sends the `quit` message and then exits.

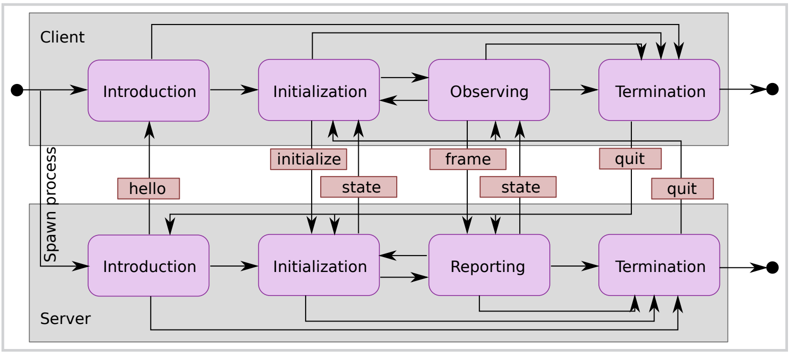


Figure B.2

A graphical representation of client and server automata together with protocol states.

B.4 Data formats

The protocol is designed to be scalable, however, only a few basic data formats have been specified in this first iteration of the protocol. There are three region formats and two image formats.

B.4.1 Region formats

Regions can be encoded in multiple ways. The traditional axis-aligned rectangles are supported, next to more sophisticated polygon format. Both region formats are shown graphically in Figure B.3.

1. Rectangle: The simplest form of region format is the axis-aligned bounding box. It is described using four values, `left`, `top`, `width`, and `height` that are separated by commas.

2. Polygon: A more complex and flexible region description that is specified by even number of at least six values, separated by commas that define points in the polygon (`x` and `y` coordinates).

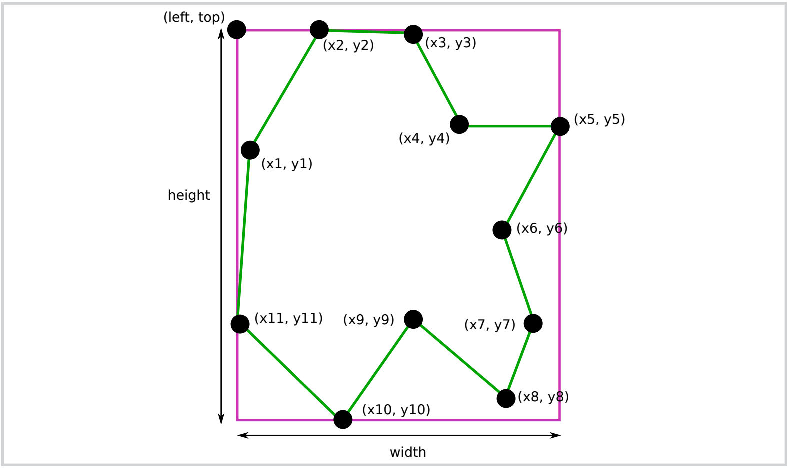


Figure B.3
An illustration of rectangle and polygon region encoding.

B.4.2 Image format

In the current version of the protocol, image can be provided to the server by specifying an absolute path on a local file-system that points to a JPEG or PNG file. The server should take care of the loading of the image to the memory.

B.5 Integration

To integrate the protocol into an existing tracker one has to identify the tracking loop in the algorithm and place the protocol handles to the appropriate locations. A sketch of integration in a pseudo-code tracker is shown in Figure B.4.

```
Setup tracker
TraX: Initialize protocol, report introduction message
loop
    TraX: Wait for message from client
    if initialize message then
        Initialize tracker with provided region and image
        TraX: Report state message
    else if frame message then
        Update tracker with provided image
        TraX: Report state message
    else if quit message then
        Break the tracking loop
    end if
end loop
Cleanup tracker
TraX: Cleanup protocol (terminate if necessary).
```

Figure B.4
Pseudo-code sketch of server protocol integration.

A far better solution than implementing the protocol yourself is to use an open-source reference implementation library, presented in Appendix C.

B.6 Summary

In this appendix we have presented the first iteration of a visual tracking exchange protocol that attempts to standardize the communication between evaluation toolkits and tracker implementations. The idea of the tracker is to make the development of tracking algorithms easier by separating the core algorithm implementation from the auxiliary functionality like visualization and performance evaluation. Publicly available evaluation tool that supports the protocol is the Visual Object Tracking toolkit, which uses the protocol as the default integration technique since the VOT2014 challenge [101]. We hope that the presented protocol will be adopted by the community which will result in interoperability of evaluation tools, which will in turn benefit the research community in general.

Software

C

In most computer vision research, the theoretical work has to be substantially supported by good software implementation. In the course of this doctoral thesis we have produced many software products that we have, or will in the near future, release to the research community under open-source software licenses. In this appendix we list the most mature and notable software projects that are closely related to the topic of this thesis.

C.1 *libtrax*

The *libtrax* library is a reference implementation of the TraX communication protocol, described in Appendix B. It is written in C programming language without any external dependencies. The library implements the server and client side of the protocol. Additionally, the library also provides a simple command-line client as well as some example trackers that demonstrate the integration principles. The native C library also has bindings to several other languages, such as Matlab and in the future also Python and Java. The source code of the library is available on Github: `https://github.com/lukacu/trax`.

C.2 *VOT toolkit*

VOT toolkit is the official evaluation toolkit for the Visual Object Tracking Challenge. It is mostly written in Matlab/Octave language, with some parts implemented in C++. The first use case of the toolkit is evaluation of multiple trackers in multiple experimental scenarios on a fixed set of sequences. By default the entire evaluation is performed sequentially as described by the pseudo-code in Figure C.1. Each trial contains one or more executions of the tracker. The idea is that if the tracker fails during tracking the execution (the failure criterion can be experiment dependent) it is repeated from the point of the failure (plus additional offset frames if specified). In the case of stochastic trackers, each sequence is evaluated multiple times. If the tracker produces identical trajectories two times in a row, the tracker is considered deterministic and further iterations are omitted. It is therefore important that the stochastic nature of a tracker is appropriately addressed (proper random seed initialization).

The second use case of the toolkit is results analysis. Results of more than one tracker can be used to generate performance reports, such as ranking report (as described in Section 3.4), and many others. The code of the VOT toolkit is available on Github: `https://github.com/vicoslab/vot-toolkit`.

```
tracker t
for experiment e = e1 to eE do
    for sequence s = s1 to sS do
        st = transfrom sequence according to e
        while repeat r times (if tracker is stochastic) do
            perform trial for (t, e, st)
        end while
    end for
end for
```

Figure C.1

A pseudo-code of an experiment stack execution in the VOT toolkit.

C.3 Aibu annotator

Aibu is an image sequence annotator written in Java. Its main goal is to enable easy and fast annotation of single-target sequences, although the architecture supports easy extension to multi-target sequences. The user interface (Figure C.2) supports easy navigation to an arbitrary position in a sequence, intuitive region editing and productivity utilities, such as region interpolation. The editor supports annotation storage format that is compatible with the VOT toolkit.

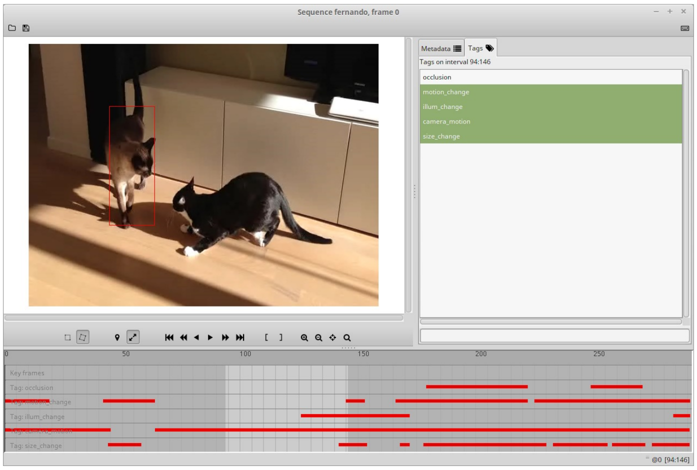
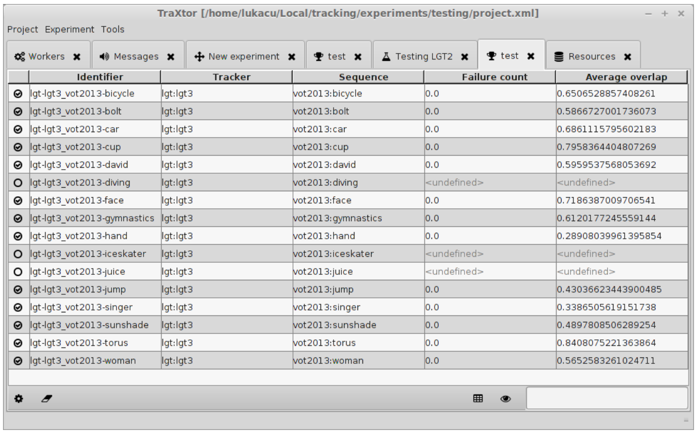


Figure C.2

A screen-shot of the Aibu annotator.

C.4 TraXtor

Similarly to VOT toolkit, TraX is a visual tracking performance evaluation tool. It is written in Java and supports TraX protocol for easy tracker integration. Contrary to VOT toolkit, TraXtor is primarily designed for fast exploratory evaluation. It can be used to quickly evaluate a new tracker on various sequences and change the parameters of the tracker without recompiling the code, as seen in Figure C.3. This functionality enables quick exploratory parameter investigation in order to gain a deeper understanding of the tracking algorithm and discover problematic behavior. On multi-core computers TraXtor supports parallel execution of trackers, which, in combination with the responsiveness of the communication protocol provides a powerful tool for tracker development and testing.



	Identifier	Tracker	Sequence	Failure count	Average overlap
✓	lgt-lgt3_vot2013-bicycle	lgt:lgt3	vot2013:bicycle	0.0	0.6506528857408261
✓	lgt-lgt3_vot2013-bolt	lgt:lgt3	vot2013:bolt	0.0	0.5866727001736073
✓	lgt-lgt3_vot2013-car	lgt:lgt3	vot2013:car	0.0	0.6861115795602183
✓	lgt-lgt3_vot2013-cup	lgt:lgt3	vot2013:cup	0.0	0.7958364404807269
✓	lgt-lgt3_vot2013-david	lgt:lgt3	vot2013:david	0.0	0.5959537568053692
○	lgt-lgt3_vot2013-diving	lgt:lgt3	vot2013:diving	<undefined>	<undefined>
✓	lgt-lgt3_vot2013-face	lgt:lgt3	vot2013:face	0.0	0.7186387009706541
✓	lgt-lgt3_vot2013-gymnastics	lgt:lgt3	vot2013:gymnastics	0.0	0.6120177245559144
✓	lgt-lgt3_vot2013-hand	lgt:lgt3	vot2013:hand	0.0	0.28908039961395854
○	lgt-lgt3_vot2013-iceskater	lgt:lgt3	vot2013:iceskater	<undefined>	<undefined>
○	lgt-lgt3_vot2013-juice	lgt:lgt3	vot2013:juice	<undefined>	<undefined>
✓	lgt-lgt3_vot2013-jump	lgt:lgt3	vot2013:jump	0.0	0.43036623443900485
✓	lgt-lgt3_vot2013-singer	lgt:lgt3	vot2013:singer	0.0	0.3386505619151738
✓	lgt-lgt3_vot2013-sunshade	lgt:lgt3	vot2013:sunshade	0.0	0.4897808506289254
✓	lgt-lgt3_vot2013-torus	lgt:lgt3	vot2013:torus	0.0	0.8408075221363864
✓	lgt-lgt3_vot2013-woman	lgt:lgt3	vot2013:woman	0.0	0.5652583261024711

Figure C.3

A screen-shot of the TraXtor interface. A project can organize multiple experiments that either compare multiple trackers or a tracker on a set of parameter values.

C.5 Legit library

Legit library is a C++ library that contains several visual tracker implementations within a common interface (API). The library is accompanied by a command-line utility that enables easy tracker testing on a various sequence formats, such as series of stored images or a video file and even a video stream from a camera. The command-line utility also supports TraX communication protocol for fast integration with client tools, such as TraXtor or VOT toolkit. In addition the library also provides bindings to other programming languages, e.g. Java and Python. The code is available on Github: `https://github.com/vicoslab/legit`.

Razširjeni povzetek

D

V doktorski disertaciji obravnavamo vizualno sledenje, ki je eno izmed pomembnih raziskovalnih podpodročij v okviru računalniškega vida. Glavni cilj vizualnega sledenja je določitev stanja enega ali več objektov v toku slik ob upoštevanju časovne soslednosti le-teh. Razviti algoritmi, ki opravljajo nalogo vizualnega sledenja in jih imenujemo tudi *vizualni sledilniki*, so lahko uporabljeni v okviru mnogih, tako novih kot tudi že uveljavljenih, tehnoloških področij, kot so npr. robotika [1], video-nadzorni sistemi [2, 3], interakcija med človekom in računalnikom [9–11], avtonomna vozila ter analiza športa [14]. Zaradi široke palete možnosti uporabe vizualnega sledenja se je razvilo veliko podvrst formalizacije problema, vsaka s svojimi izzivi in predpostavkami. V okviru te doktorske disertacije naslavljamo tip vizualnega sledenja, kjer sledimo samo enemu objektu v enem samem toku slik. Geometrijskih lastnosti objekta ne poznamo vnaprej, predpostavljamo tudi, da objekt ne bo nikoli izginil iz opazovanega območja v sliki, čemur pravimo tudi *kratkoročno sledenje*, ter da je tok slik potencialno neskončen in ga torej ne moremo shraniti in nato obdelati v celoti. Glavni cilj take formalizacije vizualnega sledenja je torej določitev položaja objekta v zaporedju slik, če imamo podan začeten položaj v prvi sliki zaporedja. Vizualni sledilniki za dosego cilja naloge uporabljajo različne *modele izgleda*, ki na različne načine opisujejo izgled objekta. Ker se le-ta tekom sekvence spreminja, je potrebno model izgleda posodabljati, to pa pogosto predstavlja problem, saj neuspešna posodobitev, ki je lahko rezultat netočne lokalizacije ali toge zasnove vizualnega modela, vodi v počasno spiralo odklona opisa izgleda objekta od realnega stanja, to pa pripelje do odpovedi sledilnika oziroma *zdrsa*.

V predlagani doktorski disertaciji naslavljamo dve pomembni vprašanji v okviru vizualnega sledenja. Predlagamo nov koncept konstrukcije vizualnega modela, ki temelji na hierarhičnemu združevanju vizualnih informacij. Tak model izgleda nudi možnosti za uspešno sledenje v mnogih težkih scenarijih, še posebej pa je primeren za sledenje netogih in artikuliranih objektov. Uporabo vizualnega modela smo potrdili z razvojem dveh sledilnikov, ki temeljita na hierarhični zasnovi in ki se, glede na empirične primerjave, uvrščata v sam vrh raziskav na tem področju. Poleg tega predlagamo tudi novo metodologijo za evaluacijo vizualnih sledilnikov, tako z namenom primerjave več sledilnikov kot tudi za pridobivanje dodatnih informacij o delovanju določenega sledilnika. V okviru doktorske disertacije torej opisujemo naslednja prispevka k raziskovalnemu področju računalniškega vida:

- *Novi algoritmi za vizualno sledenje z združevanjem lokalne in globalne vizualne* *informacije.* Predstavljamo dva sledilnika, ki vsebujeta nov tip hierarhičnega vizualnega modela in ki omogočata sledenje tarčam tudi v primeru netogih deformacij ter drugih sprememb izgleda. Analiziramo obnašanje predlaganega sledilnika v različnih scenarijih sledenja ter ga primerjamo z drugimi znanimi sledilniki. Poleg tega predlagamo razširitve vizualnega modela, ki bodo na intuitiven način omogočale integracijo predhodnih informacij o izgledu objekta, vendar še vedno omogočale modelu, da se prilagaja spremembam izgleda objekta med samim sledenjem.

- *Nova metodologija za ocenjevanje vizualnih sledilnikov.* Naslavljamo dva problema, ki se pojavljata pri ocenjevanju in primerjavi vizualnih sledilnikov: (i) definiramo reprezentativno množico mer uspešnosti, ki jih lahko uporabimo za opis različnih aspektov sledenja, (ii) predlagamo metodologijo za eksperimentalno primerjavo velikega števila sledilnikov. Predlagana metodologija je poleg teoretičnega opisa podprta tudi z implementacijo v obliki odprtokodnega evaluacijskega okolja.

D.1 Pregled področja

Vizualne modele lahko razvrstimo glede na tip uporabljenih vizualnih značilnic za opis objekta in glede na način hranjenja ter obdelave informacij o izgledu [27]. Najbolj razširjena vrsta vizualnih modelov so *holistični* vizualni modeli, ki hranijo globalno reprezentacijo izgleda objekta, kar se je izkazalo za dovolj dobro strategijo v scenarijih sledenja, kjer se objekt ne deformira preveč. Pogosto uporabljene značilnice, ki se uporabljajo v takih modelih, so barvni histogrami [9, 32, 33], slikovne predloge [28–31, 48], obrisi [36] in tekstura [37]. Pogosto uporabljene metode iskanja maksimalnega ujemanja vizualnega modela s sliko uporabljajo sekvenčno jedrno [9, 32] ter Monte-Carlo [33, 39] optimizacijo. V zadnjem desetletju je postalo popularno sledenje z uporabo diskriminativnih modelov. V tem primeru vizualni model vsebuje diskriminativni klasifikator, ki določi, če določena regija vsebuje objekt ali ne. Ta klasifikator je med sledenjem sproti osveževan, da ohrani dobro razlikovanje med izgledom objekta ter izgledom ozadja. Ena izmed prvih uspešnih implementacij sledenja z uporabo detekcije je predstavljena v [40]. V tem primeru je bil uporabljen kaskadni ojačevalni (*boosting*) detektor [51], prirejen za sprotno osveževanje. Pristop je bil kasneje razširjen na delno-nadzorovano učenje [50] ter učenje z množicami primerkov (*multiple instance learning*) [42]. V [41] avtorji predstavijo zaokrožen postopek modeliranja vizualne informacije ter sledenja z uporabo metode strukturiranih podpornih vektorjev. Avtorji v [49] predlagajo uporabo naključnih projekcij za kompresijo prostora značilnic, kar ugodno vpliva na obvladljivost problema diskriminacije. Kljub očitnemu uspehu holističnih vizualnih modelov pa hitre spremembe strukture objekta še vedno predstavljajo velik izziv. V primeru holističnih modelov je namreč celotna reprezentacija izgleda objekta osvežena naenkrat, kar povečuje verjetnost, da bo pravilen del vizualne informacije pokvarjen z na novo pridobljeno informacijo. To se lahko zgodi, ker sledilnik ne uspe določiti pravilnega položaja objekta, kar pomeni, da bo vizualni model osvežen z izgledom, ki ne pripada ozadju, ali ker sledilnik ne uporablja značilnic, ki bi bile v danem scenariju zmožne razločevati objekt od ozadja. Drugi problem holističnih vizualnih modelov je predpostavka, da objekt lahko opišemo s pravokotno regijo v sliki. Kljub temu, da je to smiselna predpostavka v mnogih praktičnih primerih (npr. sledenje obrazov ali avtomobilov), obstaja veliko scenarijev, kjer ta predpostavka ne drži, npr. pri netogih in artikuliranih objektih. Vse geometrijske deformacije tarče, ki bi jih lahko naslovili v geometrijskem okviru, morajo biti v holističnem vizualnem modelu naslovljene s korakom osveževanja, kar povečuje možnost drsenja (*drifting*). En izmed načinov naslavljanja nekaterih pomanjkljivosti posameznih holističnih sledilnikov je njihovo združevanje [44, 57, 59]. Ideja v tem primeru je, da se vsak sledilnik obnaša dobro v določenih okoliščinah in da lahko s pametnim preklapljanjem med njimi izboljšamo njihovo skupno delovanje. A tudi ta pristop dejansko ne naslavlja sledenja netogim objektom, ki se deformirajo in spreminjajo obliko.

Po drugi strani je glavna ideja vizualnih modelov, ki so osnovani na več *delih*, da je izgled razdeljen na več lokalnih vizualnih modelov, ki so med seboj omejeni preko geometrijskih povezav. Dejanski tipi lokalnih vizualnih modelov posameznega dela in oblike geometrijskih omejitev se lahko med sledilniki te družine zelo razlikujejo. Eden izmed zgodnjih primerov na delih osnovanih sledilnikov je bil predlagan v [10] in temelji na množici lokalnih sledilikov, ki sledijo z ocenjevanjem optičnega toka. Sledenje z optičnim tokom je bilo kasneje robustificirano s primerjavo ocene optičnega toka z oceno povratnega optičnega toka, pri čemer je zanesljivost definirana kot podobnost obeh [22, 71]. Nato so lokalne ocene premika združene z robustno oceno mediane. Uporaba stabilnih regij je še en pristop k sledenju z več deli. V [63] avtorji zaznajo stabilne dele ter z predpostavljanjem globalne afine transformacije omejijo iskanje ujemanj ter se izognejo drsenju. Ker je zanesljive ad-hoc geometrijske omejitve med deli težko določiti za vnaprej nepoznan objekt, avtorji v [72] za sledenje predlagajo uporabo posplošenega Houghovega transforma. Ta pristop je razširjen v [60]. V [21, 65] so uporabljene značilnice SIFT, izgled objekta je predstavljen kot množica značilnic, ki se pogosto pojavijo skupaj. Zanimiv pristop k sledenju objektom brez značilne teksture je opisan v [77], kjer sledilnik uporablja pare robov kot značilnice, na podlagi katerih določi položaj celotnega objekta. Ta pristop sicer deluje na objektih brez jasne teksture (npr. prazen list papirja), vendar pa ne omogoča robustnega obravnavanja deformacij objekta. V splošnem je število stabilnih regij odvisno od vizualnih lastnosti specifičnega objekta (npr. jasnosti teksture), to pa neposredno vpliva na uspešnost sledilnika, saj je le-ta odvisna od števila in ponovljivosti stabilnih regij. Če imamo opravka z barvno homogenimi objekti, SIFT značilnice, omenjene v prejšnjem primeru, ne bodo številčne in ponovljive, sledilnik pa bo zato neuspešen.

V [47] avtorji obravnavajo problem postavitve delov v sliko kot optimizacijski problem in predlagajo sledenje objektu s pomočjo množice lokalnih jeder, ki so med seboj povezana preko omejitev v obliki afine transformacije. Avtorji v [61] to omejitev razrahljajo in enotno afino transformacijo razbijejo na lokalne afine transformacije trojic delov. Avtorji v [69] predlagajo polno povezan graf omejitev v kombinaciji s filtrom z delci za uporabo pri sledenju obrazu. V [81] avtorji za zapis prostorskih omejitev med deli uporabijo Markovska naključna polja. Problem vseh omenjenih pristopov je, da morajo biti omejitve ročno nastavljene glede na strukturne lastnosti objekta, kar pa je v mnogo scenarijih nezaželeno. Poleg tega je množica delov v teh vizualnih modelih fiksna in se ne more prilagajati večjim spremembam v izgledu objekta. V [62] avtorji predlagajo sledenje artikuliranim objektom s požrešnim deljenjem segmentacijske maske objekta na več delov. Tej pravokotni deli so generirani za vsako novo sliko iz osvežene segmentacijske maske, ki predpostavlja približno nespremenljiv barvni opis. Bolj prilagodljiv geometrijski model, ki omogoča dolgoročno osveževanje, je predstavljen v [68]. Preprost zvezdast model povezuje posamezne dele, le-te pa lahko s časom dodajamo in odvzemamo. Novi deli so v model dodani z uporabo globalnega barvnega modela, ki je kombiniran z detektorjem stabilnih regij, kar pomeni, da je postopek omejen na teksturirane objekte. Naslednji model, ki uporablja višjenivojski globalni izgled za postavljanje delov, je predstavljen v [72]. V tem primeru je segmentacijski algoritem inicializiran z uporabo najdenih ujemanj lokalnih značilnic, rezultat segmentacije pa je nato uporabljen za učenje novih značilnic. Uspeh tega pristopa je neposredno odvisen od robustnosti segmentacije, ki je v primeru zamegljenih ali šumnih scen dokaj nizka. Bolj preprosta, hitrejša, a tudi manj zanesljiva segmentacija je uporabljena v [60]. V tem primeru je vsak slikovni element za pripadnost objektu obravnavan ločeno. Uspešnost vseh teh pristopov kaže na uporabnost visokonivojske informacije, saj le-ta omogoča daljšo življenjsko dobo sledilnikov, ki temeljijo na kombinaciji lokalnih opisov v scenarijih, kjer se izgled objekta spreminja. Kljub temu pa ostaja mehanizem integracije globalne in lokalne informacije o izgledu objekta le delno raziskan.

Razvoj na področju vizualnega sledenja poteka z bliskovito hitrostjo in vsako leto je predstavljenih na desetine novih sledilnikov. Če hočemo nov sledilnik ustrezno preučiti in ga kritično ovrednotiti s primerjavo z ostalimi sledilniki na področju, je pomembno, da izberemo standardno množico testnih sekvenc, standarden evaluacijski protokol in informativne mere performans. Na žalost na področju vizualnega sledenja trenutno ni soglasja glede izbire le-teh. Večina znanstvenih objav, ki obravnavajo metodologijo za ocenjevanje vizualnih sledilnikov, se ukvarja s sledenjem več objektom [85–87]. Na prvi pogled izgleda sledenje več objektom kot pospološitev sledenja enemu objektu, vendar se v primeru sledenja več objektom bolj osredotočimo na merjenje pravilnosti dodeljevanja identitete posameznemu objektu v vnaprej določeni domeni, npr. sledenje ljudem ali avtomobilom [17, 18], živalim [20] ali sledenje v športu [14], ne pa na lastnosti, ki jih ima sledilnik v okviru sledenja posameznega objekta. Ocenjevanje sledenja enemu samemu objektu se osredotoča ravno na to: na natančnost, robustnost, pa tudi na splošno uporabnost posameznega vizualnega sledilnika. Glavni cilj je ocenjevanje uspešnosti v raznolikih scenarijih (različni tipi osvetlitve, gibanje kamere, šum itd.). Iz tega cilja izhajajo avtorji v [66], ki primerjajo množico sledilnikov z uporabo povprečne napake središča ter mere povprečnega prekrivanja. Njihova študija je osredotočena primarno na odkrivanje pozitivnih in negativnih lastnosti omejenega števila sledilnikov. Avtorji v [91] razširijo množico sledilnikov in testnih sekvenc. Velikost njihovega eksperimenta je impresivna, vendar njihova izbira mer ni posrečena, kar je razvidno iz skope kvalitativne analize rezultatov. V [92] avtorji predstavijo sistem za evaluacijo sledilnikov, ki lahko simulira omejeno število scenarijev degradacije video sekvenc. Poleg tega predlagajo tudi novo mero ocenjevanja performans, vendar je ne podprejo s teoretično analizo. V [96] je predstavljena zanimiva ideja zbiranja obstoječih primerjalnih eksperimentov iz različnih virov za njihovo skupno povzemanje, ki naj bi zmanjšalo možno pristranskost posameznih eksperimentov. Avtorji tudi sami priznavajo, da ta pristop ni primeren za vrednotenje novih sledilnikov, poleg tega pristop ne naslavlja vpliva korelacije različnih mer. V [93] avtorji predstavijo eksperimentalen povzetek raznolike množice nedavno predstavljenih sledilnikov skupaj z analizo nekaterih mer performans. Zaradi omejene množice mer za analizo je njihova končna izbira že od začetka bolj naklonjena diskriminativnim sledilnikom, kar vpliva tudi na končno izbiro uporabljenih mer ter posledično na interpretacijo rezultatov. Vsa omenjena dela jasno kažejo na pomen dobre in razumljive evaluacije sledilnikov, vendar pa v nobenem izmed njih niso naslovljeni vsi pogoji za tako evaluacijo.

Za objektivno in temeljito evaluacijo potrebujemo tudi orodja, ki nam omogočajo delno-avtomatsko opravljanje eksperimentov. V preteklosti sta bili predlagani dve taki orodji, ODViS [19] ter ViPER [97]. Prvi sistem je narejen za sledenje v sistemih za nadzor, pri drugem pa gre za skupek orodij za anotacijo sekvenc in naknadno računanje različnih mer. Kasneje je bilo predstavljenih tudi nekaj drugih podobnih orodij (npr. [91, 92]), vendar so vsa ta orodja omejena na določen evaluacijski protokol ter omejeno množico vnaprej prilagojenih sledilnikov. Noben izmed omenjenih sistemov ne nudi fleksibilne, robustne in avtomatske evaluacije večjega števila sledilnikov, niti ne omogoča preproste in hitre integracije poljubnega sledilnika. Te lastnosti so ključne za širše sprejemanje orodja ter posledično standardizirano evaluacijo, ki bi raziskovalcem omogočala konsistetno primerjavo njihovih sledilnikov.

D.2 Hierarhični vizualni model

Naše delo v okviru izdelave robustnega vizualnega sledilnika temelji na fuziji obeh glavnih paradigem zasnove vizualnih modelov, se pravi holističnega načina opisa izgleda v kombinaciji z opisom z deli, saj holistični vizualni modeli niso primerni za vse scenarije sledenja. V tej doktorski disertaciji predstavljamo novo formalizacijo vizualnega modela, ki mu pravimo hierarhični vizualni model. Motivacija za hierarhični opis izgleda objekta izhaja iz želje po prostorskem in časovnem strukturiranju teh podatkov, kar omogoča vizualnemu modelu, da je dovolj specifičen, da lahko lokalizira objekt v sliki, obenem pa tudi dovolj prožen, da se lahko hitro prilagodi poljubni spremembi izgleda objekta. Konceptualno je hierarhični model definiran kot množica plasti, vsaka izmed njih opisuje izgled na drugačen način. Spodnja plast vsebuje najbolj jasno informacijo o trenutnem izgledu objekta, višje plasti pa informacijo o bolj splošnem izgledu, ki je manj odvisen od trenutka v času. Funkcija posameznih plasti se odraža tudi v osveževanju vizualnega modela. Spodnje plasti so pri svojem osveževanju vodene s strani višjeležečih plasti, višje plasti pa so osveževane z izluščeno in posplošeno vizualno informacijo spodnjih plasti, če je le-ta dovolj zanesljiva. Če informacija v nekem trenutku ni zanesljiva, se osveževanje višjih plasti ustavi, plasti pa so tako zaščitene pred drsenjem in lahko z vodenjem osveževanja spodnjih plasti pripomorejo k okrevanju celotnega vizualnega modela.

Hierarhični vizualni model nudi odprt in prožen teoretični okvir, ki lahko služi kot vodilo za razvoj bolj robustnih sledilnikov. Spodnja plast je najbližje trenutnemu izgledu objekta, vendar se mora neprestano spreminjati in prilagajati spremembam v sliki. To lahko dosežemo z uporabo vizualnega modela z visoko stopnjo prostih parametrov, kot je prožna konstelacija delov, vendar pa se lahko pri taki predstavitvi na dolgi rok hitro pojavijo problemi. Prav pri tem pridejo do izraza višje plasti vizualnega modela, ki nudijo spodnji plasti vodenje, na primer z odvzemanjem zastarelih delov ter dodajanjem novih.

V doktorski disertaciji predstavljamo dva vizualna modela, ki sledita ideji hierarhične organizacije vizualne informacije. Prvi model imenujemo *sklopljeni vizualni model*. Gre za vizualni model, ki izgled objekta opisuje v dveh plasteh in tako združuje lokalno in globalno predstavitev izgleda objekta. Spodnja plast je sestavljena iz več med seboj povezanih delov, ki so se sposobni prilagajati geometrijskim spremembam netogih objektov, zgornja plast pa vsebuje večmodalno globalno predstavitev izgleda, ki vodi proces posodobitve spodnje plasti. Prilagajanje spodnje plasti je formalizirano kot stohastična optimizacija, ki upošteva tako vizualno podobnost delov s sliko kot tudi geometrijske omejitve med deli. Ker določeni deli objekta med dolgoročnim sledenjem izginejo, pokažejo pa se lahko novi, je osveževanje množice delov nujno za uspeh sledilnika. Odvzemanje in oddajanje delov je vodeno preko zgornje plasti, ki določi področja v sliki, ki, glede na vizualno informacijo, z veliko verjetnostjo pripadajo objektu. Prav tako se preko dobrih delov iz spodnje plasti osvežuje tudi zgornja – obe plasti lahko vzajemno osvežujeta ena drugo. Opisani vizualni model smo uporabili za razvoj sledilnika. Delne rezultate teh raziskav smo kot del znanstvene diseminacije v okviru doktorske disertacije objavili v večih objavah [106, 107], v katerih smo pokazali, da je predlagana kombinacija lokalne in globalne informacije smiselna, sledilnik namreč zagotavlja robustno in računsko učinkovito sledenje objektom, še posebno pa se vizualni model izkaže pri sledenju objektov, ki se netogo deformirajo. S preprosto implementacijo opisanih idej smo v eksperimentalni primerjavi presegli najboljše sledilnike v časovnem obdobju objave člankov, kot prikazujemo v eksperimentih, opravljenih v okviru doktorske disertacije, pa se lahko uspešno kosa tudi z novejšimi pristopi. Po drugi strani pa analiza razkrije tudi nekaj pomanjkljivosti modela, ki se kažejo v nizki natančnosti sledenja, še posebno v primerih, ki so dokaj preprosti za sledilnike, ki objekt ocenjujejo z manj parametri, le-te pa lahko tako ocenijo bolj natančno. V okviru analize smo predlagali nekaj možnosti za izboljšave, ki smo jih naslovili v drugem opisanem vizualnem modelu.

Drugi prestavljeni vizualni model razširja hierarhijo s tretjo plastjo, uvaja pa tudi koncept sidrnih predlog. Prvi dve plasti drugega vizualnega modela sta konceptualno zelo podobni prvemu vizualnemu modelu. V spodnji plasti za iskanje ujemanja namesto stohastične optimizacije uporabljamo deterministično, ki pa jo inicializiramo z uporabo optičnega toka v posameznih delih. Srednja plast je, podobno kot v prvem vizualnem modelu, namenjena določanju področij v sliki, ki pripadajo objektu, le-to pa dosežemo s preprosto in hitro segmentacijo na podlagi barvnega modela. Tretja plast vsebuje spominski sistem statičnih predlog, ki vizualnemu modelu nudijo zanesljivo informacijo o položaju in velikosti objekta v primeru dobrega ujemanja ene izmed predlog s sliko. Na ta način tretja plast pripomore k hitremu okrevanju celotnega vizualnega modela. Predstavljena eksperimentalna analiza koristi tretje plasti potrdi, saj sledilnik s tem vizualnim modelom izboljša natančnost, pa tudi splošno kvaliteto sledenja.

D.3 Metodologija za primerjavo sledilnikov

Zaradi kompleksnosti algoritmov za vizualno sledenje in sekvenčne narave sledenja je ocenjevanje kvalitete posameznih algoritmov netrivialna naloga, katere pomembnost je šele pred kratkim pritegnila širšo pozornost raziskovalne skupnosti. Zanesljivo ocenjevanje kvalitete sledilnikov je pomembno tako za kritično ovrednotenje napredka v raziskavah kot tudi za interpretacijo delovanja različnih pristopov pri razvoju sledilnikov, pomanjkanje soglasja pa upočasnjuje napredek na tem področju. Ključno je predvsem soglasje glede mer performans ter metodologije, pa tudi glede množic posnetkov, na katerih se algoritme primerja, v večini trenutnih del je izbira vseh treh komponent bolj ali manj poljubna.

V okviru te doktorske disertacije zato naslavljamo problem izbire enotne metodologije za kritično ocenjevanje performans kartkoročnih sledilnikov. V nasprotju s prevladujočimi trendi v zadnjih desetletjih trdimo, da lastnosti vizualnih sledilnikov ni mogoče opisati z eno samo mero uspešnosti, po drugi strani pa tudi ne smemo uporabiti poljubne množice mer, za katere ne poznamo medsebojnih odnosov. Razvoj nove evalucijske metodologije smo pričeli s kritično analizo mer performans ter evalucijskih protokolov. V ta namen smo opravili teoretično in empirično analizo mer, ki so pogosto uporabljene za ocenjevanje zmogljivosti vizualnih sledilnikov. Analiza obsega tako pogosto uporabljane ocene, ki se uporabljajo za primerjavo sledilnikov, kot tudi bolj opisne mere, npr. različne tipe grafov. V naši raziskavi smo s pregledom in analizo pokazali, da nekatere izmed mer odražajo iste kvalitete ali pa so celo teoretično ekvivalentne. Na temelju te analize smo predlagali par dveh šibko koreliranih mer, ki odražata natančnost in robustnost sledilnega algoritma. Natančnost sledilnika se odraža v povprečnem prekrivanju regije objekta v anotacijah z regijo, ki jo za objekt predlaga sledilnik, robustnost pa se odraža v številu odpovedi sledilnika, ko je le-ta zdrsnil s tarče in ga je bilo potrebno ponovno inicializirati. Predlagamo tudi ustrezen prikaz takih rezultatov v obliki dvodimenzionalnega točkastega grafa ter analizo celotne metodologije s pomočjo predlaganih teoretičnih sledilnikov, ki izražajo ekstremno obnašanje sledilnih algoritmov, na primer določitev celotne slike za regijo objekta ali neprestano odpovedovanje. Poleg tega predlagamo tudi indikator fragmentacije, ki odraža razporeditev odpovedi po sekvenci, ter pokažemo, kako se lahko teoretične sledilnike uporabi za določanje preprostih lastnosti sekvenc, kar nam olajša interpretacijo rezultatov. Izbiro mer nato nadgradimo še z metodologijo razvrščanja večjega števila sledilnikov. Pri tem upoštevamo morebitno stohastično naravo sledilnikov, kar pomeni, da moramo sledilnik na isti sekvenci pognati večkrat, saj rezultati ne bodo vedno enaki. Poleg tega pa pri določitvi vrstnega reda upoštevamo tudi statistično ekvivalenco sledilnikov.

Primerjava večjega števila sledilnikov na veliki množici sekvenc je zapletena naloga, ki pa mora biti opravljena ponovljivo in brez napak. V ta namen smo v okviru disertacije implementirali odprtokodno okolje, ki je zmožno samodejno opraviti eksperimente na podlagi naše predlagane metodologije, obdelati rezultate ter generirati informativna poročila. V okviru tega okolja naslavljamo tudi problem hitre integracije različnih implementacij sledilnikov, spisanih v različnih programskih jezikih. V ta namen predlagamo preprost komunikacijski protokol, ki za medij komunikacije med sledilnikom in evalvacijskim okoljem uporablja standardne koncepte operacijskih sistemov, kot so datoteke ter vhodno-izhodni tokovi. Glavno vodilo za načrtovanje protokola je preprostost integracije, ki bo raziskovalcem omogočala hitro vključitev podpore v lastno kodo. Na ta način se lahko raziskovalci ukvarjajo predvsem z razvojem novih metod, po drugi strani pa lahko nova orodja za analizo sledilnikov hitro postanejo dostopna in uporabna široki množici uporabnikov. Orodje za primerjavo sledilnikov, preko njega pa tudi predlagano metodologijo, sedaj uporabljajo tudi v okviru delavnic in tekmovanj *Visual Object Tracking (VOT) challenge*.

D.4 Zaključek

V doktorski disertaciji smo obravnavali problem vizualnega sledenja, v okviru dela pa smo predstavili dva velika prispevka k znanosti:

- *Nov algoritem za vizualno sledenje z združevanjem lokalne in globalne vizualne informacije za sledenje netogih artikuliranih objektov.* Predstavljamo in analiziramo dva sledilnika, ki vsebujeta nov tip hierarhičnega vizualnega modela in ki omogočata sledenje tarčam tudi v primeru netogih deformacij ter drugih sprememb izgleda.

- *Nova metodologija za ocenjevanje vizualnih sledilnikov.* Definiramo reprezentativno množico mer uspešnosti, ki jih lahko uporabimo za opis različnih aspektov sledenja ter predlagamo metodologijo za eksperimentalno primerjavo velikega števila sledilnikov.

Ob tem je potrebno poudariti, da je problem vizualnega sledenja sam po sebi zelo slabo definiran, saj sledenje stanju poljubnega objekta zahteva integracijo veliko večje količine znanja, kot je samo trenutni izgled objekta. Da bi lahko poljuben objekt lahko zanesljivo sledili v poljubni situaciji, kjer je tega zmožen človek, bi moral sistem integrirati podsisteme iz več področij računalniškega vida in sklepanja, kar daleč presega trenutno stanje na tem raziskovalnem področju. Po drugi strani pa že sedaj obstaja veliko možnosti uporabe vizualnega sledenja v okviru določenih aplikacij, kjer je scenarij sledenja bolj definiran in omejen. Prav med tema dvema pogledoma vidimo veliko priložnost hierarhičnih vizualnih modelov, saj nudijo teoretični okvir, ki omogoča po eni strani postopen prehod iz problema sledenja na druge domene računalniškega vida, kot sta kategorizacija in detekcija, po drugi strani pa na podoben način omogoča tudi intuitivno uvajanje omejitev, ki izvirajo iz aplikacije.

Napredek v vizualnem sledenju sloni tudi na razumljivi empirični analizi teoretičnih modelov. V tem okviru je pomembno poudariti, da predlagana metodologija za analizo in primerjavo kratkoročnih sledilnikov pridobiva prepoznavnost v raziskovalni skupnosti, predvsem po zaslugi tekmovanja VOT, ki predstavlja platformo za razvoj in promocijo konsistentne metodologije, preko tega pa za pospešen razvoj področja vizualnega sledenja. Največji izziv, s katerim se soočajo iniciativa VOT ter podobne ideje, je, kako se upreti skušnjavi redukcije opisa lastnosti sledilnih algoritmov na eno samo številčno oceno ter nesmiselni bitki za njeno izboljšavo. Namesto tega moramo priznati raznovrstnost pristopov ter sprejeti smiselne interpretacije prednosti in slabosti le-teh, k temu razumevanju pa bo najverjetneje prispevala tudi konsolidacija evaluacijskih orodij, ki bodo omogočala temeljito analizo posameznih algoritmov ter s tem njihovo nadgradnjo ter napredek celotnega področja.

BIBLIOGRAPHY

[1] N.P. Papanikolopoulos, P.K. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision. *IEEE Trans. Robot. Autom.*, 9(1):14–35, 1993. ISSN 1042296X. doi: 10.1109/70.210792.

[2] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, volume 2, pages 246–252. IEEE Comput. Soc, 1999. ISBN 0-7695-0149-4. doi: 10.1109/CVPR.1999.784637.

[3] Pakorn KaewTrakulPong and Richard Bowden. A real time adaptive visual surveillance system for tracking low-resolution colour targets in dynamically changing scenes. *Image Vis. Comput.*, 21(10):913–929, September 2003. ISSN 02628856. doi: 10.1016/S0262-8856(03)00076-3.

[4] Robert T Collins, Alan Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, Osamu Hasegawa, Peter Burt, and Others. *A system for video surveillance and monitoring*, volume 2. Carnegie Mellon University, The Robotics Institute Pittsburg, Pittsburgh, 2000.

[5] Ismail Haritaoglu, David Harwood, and Larry S Davis. W4: Real-time surveillance of people and their activities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8): 809–830, 2000.

[6] Osama Masoud and Nikolaos P Papanikolopoulos. A novel method for tracking and counting pedestrians in real-time using a single camera. *Veh. Technol. IEEE Trans.*, 50(5):1267–1278, 2001.

[7] Jen-Chao Tai, Shung-Tsang Tseng, Ching-Po Lin, and Kai-Tai Song. Real-time image tracking for automatic traffic monitoring and enforcement applications. *Image Vis. Comput.*, 22(6):485–501, 2004.

[8] V. I Pavlovic, R. Sharma, and T. S Huang. Visual interpretation of hand gestures for human-computer interaction: a review. *Pattern Anal. Mach. Intell. IEEE Trans.*, 19(7):677–695, July 1997. ISSN 0162-8828. doi: 10.1109/34.598226.

[9] Gary R. Bradski. Real Time Face and Object Tracking as a Component of a Perceptual User Interface. In *Winter Conf. Appl. Comput. Vis.*, page 214. IEEE Computer Society, 1998. ISBN 0-8186-8606-5.

[10] M. Kölsch and M. Turk. Fast 2D Hand Tracking with Flocks of Features and Multi-Cue Integration. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, volume 10, page 158, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2158-4.

[11] Jesse Hoey, A. von Bertoldi, P. Poupart, and A. Mihailidis. Tracking using flocks of features, with application to assisted handwashing. In *Br. Mach. Vis. Conf.*, pages 367–376, 2006.

[12] Stavros Paschalakis and Miroslaw Bober. Real-time face detection and tracking for mobile videoconferencing. *Real-Time Imaging*, 10(2):81–94, 2004.

[13] Zhen Jia, Arjuna Balasuriya, and Subhash Challa. Sensor fusion-based visual target tracking for autonomous vehicles with the out-of-sequence measurements solution. *Rob. Auton. Syst.*, 56(2):157–176, February 2008. ISSN 09218890. doi: 10.1016/j.robot.2007.05.014.

[14] Matej Kristan, Janez Perš, Matej Perše, and Stanislav Kovačič. Closed-world tracking of multiple interacting targets for indoor-sports applications. *Comput. Vis. Image Underst.*, 113(5):598–611, May 2009. ISSN 10773142. doi: 10.1016/j.cviu.2008.01.009.

[15] J P Lewis. Fast normalized cross-correlation. In *Vis. interface*, volume 10, pages 120–123, 1995.

[16] Thomas Sikora. The MPEG-4 video standard verification model. *Circuits Syst. Video Technol. IEEE Trans.*, 7(1):19–31, 1997.

[17] J Black, T Ellis, and P Rosin. A novel method for video tracking performance evaluation. In *Vis. Surveill. Perform. Eval. Track. Surveill.*, pages 125–132, 2003.

[18] Lisa M. Brown, Andrew W. Senior, Ying-li Tian, Jonathan Connel, Arun Hampapur, Chiao-Fe Shu, Hans Merkl, and Max Lu. Performance evaluation of surveillance systems under varying conditions. In *Perform. Eval. Track. Surveill.*, pages 1–8, 2005.

[19] C Jaynes, S Webb, RM Steele, and Q Xiong. An open development environment for evaluation of video surveillance systems. In *Perform. Eval. Track. Surveill.*, 2002.

[20] Z. Khan, T. Balch, and F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27: 1805–1819, 2005.

[21] Federico Pernici, Alberto Del Bimbo, and Alberto Del Bimbo. Object Tracking by Oversampling Local Features. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(12):2538–2551, 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.250.

[22] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422, 2012.

[23] Seunghoon Hong and Bohyung Han. Visual tracking by sampling tree-structured graphical models. In *Eur. Conf. Comput. Vis.*, pages 1–16. Springer, 2014.

[24] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *Pattern Anal. Mach. Intell. IEEE Trans.*, 30(2):267–282, 2008.

[25] Omar Javed, Khurram Shafique, and Mubarak Shah. Appearance modeling for tracking in multiple nonoverlapping cameras. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, volume 2, pages 26–33. IEEE, 2005.

[26] S. Fidler and A. Leonardis. Towards Scalable Representations of Object Categories: Learning a Hierarchy of Parts. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1–8, 2007. doi: 10.1109/CVPR.2007.383269.

[27] Xi Li, Weiming Hu, Chunhua Shen, Zhongfei Zhang, Anthony Dick, and Anton van den Hengel. A Survey of Appearance Models in Visual Object Tracking. *ACM Trans. Intell. Syst. Technol.*, 4(4):2157–6904, 2013.

[28] G. D. Hager, M. Dewan, and C. V. Stewart. Multiple kernel tracking with SSD. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, volume 1, pages 790–797, 2004. ISBN 1063-6919. doi: 10.1109/CVPR.2004.1315112.

[29] Xue Mei and Haibin Ling. Robust visual tracking and vehicle classification via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(11):2259–72, November 2011. ISSN 1939-3539. doi: 10.1109/TPAMI.2011.66.

[30] Chenglong Bao, Yi Wu, Haibin Ling, and Hui Ji. Real time robust L1 tracker using accelerated proximal gradient approach. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1830–1837. IEEE, June 2012. ISBN 978-1-4673-1228-8. doi: 10.1109/CVPR.2012.6247881.

[31] Yi Wu, Bin Shen, and Haibin Ling. Online robust image alignment via iterative convex optimization. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1808–1814, 2012. doi: 10.1109/CVPR.2012.6247878.

[32] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-Based Object Tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, 2003.

[33] M. Kristan, J. Perš, S. Kovačič, and A. Leonardis. A Local-motion-based probabilistic model for visual tracking. *Pattern Recognit.*, 2008.

[34] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon University, Pittsburgh, 1991.

[35] Stephen J McKenna, Yogesh Raja, and Shaogang Gong. Tracking colour objects using adaptive mixture models. *Image Vis. Comput.*, 17(3):225–231, 1999.

[36] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In Bernard Buxton and Roberto Cipolla, editors, *Eur. Conf. Comput. Vis.*, volume 1064 of *Lecture Notes in Computer Science*, pages 343–356. Springer Berlin Heidelberg, 1996. ISBN 978-3-540-61122-6. doi: 10.1007/BFb0015549.

[37] F. Porikli, O. Tuzel, and P. Meer. Covariance Tracking using Model Update Based on Means on Riemannian Manifolds. Technical report, Mitsubishi Electric Research Laboratories, 2006.

[38] J F Henriques, R Caseiro, P Martins, and J Batista. High-Speed Tracking with Kernelized Correlation Filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2014. doi: 10.1109/TPAMI.2014.2345390.

[39] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-Based Probabilistic Tracking. In *Eur. Conf. Comput. Vis.*, volume 1, pages 661–675. Springer-Verlag, 2002. ISBN 3-540-43745-2.

[40] H. Grabner, M. Grabner, and H. Bischof. Real-Time Tracking via On-line Boosting. In *Br. Mach. Vis. Conf.*, pages 47–56, 2006.

[41] Sam Hare, Amir Saffari, and Philip H. S. Torr. Struck: Structured output tracking with kernels. In *IEEE Int. Conf. Comput. Vis.*, pages 263–270. IEEE, November 2011. ISBN 978-1-4577-1102-2. doi: 10.1109/ICCV.2011.6126251.

[42] B. Babenko, M.-H. Yang, and S. Belongie. Robust Object Tracking with Online Multiple Instance Learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1619–1632, August 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.226.

[43] R. T. Collins, Y. Liu, and M. Leordeanu. Online Selection of Discriminative Tracking Features. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1631–1643, 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.205.

[44] B. Stenger, T. Woodley, and R. Cipolla. Learning to track with multiple observers. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 2647–2654. IEEE, June 2009. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPR.2009.5206634.

[45] Qi Zhao, Zhi Yang, and Hai Tao. Differential earth mover's distance with its applications to visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(2): 274–287, 2010.

[46] C. Yang, R. Duraiswami, and L. Davis. Efficient Mean-Shift Tracking via a New Similarity Measure. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 176–183. IEEE Computer Society, 2005. ISBN 0-7695-2372-2.

[47] Z. Fan, M. Yang, and Y. Wu. Multiple Collaborative Kernel Tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(7):1268–1273, 2007. ISSN 0162-8828.

[48] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental Learning for Robust Visual Tracking. *Int. J. Comput. Vis.*, 77(1-3):125–141, May 2008. ISSN 0920-5691. doi: 10.1007/s11263-007-0075-7.

[49] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time Compressive Tracking. In *Eur. Conf. Comput. Vis.*, 2012.

[50] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *Eur. Conf. Comput. Vis.*, pages 234–247. Springer, 2008.

[51] Paul Viola and Michael Jones. Robust Real-time Object Detection. *Int. J. Comput. Vis.*, 57(2):137–154, 2004.

[52] Liam Ellis, Nicholas Dowson, Jiri Matas, and Richard Bowden. Linear regression and adaptive appearance models for fast simultaneous modelling and tracking. *Int. J. Comput. Vis.*, 95(2):154–179, 2011.

[53] D S Bolme, J R Beveridge, B A Draper, and Y M Lui. Visual object tracking using adaptive correlation filters. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 2544–2550, 2010.

[54] M Danelljan, F Shahbaz Khan, M Felsberg, J Van de Weijer, G Häger, F Shahbaz Khan, and M Felsberg. Accurate scale estimation for robust visual tracking. In *Br. Mach. Vis. Conf.*, 2014.

[55] Kaihua Zhang, Lei Zhang, Qingshan Liu, David Zhang, and Ming-Hsuan Yang. Fast Visual Tracking via Dense Spatio-temporal Context Learning. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Eur. Conf. Comput. Vis.*, volume 8693 of *Lecture Notes in Computer Science*, pages 127–141. Springer International Publishing, 2014. ISBN 978-3-319-10601-4. doi: 10.1007/978-3-319-10602-1_9.

[56] Jakob Santner, Christian Leistner, Amir Saffari, Thomas Pock, and Horst Bischof. PROST - Parallel Robust Online Simple Tracking. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 723–730, San Francisco, CA, USA, 2010.

[57] V. Badrinarayanan, P. Perez, F. Le Clerc, and L. Oisel. Probabilistic Color and Adaptive Multi-Feature Tracking with Dynamically Switched Priority Between Cues. In *IEEE Int. Conf. Comput. Vis.*, pages 1–8, 2007. ISBN 1550-5499. doi: 10.1109/ICCV.2007.4408955.

[58] J. Kwon and K. M. Lee. Tracking by Sampling Trackers. In *IEEE Int. Conf. Comput. Vis.*, pages 1195–1202. IEEE, November 2011. ISBN 978-1-4577-1102-2. doi: 10.1109/ICCV.2011.6126369.

[59] Junseok Kwon and Kyoung M. Lee. Visual tracking decomposition. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1269–1276. IEEE, June 2010. ISBN 978-1-4244-6984-0. doi: 10.1109/CVPR.2010.5539821.

[60] Stefan Duffner and Christophe Garcia. PixelTrack: A Fast Adaptive Algorithm for Tracking Non-rigid Objects. In *IEEE Int. Conf. Comput. Vis.*, December 2013.

[61] B. Martinez and X. Binefa. Piecewise affine kernel tracking for non-planar targets. *Pattern Recognit.*, 41(12):3682–3691, 2008. ISSN 0031-3203. doi: 10.1016/j.patcog.2008.06.001.

[62] S.M. Shahed Nejhum, Jeffrey Ho, and Ming-Hsuan Yang. Online visual tracking with histograms and articulating blocks. *Comput. Vis. Image Underst.*, 114(8):901–914, August 2010. ISSN 1077-3142. doi: 16/j.cviu.2010.04.002.

[63] Z. Yin and R. Collins. On-the-fly Object Modeling while Tracking. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1–8, 2007. doi: 10.1109/CVPR.2007.383237.

[64] Tomáš Vojír and Jirí Matas. Robustifying the Flock of Trackers. In Andreas Wendel, Sabine Sternig, and Martin Godec, editors, *Comput. Vis. Winter Work.*, pages 91–97, Inffeldgasse 16/II, Graz, Austria, 2011. Graz University of Technology. ISBN 978-3-85125-129-6.

[65] F. Tang and H. Tao. Object tracking with dynamic feature graph. In *Perform. Eval. Track. Surveill.*, pages 25–32, 2005. doi: 10.1109/VSPETS.2005.1570894.

[66] Qing Wang, Feng Chen, Wenli Xu, and Ming-Hsuan Yang. An experimental comparison of online object-tracking algorithms. In *SPIE Opt. Eng. Appl.*, pages 81381A–81381A–11, San Diego, 2011. doi: 10.1117/12.895965.

[67] Zhaowei Cai, Longyin Wen, Jianwei Yang, Zhen Lei, and Stan Z Li. Structured visual tracking with dynamic graph. In *Asian Conf. Comput. Vis.*, pages 86–97. Springer, 2013.

[68] J. S. Kwon and K. M. Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive Basin Hopping Monte Carlo sampling. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1208–1215, 2009. ISBN 1063-6919.

[69] V. Badrinarayanan, F. Le Clerc, L. Oisel, and P. Perez. Geometric Layout Based Graphical Model for Multi-Part Object Tracking. In *Int. Work. Vis. Surveill.*, 2008.

[70] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust Fragments-based Tracking using the Integral Histogram. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, volume 1, pages 798–805. IEEE Computer Society, June 2006. ISBN 0-7695-2597-0. doi: 10.1109/CVPR.2006.256.

[71] Z Kalal, K Mikolajczyk, and Jiří Matas. Forward-Backward Error: Automatic Detection of Tracking Failures. In *Int. Conf. Pattern Recognit.*, pages 2756–2759, 2010. doi: 10.1109/ICPR.2010.675.

[72] Martin Godec, Peter M. Roth, and Horst Bischof. Hough-based tracking of non-rigid objects. In *IEEE Int. Conf. Comput. Vis.*, pages 81–88, Barcelona, November 2011. IEEE. ISBN 978-1-4577-1102-2. doi: 10.1109/ICCV.2011.6126228.

[73] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using SIFT features and mean shift. *Comput. Vis. image Underst.*, 113(3):345–352, 2009.

[74] Son Tran and Larry Davis. Robust Object Tracking with Regional Affine Invariant Features. In *IEEE Int. Conf. Comput. Vis.*, pages 1–8, 2007. ISBN 1550-5499. doi: 10.1109/ICCV.2007.4408948.

[75] Michael Donoser and Horst Bischof. Efficient maximally stable extremal region (MSER) tracking. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, volume 1, pages 553–560. IEEE, 2006.

[76] Wei He, T Yamashita, Hongtao Lu, and Shihong Lao. SURF Tracking. In *IEEE Int. Conf. Comput. Vis.*, pages 1586–1592, September 2009. doi: 10.1109/ICCV.2009.5459360.

[77] Karel Lebeda, Simon Hadfield, Jiri Matas, and Richard Bowden. Long-Term Tracking Through Failure Cases. In *IEEE Int. Conf. Comput. Vis. Work.*, 2013.

[78] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *IEEE Int. Conf. Comput. Vis.*, pages 670–677. IEEE, September 2009. ISBN 978-1-4244-4420-5. doi: 10.1109/ICCV.2009.5459175.

[79] Shu Wang, Huchuan Lu, Fan Yang, and Ming-Hsuan Yang. Superpixel tracking. In *IEEE Int. Conf. Comput. Vis.*, pages 1323–1330. IEEE, November 2011. ISBN 978-1-4577-1102-2. doi: 10.1109/ICCV.2011.6126385.

[80] L. Liu, J. Xing, H. Ai, and S. Lao. Semantic superpixel based vehicle tracking. In *Int. Conf. Pattern Recognit.*, pages 2222–2225, 2012. ISBN 978-1-4673-2216-4.

[81] W.-Y. Chang, C.-S. Chen, and Y.-P. Hung. Tracking by Parts: A Bayesian Approach With Component Collaboration. *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, 39(2):375–388, 2009. ISSN 1083-4419.

[82] Mark Everingham, Luc Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.*, 88(2):303–338, September 2009. ISSN 0920-5691. doi: 10.1007/s11263-009-0275-4.

[83] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A Database and Evaluation Methodology for Optical Flow. *Int. J. Comput. Vis.*, 92(1):1–31, November 2010. ISSN 0920-5691. doi: 10.1007/s11263-010-0390-2.

[84] Sharon Alpert, Meirav Galun, Ronen Basri, and Achi Brandt. Image Segmentation by Probabilistic Bottom-Up Aggregation and Cue Integration. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1–8. IEEE, June 2007. ISBN 1-4244-1179-3. doi: 10.1109/CVPR.2007.383017.

[85] K. Smith, D. Gatica-Perez, and J. Odobez. Evaluating Multi-Object Tracking. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, volume 3, pages 36–36. IEEE, 2005. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.453.

[86] Rangachar Kasturi, Dmitry Goldgof, Padmanabhan Soundararajan, Vasant Manohar, John Garofolo, Rachel Bowers, Matthew Boonstra, Valentina Korzhova, and Jing Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: data, metrics, and protocol. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):319–36, February 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.57.

[87] F. Bashir and F. Porikli. Performance Evaluation of Object Detection and Tracking Systems. In *Perform. Eval. Track. Surveill.*, 2006.

[88] Edward K. Kao, Matthew P. Daggett, and Michael B. Hurley. An information theoretic approach for tracker performance evaluation. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1523–1529. IEEE, September 2009. ISBN 978-1-4244-4420-5. doi: 10.1109/ICCV.2009.5459275.

[89] Pedro Carvalho, Jaime S. Cardoso, and Luís Corte-Real. Filling the gap in quality assessment of video object tracking. *Image Vis. Comput.*, 30 (9):630–640, September 2012. ISSN 02628856. doi: 10.1016/j.imavis.2012.06.002.

[90] Ido Leichter and Eyal Krupka. Monotonicity and Error Type Differentiability in Performance Measures for Target Detection and Tracking in Video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(10):2553–2560, 2013.

[91] Yi Wu, Jongwoo Lim, and Ming-hsuan Yang. Online Object Tracking: A Benchmark. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 2411–2418, 2013.

[92] T. Nawaz and A. Cavallaro. A protocol for evaluating video trackers under real-world conditions. *Image Process. IEEE Trans.*, 22(4):1354—-1361, 2013. ISSN 1057-7149. doi: 10.1109/TIP.2012.2228497.

[93] Arnold W M Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual Tracking: an Experimental Survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1442–1468, 2013. doi: 10.1109/TPAMI.2013.230.

[94] Hao Wu, Aswin C Sankaranarayanan, and Rama Chellappa. Online empirical evaluation of tracking algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1443–58, August 2010. ISSN 1939-3539. doi: 10.1109/TPAMI.2009.135.

[95] Juan C SanMiguel, Andrea Cavallaro, and José M Martínez. Adaptive online performance evaluation of video trackers. *IEEE Trans. Image Process.*, 21(5):2812–23, May 2012. ISSN 1941-0042. doi: 10.1109/TIP.2011.2182520.

[96] Yu Pang and Haibin Ling. Finding the Best from the Second Bests – Inhibiting Subjective Bias in Evaluation of Visual Tracking Algorithms. In *IEEE Int. Conf. Comput. Vis.*, pages 2784–2791, 2013. doi: 10.1109/ICCV.2013.346.

[97] D Doermann and D Mihalcik. Tools and techniques for video performance evaluation. In *Int. Conf. Pattern Recognit.*, pages 167–170, 2000.

[98] Luka Čehovin, Matej Kristan, and Aleš Leonardis. Is my new tracker really better than yours? In *Winter Conf. Appl. Comput. Vis.*, pages 540–547. IEEE, 2014.

[99] Luka Čehovin, Aleš Leonardis, and Matej Kristan. Visual object tracking performance measures revisited. *arXiv Prepr. arXiv1502.05803*, 2015.

[100] Matej Kristan, Roman Pflugfelder, Aleš Leonardis, Jiri Matas, Fatih Porikli, Luka Čehovin, Georg Nebehay, Gustavo Fernandez, Tomáš Vojir, Adam Gatt, Ahmad Khajenezhad, Ahmed Salahledin, Ali Soltani-Farani, Ali Zarezade, Alfredo Petrosino, Anthony Milton, Behzad Bozorgtabar, Bo Li, Chee Seng Chan, Cherkeng Heng, Dale Ward, David Kearney, Dorothy Monekosso, Hakki Can Karaimer, Hamid R. Rabiee, Jianke Zhu, Jin Gao, Jingjing Xiao, Junge Zhang, Junliang Xing, Kaiqi Huang, Karel Lebeda, Lijun Cao, Mario Edoardo Maresca, Mei Kuan Lim, Mohamed El Helw, Michael Felsberg, Paolo Remagnino, Richard Bowden, Roland Goecke, Rustam Stolkin, Samantha Yueying Lim, Sara Maher, Sebastien Poullot, Sebastien Wong, Shin'Ichi Satoh, Weihua Chen, Weiming Hu, Xiaoqin Zhang, Yang Li, and Zhiheng Niu. The Visual Object Tracking VOT2013 challenge results. In *IEEE Int. Conf. Comput. Vis. Work.*, pages 98–111, 2013. ISBN 9780769551616. doi: 10.1109/ICCVW.2013.20.

[101] Matej Kristan, Roman Pflugfelder, Aleš Leonardis, Jiri Matas, Luka Čehovin, Georg Nebehay, Tomáš Vojir, Gustavo Fernández, Alan Lukežič, Aleksandar Dimitriev, Alfredo Petrosino, Amir Saffari, Bo Li, Bohyung Han, CherKeng Heng, Christophe Garcia, Dominik Pangeršič, Gustav Häger, Fahad Shahbaz Khan, Franci Oven, Horst Possegger, Horst Bischof, Hyeonseob Nam, Jianke Zhu, JiJia Li, Jin Young Choi, Jin-Woo Choi, João F Henriques, Joost van de Weijer, Jorge Batista, Karel Lebeda, Kristoffer Öfjäll, Kwang Moo Yi, Lei Qin, Longyin Wen, Mario Edoardo Maresca, Martin Danelljan, Michael Felsberg, Ming-Ming Cheng, Philip Torr, Qingming Huang, Richard Bowden, Sam Hare, Samantha YueYing Lim, Seunghoon Hong, Shengcai Liao, Simon Hadfield, Stan Z Li, Stefan Duffner, Stuart Golodetz, Thomas Mauthner, Vibhav Vineet, Weiyao Lin, Yang Li, Yuankai Qi, Zhen Lei, and ZhiHeng Niu. The Visual Object Tracking VOT2014 challenge results. In *Eur. Conf. Comput. Vis. Work.*, 2014.

[102] Matej Kristan, Jiri Matas, Ales Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A Novel Performance Evaluation Methodology for Single-Target Trackers. *arXiv Prepr. arXiv1503.01313*, 2015.

[103] Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Online object tracking with sparse prototypes. *IEEE Trans. Image Process.*, 22(1):314–325, 2013.

[104] Xian Yang, Quan Xiao, Shoujue Wang, and Peizhong Liu. Real-time Tracking via Deformable Structure Regression Learning. *Int. Conf. Pattern Recognit.*, 2014.

[105] Matej Kristan, Stanislav Kovačič, Aleš Leonardis, and Janez Perš. A two-stage dynamic model for visual tracking. *Trans. Syst. Man Cybern. Part B Cybern.*, 40(6):1505–1520, December 2010. ISSN 1083-4419. doi: 10.1109/TSMCB.2010.2041662.

[106] Luka Čehovin, Matej Kristan, and Aleš Leonardis. An adaptive coupled-layer visual model for robust visual tracking. In *IEEE Int. Conf. Comput. Vis.*, Barcelona, 2011.

[107] Luka Čehovin, Matej Kristan, and Aleš Leonardis. Robust Visual Tracking using an Adaptive Coupled-layer Visual Model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(4):941–953, April 2013. ISSN 1939-3539. doi: 10.1109/TPAMI.2012.145.

[108] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, January 1949. ISBN 0-252-72548-4.

[109] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognit. Lett.*, 27(8):861–874, June 2006. ISSN 01678655. doi: 10.1016/j.patrec.2005.10.010.

[110] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 49–56, San Francisco, CA, USA, June 2010. IEEE. ISBN 978-1-4244-6984-0. doi: 10.1109/CVPR.2010.5540231.

[111] Brendan J Frey and Delbert Dueck. Clustering by Passing Messages Between Data Points. *Science (80-.).*, 315:972–976, 2007.

[112] Janez Demšar. Statistical Comparisons of Classifiers over multiple datasets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[113] Nil Goyette, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, and Prakash Ishwar. Changedetection.net: A new change detection benchmark dataset. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, pages 1–8. IEEE, June 2012. ISBN 978-1-4673-1612-5. doi: 10.1109/CVPRW.2012.6238919.

[114] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation: theory algorithms and software*. Wiley-Interscience, 1 edition, June 2004. ISBN 047141655X.

[115] T W Anderson and D A Darling. Asymptotic Theory of Certain 'Goodness of Fit' Criteria Based on Stochastic Processes. *Ann. Math. Stat.*, 23(2):193–212, 1952.

[116] J Demšar. On the Appropriateness of Statistical Tests in Machine Learning. In *Work. Eval. Methods Mach. Learn. ICML*, 2008.

[117] A. Blake and A. Zisserman. *Visual reconstruction*. MIT Press, 1987. ISBN 9780262022712.

[118] R. Y. Rubinstein. Optimization of computer simulation models with rare events. *Eur. J. Oper. Res.*, 99(1): 89–112, May 1997. ISSN 0377-2217. doi: 10.1016/S0377-2217(96)00385-2.

[119] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Int. Jt. Conf. Artif. Intell.*, pages 674–679, April 1981.

[120] C. Harris and M. Stephens. A Combined Corner and Edge Detection. In *Alvey Vis. Conf.*, pages 151, 147, 1988.

[121] Matej Kristan, Aleš Leonardis, and Danijel Skočaj. Multivariate online kernel density estimation with Gaussian kernels. *Pattern Recognit.*, 44(10-11):2630–2642, October 2011. ISSN 0031-3203. doi: 10.1016/j.patcog.2011.03.019.

[122] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. Am. Soc. Mech. Eng. J. Basic Eng.*, 82:34–45, 1960.

[123] X. Rong Li, V. Jilkov P., and V. P Jilkov. Survey of Maneuvering Target Tracking: Dynamic Models. *IEEE Aerosp. Electron. Syst.*, 39(4):1333– 1364, October 2003. ISSN 0018-9251. doi: 10.1109/TAES.2003.1261132.

[124] Jingjing Xiao, R Stolkin, and A Leonardis. An Enhanced Adaptive Coupled-Layer LGTracker++. In *IEEE Int. Conf. Comput. Vis. Work.*, pages 137–144, December 2013. doi: 10.1109/ICCVW.2013.24.

[125] Michael Felsberg. Enhanced Distribution Field Tracking using Channel Representations. In *IEEE Int. Conf. Comput. Vis. Work.*, 2013.

[126] M E Maresca and A Petrosino. MATRIOSKA: A Multi-level Approach to Fast Tracking by Learning. In *Int. Conf. Image Anal. Process.*, pages 419–428, 2013.

[127] Weihua Chen, Lijun Cao, Junge Zhang, and Kaiqi Huang. An Adaptive Combination of Multiple Features for Robust Tracking in Real Scene. In *IEEE Int. Conf. Comput. Vis. Work.*, 2013.

[128] Laura Sevilla-Lara and Erik G Learned-Miller. Distribution fields for tracking. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1910–1917. IEEE, 2012. ISBN 978-1-4673-1226-4.

[129] A Salaheldin, Sara Maher, and Mohamed El Helw. Robust Real-Time Tracking with Diverse Ensembles and Random Projections. In *IEEE Int. Conf. Comput. Vis. Work.*, 2013.

[130] Martin Godec, Peter M Roth, and Horst Bischof. Hough-based Tracking of Non-rigid Objects. *Comput. Vis. Image Underst.*, 117(10):1245–1256, 2013.

[131] Dana H Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognit.*, 13(2):111–122, 1981.

[132] Julian Besag. On the Statistical Analysis of Dirty Pictures. *J. R. Stat. Soc. Ser. B*, 48(3):pp. 259–302, 1986. ISSN 00359246.

[133] Zhaowei Cai, Longyin Wen, Zhen Lei, N Vasconcelos, and S Z Li. Robust Deformable and Occluded Object Tracking With Dynamic Graph. *IEEE Trans. Image Process.*, 23(12):5497–5509, 2014.

[134] Tomas Vojir, Jana Noskova, and Jiri Matas. Robust scale-adaptive mean-shift for tracking. *Pattern Recognit. Lett.*, 49:250–258, 2014.

[135] S Duffner and C Garcia. Exploiting contextual motion cues for visual object tracking. In *Eur. Conf. Comput. Vis. Work.*, pages 1–12, 2014.

[136] M Danelljan, F S Khan, M Felsberg, and J Van de Weijer. Adaptive color attributes for real-time visual tracking. In *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2014.

[137] Kristoffer Öfjäll and Michael Felsberg. Weighted Update and Comparison for Channel{-}Based Distribution Field Tracking. In *Eur. Conf. Comput. Vis. Work.*, 2014.

[138] Mario Maresca and Alfredo Petrosino. Clustering Local Motion Estimates for Robust and Efficient Object Tracking. In *Eur. Conf. Comput. Vis. Work.*, 2014.

[139] H Nam, S Hong, and B Han. Online graph-based tracking. In *Eur. Conf. Comput. Vis.*, 2014.

[140] Georg Nebehay and Roman Pflugfelder. Consensus-based Matching and Tracking of Keypoints for Object Tracking. In *Winter Conf. Appl. Comput. Vis.* IEEE, 2014.

[141] K Briechle and U D Hanebeck. Template matching using fast normalized cross correlation. In *Aerospace/Defense Sensing, Simulation, Control. Int. Soc. Opt. Photonics*, pages 95–102, 2001.

[142] Katalin Tarnay, Gusztáv Adamis, and Tibor Dulai. *Advanced Communication Protocol Technologies: Solutions, Methods, and Applications*. IGI Global, 2011. ISBN 1609607325.

[143] Luka Čehovin. TraX: Visual Tracking eXchange Protocol, April 2014.