CENTER FOR
MACHINE PERCEPTION

CZECH TECHNICAL
UNIVERSITY IN PRAGUE

Short-Term Visual Object Tracking in Real-Time

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

Tomáš Vojíř

May, 2017

Thesis Advisor
Prof. Ing. Jiří Matas, Ph.D.

CTU–CMP–2017–04

Available at
ftp://cmp.felk.cvut.cz/pub/cmp/articles/vojir/Vojir-TR-2017-04.pdf

Research Reports of CMP, Czech Technical University in Prague, No. 4, 2017

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University in Prague
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

PhD THESIS

ISSN 1213-2365

Abstract

In the thesis, we propose two novel short-term object tracking methods, the Flock of Trackers (FoT) and the Scale-Adaptive Mean-Shift (ASMS), a framework for fusion of multiple trackers and detector and contributions to the problem of tracker evaluation within the Visual Object Tracking (VOT) initiative.

The Flock of Trackers partitions the object of interest to an equally sized parts. For each part, the FoT computes an optical flow correspondence and estimates its reliability. Reliable correspondences are used to robustly estimates a target pose using RANSAC technique, which allows for range of complex rigid transformation (e.g. affine transformation) of a target. The scale-adaptive mean-shift tracker is a gradient optimization method that iteratively moves a search window to the position which minimizes a distance of a appearance model extracted from the search window to the target model. The ASMS propose a theoretically justified modification of the mean-shift framework that addresses one of the drawbacks of the mean-shift trackers which is the fixed size search window, i.e. target scale. Moreover, the ASMS introduce a technique that incorporates a background information into the gradient optimization to reduce tracker failures in presence of background clutter.

To take advantage of strengths of the previous methods, we introduce a novel tracking framework HMMTxD that fuses multiple tracking methods together with a proposed feature-based online detector. The framework utilizes a hidden Markov model (HMM) to learn online how well each tracking method performs using sparsely "annotated" data provided by a detector, which are assumed to be correct, and confidence provided by the trackers. The HMM estimates the probability that a tracker is correct in the current frame given the previously learned HMM model and the current tracker confidence. This tracker fusion alleviates the drawbacks of the individual tracking methods since the HMMTxD learns which trackers are performing well and switch off the rest.

All of the proposed trackers were extensively evaluated on several benchmarks and publicly available tracking sequences and achieve excellent results in various evaluation criteria. The FoT achieved state-of-the-art performance in the VOT2013 benchmark, finishing second. Today, the FoT is used as a building block in complex applications such as multi-object tracking frameworks. The ASMS achieved state-of-the-art results in the VOT2015 benchmark and was chosen as the best performing method in terms of a trade-off between performance and running time. The HMMTxD demonstrated state-of-the-art performance in multiple benchmarks (VOT2014, VOT2015 and OTB).

The thesis also contributes, and provides an overview, to the Visual Object Tracking (VOT) evaluation methodology. This methodology provides a means for unbiased comparison of different tracking methods across publication, which is crucial for advancement of the state-of-the-art over a longer timespan and also provides a tools for deeper performance analysis of tracking methods. Furthermore, a annual workshops are organized on major computer vision conferences, where the authors are encouraged to submit their novel methods to compete against each other and where the advances in the visual object tracking are discussed.

Acknowledgments

I would like to thank my advisor Prof. Jiří Matas for guiding me throughout my Ph.D., teaching me how to be a good researcher and for the immense patience he showed during the periods of academic paper writing.

I am grateful to my co-authors Jana Nosková, Matej Kristan, Luka Čehovin Zajc and Alan Lukežič for a wonderful collaboration on many interesting projects and for making the research truly enjoyable.

I also thank my colleagues Jan Šochman, Michal Bušta, Jan Čech, Ondřej Chum, Štěpán Obdržálek, Michal Perďoch, Andrej Mikulík, Lukáš Neumann, James Pritts, Milan Šulc, Filip Radenovic, Tomáš Hodaň and many others for countless fruitful discussions on various topics and for helping me to learn from my mistakes. They all together created a great working environment in Center for Machine Perception and transformed the endless hours I spent in the office from bearable to almost glamorous.

Finally, I would like to thank my soon-to-be wife Petra and my parents for unconditionally supporting me in both professional and personal lives.

I gratefully acknowledge the support by the Toyota Motor Europe, the Czech Science Foundation Project GACR P103/12/G084, the Technology Agency of the Czech Republic project TE01020415 (V3C – Visual Computing Competence Center) and by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

Contents

Chapter 1

Introduction

The general visual object tracking is one of the most challenging problem in the computer vision field. One of the reason is the large space of possible application and theoretical problems which are considered to belong to tracking. Since it is currently not feasible to address the problem in full generality, it is naturally divided into more narrow subproblems. These subproblems differ greatly in assumptions they impose on the task or data and in requirements on output. Examples of factors of tracking specialization are e.g.the number of cameras, the number of objects of interest, causality (e.g. video analysis vs. causal processing), data domain type (e.g. medical, depth data, infrared imaging or classical RGB camera images), assumed frame-to-frame transformation or static vs. moving camera.

The visual object tracking problem addressed in this thesis is restricted to single camera, single object and causal processing. The objective of such tracking task is to estimate trajectory of the object of interest – target – and its state in each frame in the video. The target position is usually denoted more generally as a target pose, which may represent additional geometrical properties such as target size or rotation. The target is defined by an external source in the first frame of the tracking sequence, e.g. by rectangular area drawn by a user. The target pose representations may vary depending on the specific problem and tracking methods. The pose can be for example position of the target center, a bounding box which also incorporates the target size or a set of pixels belonging to the target.

Despite addressing only a ”small part” of the general visual object tracking problem, the topic of the thesis is challenging due to its unconstrained character in terms of objects types (e.g. from rigid to highly articulated objects), object motions (e.g. planar or 3D motion), object appearance variations (e.g. changes due to lighting conditions or object deformations) and unconstrained environment conditions (e.g. high illumination changes, occlusions, video capture noise, different weather conditions or unconstrained camera motion and viewpoint). Visual examples are shown in Fig. 1.1.

Evaluation and comparison of different tracking approaches is another challenging aspect in visual object tracking. In recent years, many tracking methods were published and the necessity to objectively compare these methods with each other becomes essential problem. Commonly, results of published methods cannot be directly compared for two main reasons, i.e. each method may have used different set of testing video sequences and different evaluation metric. A unified framework for evaluation and

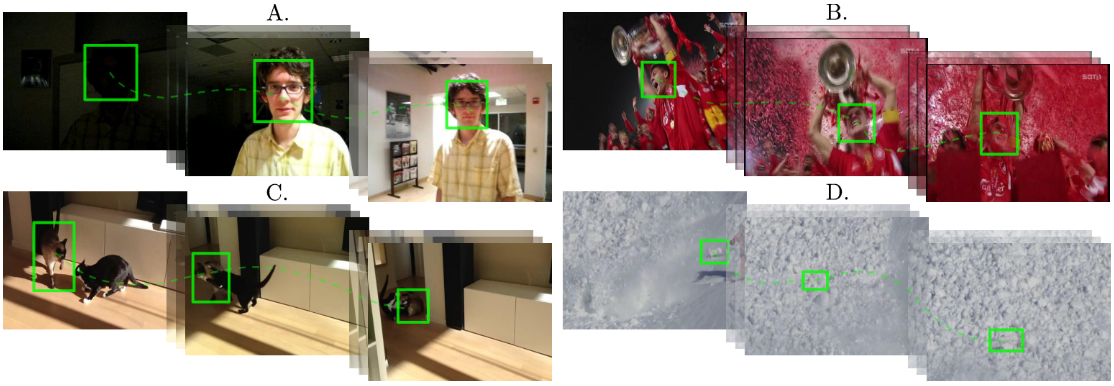


Figure 1.1: Examples of various visual challenges in tracking scenarios. From left to right and top to bottom: A) extreme global and local illumination changes, B) view point changes and occlusion, C) highly deformable and articulated target and D) background clutter and texture-less object.

comparison of tracking methods is imperative for continuous advancement of state-of-the-art.

Visual tracking is utilized in a vast amount of direct applications or as a building block in more complex applications of computer vision. Well known applications of tracking are for example a face tracking in digital cameras, body skeleton tracking in Microsoft Kinect[1] device for video game control or CCTV surveillance application such as left luggage or pedestrian tracking in the airport[2]. Many other applications use the visual tracking as a module and build more complex computer vision algorithms on top of the outputs of tracking methods. Most prominent examples of these applications are car assistant systems[3] [4] used for emergency breaking and adaptive cruise control, eye tracking for driver attention monitoring[5], non-intrusive visual sport analysis software[6] which provides rich information about players performance (such as distance run, acceleration or team formations) or systems for augmented reality [KM09]. Figure 1.2 shows several of these applications.

The topic of visual tracking is far from being solved and there is a growing demand for faster and more robust methods, that would open a path to new applications and, ultimately, toward a single tracking method that would address all mentioned difficulties and possible scenarios in the causal single object visual tracking problem.

1.1 The Scope of the Thesis – Tracking Problem

As discussed in the previous section, the general tracking problem is too broad and diverse to be addressed as whole and this thesis focuses only on a specific area which will be described more precisely in the following paragraphs.

[1] http://www.xbox.com/en-US/xbox-one/accessories/kinect

[2] http://www.fujitsu.com/uk/solutions/industry/transport/aviation/

[3] http://www.mobileye.com/

[4] https://www.google.com/selfdrivingcar/

[5] http://www.lexus.com/models/LS/safety

[6] http://chyronhego.com/sports-data/tracab

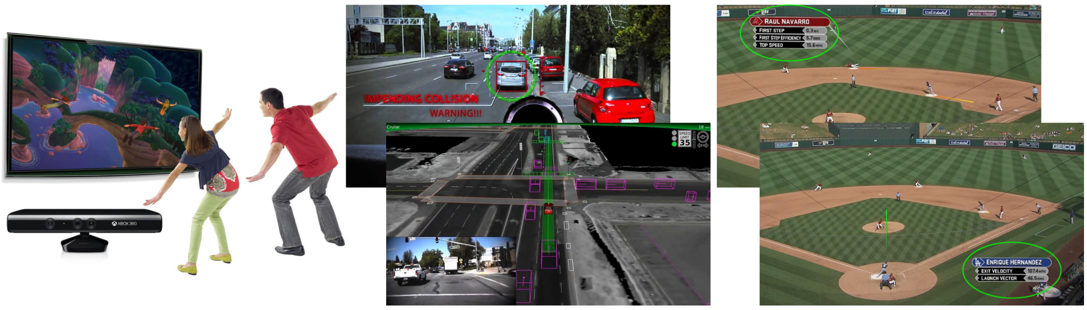


Figure 1.2: Examples of the use of tracking algorithms in real-world applications. From left to right: Kinect gaming sensor for body tracking, tracking of cars for collision detection and autonomous driving and sport analysis.

We categorize tracking problems by four major factors, i.e. single camera vs multiple camera setting, single target vs. multi-target, causal vs. non-causal and short-term vs. long-term tracking. This thesis is focusing on single camera, causal and single target tracking for mostly short-term scenarios. In rest of the thesis, by tracking problem we refer to this definition unless stated otherwise. In next paragraphs, the individual terms of the tracking problem are explicitly defined.

By single camera we means that the tracking algorithm operates on a sequence of single 2D images, i.e. most common case when using a standard camera device. On the other hand, multi-camera system provides multiple 2D images at the same time. Usually, these images are taken from different overlapping viewpoints and are time synchronized, e.g. multiple cameras in the corners of a hockey stadium capturing a game.

The single target tracking considers only one object of interest in the scene and the rest is treated as a background even when similar object may be present. Multi-target tracking objective is to report all instances of object of interests in the scene that are usually from the same category (e.g. group of pedestrians or cars on a road) but not limited to (e.g. tracking all moving object w.r.t. static background).

Causal tracking requires that the output of the method is produced in each frame using only information from previous and current frames. Most of the research in object tracking falls into the category of casual trackers. On the other hand, non-causal tracker may process information from a whole video sequence at the same time and is not required to output target poses sequentially. The non-causal tracking task can be viewed as a video analysis techniques, where the delay in output is not an essential parameter.

The short-term and long-term tracking mainly differ in the expected length of the video sequence. In the short-term tracking task, the main optimized criterion is a precise estimation of the object of interest state until for as long as possible until the tracking method fails after which a recovery is not necessary required. Long-term tracking is more focused on the online learning of an object appearance and applying this knowledge to recover in case of tracker failure or object disappearance from the field of view due to other effects such as full occlusion or out-of-image disappearance. Short-term tracking is usually evaluated up to 1000 frames per sequence while long-term tracking should work for arbitrary long sequences (benchmark sequence lengths are typically in a range from 1000 to 10000 frames).

Given the specification of tracking problem (i.e. single camera and target, casual and short-term) we can define more formally what the inputs and expected outputs are. There are several well-established definitions used in the literature from which two – one standardly used and one more general that represents an ongoing trend of how the tracking task is interpreted – are presented.

The first definition 1, adapted for example in [MC11], is applicable for ”correspondence” based trackers. The unit for which the correspondence between frames is established can vary a lot in size and meaning from small size pixel keypoints (e.g. optical-flow, sparse or dense, methods) to very large size correspondence of the whole (parts of) object (e.g. part based, detection based or correlation trackers), see Section 2.2 for particular tracking methods examples.

Definition 1 *Given an initial estimate of X, locate X (or transformation of X) in all images in a sequence, where X may mean*

1. *An "interest point" and its neighbourhood*
2. *A (rectangular) region*
3. *An "object"*

This definition is sufficient when we consider mostly rigid object and planar transformation (e.g. translation, scale and rotation in the image plane), because it makes an assumption that an object is well represented by a rectangle and the rectangle transformation between consecutive frames correlates to the transformation that the object undergoes. This assumption is however not true when we consider more complex objects (e.g. highly articulated or deformable) or object 3D motion (e.g. out-of-plane rotation where the object undergoes complex 3D motion that does not manifest in change of 2D bounding box).

The definition 2 generalize the previous definition 1 in such way that it recognizes that an object has its state and that the state is, at some point, used to estimate the output object pose (e.g. region); however, there is not necessarily a correlation between object poses in consecutive frames. For example, in segmentation based methods, the object state is represented by a set of pixels that belongs to the object and output pose can be a rectangle that encloses them. However, the corresponding sets of pixels between frames may differ in size, shape properties or number of connected components and there is no obvious mapping between them. The only requirement being that the pixels in the set belongs to the target. This definition also separates target pose from target state more clearly. The pose is viewed as an extrinsic geometrical property related to the observer, e.g. bounding box, contour, frame-to-frame transformation and the state represent the intrinsic parameters of the target, e.g. pixel that belong to the object, articulated parts of the target or more high level information such as closed/opened eyes during face tracking.

Definition 2 *Given an initial estimate of the pose (and state of X) : In all images in a sequence, (in a causal manner)*

1. *estimate the pose of X (e.g. any geometric parameter – position, scale, ..., bounding box, contour, etc...)*

2. (optional) *estimate the state of X (e.g. pixel-wise segmentation, shape, articulation, etc...)*

The definition 2 emerged when first tracking by segmentation methods appeared; however, it describe very well the recent state of tracking methods and evaluation frameworks. In modern tracking benchmarks (e.g. VOT [KML$^+$16]), the ground truth annotation is understood as a rough approximation of the target segmentation opposes to frame-to-frame target motion (when taking consecutive ground truth annotations) and it also applies for recent tracking methods, which does not estimate explicit transformation of target. This definition easily captures the cases of articulated and deformable object as well as non-rigid and 3D transformations.

The tracking methods proposed in this thesis (Chapter 4, 5) follows both definitions. The FoT tracker is based on sparse point correspondences and assumes rigid object which frame-to-frame motion can be described by one global transformation (e.g. translation, scale, and rotation) of object bounding box. The ASMS tracker is based on an iterative procedure that optimizes a similarity between two probability distributions functions, i.e. seeking a region with the highest density of this similarity function, which can be seen as a segmentation without explicitly defined the segmented pixels. Finally, the HMMTxD tracker combines several out-of-box tracking methods, therefore, it fits to both definition depending on the tracking methods that participate in the HMMTxD tracker.

1.2 Thesis Overview

This thesis presents several contributions to the causal single object short-term tracking problem. First, we introduce a novel visual object tracking evaluation methodology (VOT) which goal is to create a well-defined protocol, dataset and measurements for evaluation of tracking algorithms and a principal way how to compare tracking algorithms against each other. This is necessary for systematic advance in the state-of-the-art as well as for development of new tracking algorithms.

The VOT started in 2013 when there was no standard evaluation benchmark for a single-target tracking and the first results were presented on VOT workshop which was organized in conjunction with ICCV 2013 conference. Since then, we have been working on improving all parts of the methodology (i.e. dataset, measurements, evaluation toolkit code base) and we have been presenting the progress annually on the VOT workshops at major conferences. The main contributions to the VOT were in development of methodology for selecting a dataset and in the evaluation methodology (i.e. single evaluation measure for tracker ranking and statistical validation of results). Chapter 3 presents these contributions along with the latest development of the VOT methodology and results from the last VOT 2015 workshop, where we compared 62 tracking methods on a dataset of 60 sequences.

In Chapter 4, we present two short-term tracking algorithms that are simple yet robust and achieve a state-of-the-art performance while running at a speed more than 100 frames per second on average. First approach presented in Section 4.1, called Flock of Trackers (FoT), is based on a robust combination of sparse optical flow correspondences. The object of interest is covered uniformly by local trackers (i.e. regions for

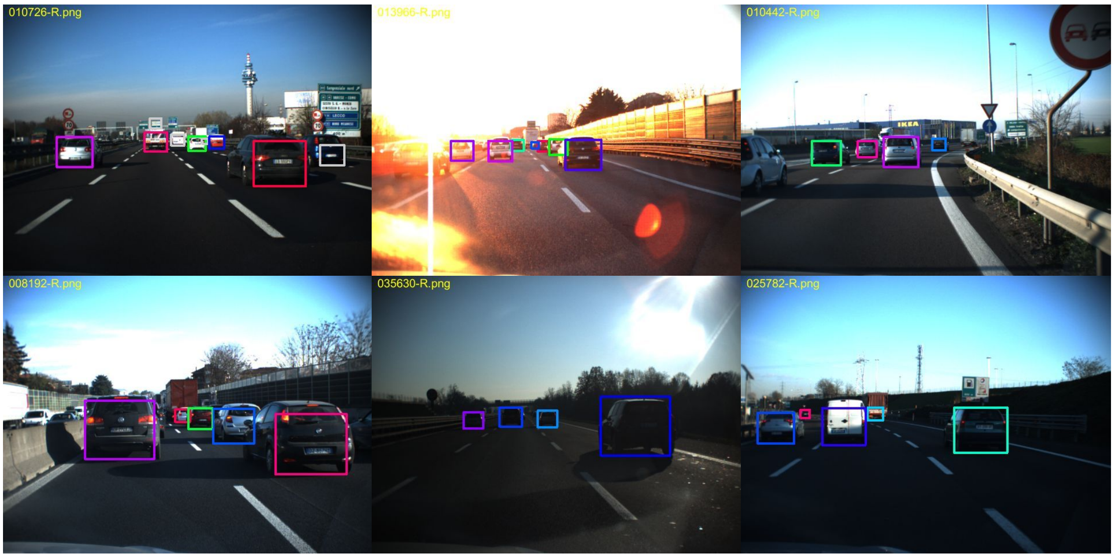


Figure 1.3: FoT tracker outputs of the Toyota car tracking framework [CVT$^+$12]. Diverse visual conditions in highway scenarios are shown: left - cluttered scene with partial occlusions, center - difficult lighting condition and right - highway turns and major road shadows. The whole video sequence, dataset and evaluation protocol are described at `http://cmp.felk.cvut.cz/data/motorway/`.

which the correspondences to a consecutive image are estimated) and using multiple quality predictors a reliable set of local trackers is selected. This reliable set is used to estimate a global motion of the object, which can be modeled up to projective transformation. This method achieved state-of-the-art results in the VOT2013 benchmark and even today it provides a robust tracking method for rigid objects with the possibility of more complex motion model (very few methods in visual object tracking are able to estimate more than translation and scale transformation) at speed $100-300$ fps. The FoT tracker is also used in two real-world applications: i) in the project with Toyota [CVT$^+$12] (see Fig. 1.3 for visual results), where the algorithm currently runs in the testing cars, to track cars from a single front-mounted camera and ii) in the *TextSpotter* software[7] for text localization and recognition in videos (see Fig. 1.4 for visual results), where the FoT provides temporal smoothing for the text localization between frames, i.e. if some text is detected a new instance of FoT is run to track that location and helps to locate the same text in consecutive frames where there is no detection. The Baseline-TextSpotter algorithm with the tracking module placed first[8] in the ICDAR 2015 robust reading competition[KGBN$^+$15] for the end-to-end system.

The second short-term tracking method, which builds upon a *mean-shift tracking* approach [CRM00], is introduced in Section 4.2. The mean-shift tracking minimizes a distance of a target model to the currently observed model by shifting its position using a gradient descent technique. The proposed scale-adaptive mean-shift tracker (ASMS) introduces a theoretically justified scale estimation to the mean-shift tracking. The novel scale estimation allows the mean-shift tracker to compete with other state-of-the-art trackers in challenging tracking video sequences, where scale estima-

[7] `http://textspotter.org/`

[8] `http://rrc.cvc.uab.es/?ch=3&com=evaluation`

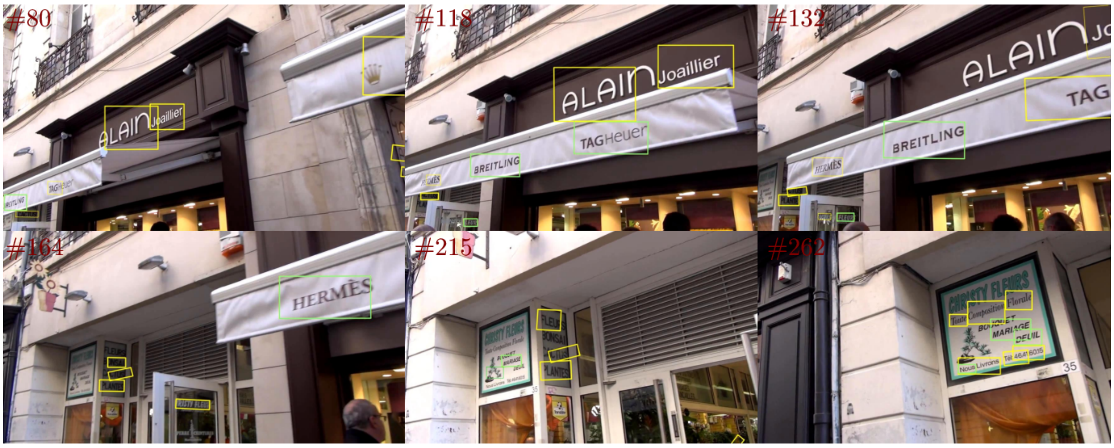


Figure 1.4: Output of the TextSpotter software for text localization and recognition. First, text is detected at some point in the video sequence (green regions) and then the region is tracked using the FoT tracker (yellow regions). The text detections are sparse and the FoT provides the text positions through frames where the detections are missing. Images courtesy of Michal Busta.

tion is crucial for achieving good performance. Moreover, a novel general approach for mean-shift trackers to exploit context information (i.e. background around the object of interest) is proposed. It computes the background model and uses it to identify discriminative parts of the target model, which become more prominent in the gradient optimization. Two major issues with scale estimation in the mean-shift framework are discussed (i.e. object symmetry in a feature space and a background clutter) and a solution is proposed in the form of a scale regularization, forward-backward validation and, the previously mentioned context exploitation. The ASMS tracker achieved state-of-the-art results and is the best performing tracker in terms of trade-off between the performance and the speed on the VOT 2015 benchmark with processing speed around $80 - 150$ fps in average.

In Chapter 5, we introduce a novel tracking and detection method (HMMTxD) which fuses multiple short-term tracking methods (Tx) utilizing the hidden Markov model (HMM) framework and using a feature-based detector (D) as a glue to connect the learning process of the HMM with the short-term trackers outputs. This method allows to utilize tracking approaches with complementary designs and alleviate their failure modes, e.g. fuse a FoT tracker that works for better for rigid object and ASMS mean-shift tracker that cope well with deformable and articulated object. The feature-based detector marks key frames, with positive object detections, which are used as annotated data during the learning of the parameters of the HMM. The state of the Markov model represents the ”correct” short-term trackers combination $\{0, 1\}^n$, where n is a number of trackers and 0 (1) denotes tracker incorrectness (correctness) in the state. Therefore by inferring the most probable state the output of the method is defined by the tracker’s combination in that particular state. The HMM models two major characteristics of trackers: i) the temporal performance, i.e. whether this tracker is working correctly for this particular object and a scene and ii) correlation of current tracker observables (e.g. tracker confidence measure or model similarity) with the performance. Both of these parts of the HMM (transition probabilities and observable probabilities) are learned online by the proposed modified Baum-Welch algorithm. The modularity of the proposed method allows replacing individual parts in an application-driven fashion, e.g. to use an object specific detector (a pedestrian detector) instead of the generic feature-based detector or a different combination of trackers that performs better for a given task. The proposed method achieves a state-of-the-art performance on VOT benchmarks, OTB [WLY13] benchmark and TV77 collection of sequences. The processing speed of the method is mainly limited by the feature-based detector and is between $5 - 15$ frames per second on average.

1.3 Contributions

The contributions lie in two areas. We present three novel tracking methods, and second, we present contributions to the evaluation methodology of the tracking algorithms. The contributions are listed below,

- Section 4.2 present a novel formulation of the mean-shift tracker [CRM00] which introduce theoretically justified scale estimation inside the mean-shift framework. Furthermore, we introduce a background information to the mean-shift procedure which allows for robust tracking in cluttered scenes. Lastly, a regularization is proposed for the mean-shift gradient descent optimization to make the motion and scale estimation more robust and reliable.

  The work was published in [VNM14, VNM14] and received the *best paper award*. The source code is publicly available at `https://github.com/vojirt/asms`.

- Section 4.1 propose a novel short-term tracking method. This method is based on work of Kalal et al. [KMM10b]. The tracking approach uses local features placed on the object of interest and their correspondences to the next frame to estimate an object motion. There are three major contributions to the design. First, the placement of the local features is discussed and an improved solution is proposed. Then several techniques for assessing of reliability of the individual local features are proposed and two approaches for efficient fusion of these indicators are presented. Finally, a RANSAC technique is used for the motion estimation, which enables for more complex motion models to be used.

  These contributions were published in [VM11, VM14] and indirectly in [CVT$^+$12]. The algorithm was licensed to Samsung and Toyota corporations.

- Chapter 5 propose a novel method which allows fusing information from multiple ”black box” tracking approaches and online learned object detector, e.g. fusing previously proposed FoT and ASMS with complementary design and therefore in different failure modes. Trackers performance statistics are learned online utilizing a Hidden Markov model. The method effectively enables to ”switch off” trackers that are not suitable for a particular sequence in an online manner. We introduce an algorithm for learning the HMM model from online partially annotated sequence of observed data. For the data annotation, we propose an online learned feature-based detector set to the operating point of high precision that annotates the keyframes where the tracker's performance is evaluated. This work was published in [VNM16].

- Chapter 3 presents the Visual Object Tracking (VOT) evaluation methodology. This methodology uses two standard performance measures[ČKL14] and novel expected average overlap measure(EAO), a process for automatic selection of a representative set of sequences for trackers evaluation and toolkit for automatic evaluation of tracking algorithms and generating performance statistics. The VOT methodology is used in the workshops, organized in conjunctions with major conferences, and enables the competition between tracking algorithms which allows monitoring the advance in the state-of-the-art through the years. The author of this thesis is a major contributor to the dataset selection process, development of the evaluation toolkit and methodology (single performance measure for tracker ranking, i.e. EAO) and organization of the workshops.

  The paper related to this topics are [KPL$^+$13, KPL$^+$14, FBH$^+$15, KML$^+$15, KML$^+$16]. Everything related to the VOT is open source and can be accessed from `http://www.votchallenge.net/`.

Chapter 2

Related Work

In this chapter, related work for a single target tracking evaluation and single target tracking is reviewed. In the first part, performance measures, visualizations approaches and tracking benchmarks are presented. They are crucial for assessing the quality of tracking methods, comparing different algorithms across publications and for advancing the state-of-the-art by allowing to easily identify methods weaknesses.

In the second part of this chapter, related tracking methods are categorized by their most common characteristics and briefly described. The section does not exhaustively list all tracking methods, which is not feasible (the number of papers published on major conferences between years $2013 - 2016$ is around 1500), but rather selects the most influential methods and the recent state-of-the-art.

2.1 Tracking Performance Evaluation

Tracking performance evaluation consists of three key parts: performance measure, visualization of the measures and evaluation dataset. There is number of performance measures proposed in literature, where a measure criterion was often proposed to fit a tracking task narrative. For example, when the tracking task is to estimate a trajectory of particular cell in microscope images the size of the cell is not an important factor but the precision of the trajectory (i.e. motion of the cell center) is or when the task is to remove a moving object from a video sequence a more precise measure that incorporate object position, size and rotation is desirable for evaluation of tracking methods for this task.

The other important part of the evaluation is presentation and visualization of results. Currently, the trend is to use large datasets, therefore, the techniques to clearly articulate the performance results are crucial, since is not feasible to include a per-sequence performance in form of a table. Several visualization techniques were design to address this need. The important aspect that the visualization should have is the ability to either compare methods with each other or to provide a insides to how the method perform for certain scenarios (e.g. visual attribute like illumination change) in a clear compact way.

The third key part is the dataset on which the trackers are evaluated. The dataset should be ”well-rounded” and ”comprehensive” in terms of visual attributes. Because of the enormous space of possible sequences and visual phenomenons there is no proper definition or guide lines how to select sequences for the dataset. The authors usually choose the sequences arbitrary or to emphasis particular visual attribute. The problem of dataset construction is not unique for the tracking but rather it is a general problem in computer vision.

For the purpose of the comparison of tracking methods across publication, three popular (standard) single target tracking benchmarks were proposed. Each benchmark provides datasets, performance measure, results visualization and toolkit for easy and automatic evaluation of tracking methods. Short overview of these benchmarks is provided in Section 2.1.3 where their strengths and weaknesses are also discussed.

The scope of this thesis is limited to a single target tracking, which may differ significantly in evaluation methodology w.r.t. multi-target tracking, so for details about multi-target tracking evaluation readers are referred to the MOTChallenge[1] and to the paper of Bernardin et al. [BS08].

2.1.1 Performance Measures

Many performance measures have been proposed for tracker evaluation. The range of different measures varies from a simple center of regions difference to more sophisticated measures that combine multiple aspects of reported regions. This section provides an overview of the most used measures and their advantages and shortcomings.

The simplest performance measure is the localization error [RLLY08]. It is computed for frame t as an error between centroids of the ground truth position $\mathbf{c}_t^{gt} = [x_t^{gt}, y_t^{gt}]$ and the estimated position $\hat{\mathbf{c}}_t = [\hat{x}_t, \hat{y}_t]$ using Euclidean l_2 norm $\delta_t = ||\mathbf{c}_t^{gt} - \hat{\mathbf{c}}_t||$. This measure is usually summarized as an average error (Eg. 2.1) or as a root mean squared error (Eg. 2.2) for a sequence of length N.

$$\text{AE} = \frac{1}{N}\sum_{t=1}^{N}\delta_t \tag{2.1}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\delta_t^2} \tag{2.2}$$

The localization error was also adapted for the tracking length [KL09] measure. The tracking length is defined as the frame number after which the tracking method has the center localization error higher than a predefined threshold (in [KL09] set to 30 pixels). Kristan et al. [KPPK09, KKLP10] take it one step further and introduce a failure rate measure. The failure rate measures the number of times the tracker ”loss” a target. In the [KPPK09, KKLP10], a ”lost” of the target is indicated by the center localization error being above a threshold. The same idea can be used with other types of measures for which a decision whether the tracker is correct or not can be made for each frame. However, this measure requires an outside intervention to reinitialize the tracker when the target is lost. Wu et al. [WLY13] propose to measure the percentage of frames in which the center distance is less than a threshold and plot this in the graph for different thresholds. The measures based on the average center localization error have two major

[1] https://motchallenge.net/

flaws: (i) The error value is not bounded, therefore, for two trackers that failed the error can become arbitrarily high and the penalization uneven. Even though a normalization to image size can be applied, this problem perseveres. (ii) This measure takes into account only translation disregarding other object transformations such as rotation or scale.

The next set of measures is based on a region overlap [LSS11] criterion inspired by the Pascal detection challenge [EEVG+14]. For two rectangles, ground truth R^{gt} and estimated $\hat{R}$, the region overlap is defined by the Eq. 2.3.

$$o = \frac{R^{gt} \cap \hat{R}}{R^{gt} \cup \hat{R}} \tag{2.3}$$

This formula is general and can be easily applied to polygons or pixel-wise segmentation. The basic use is to evaluate an average frame overlap over a sequence, which, to some degree, describes the quality of a tracker. In [WLY13], the overlap measure is used to produced an ROC-like plot of thresholded overlaps and report the area under this curve. This presents an interesting visualization of trackers accuracy with a different threshold on the overlap measure. However, as it was shown later by Čehovin et al. [ČKL14], the area under the ROC-curve is equivalent to the average region overlap computed from all frames. In the extensive analysis of performance measures, Čehovin et al. [ČLK15], argue that the region overlap has superior quality to the center error. Similarly to the tracking length, a recall measure (the Pascal region overlap criterion [EEVG+14]) appeared as a prominent alternative for evaluation of long-term trackers, which often possesses a re-detection capability. The recall on sequences is the percentage of frames where a tracker has overlap higher than a given threshold. A complementary measure to the recall is precision. It is mostly used in tracking tasks where the answer ”target is not visible” is important, e.g. detection-based methods [KMM12, LHMB16]. The precision is the percentage of frames where the tracker provides output and is correct (correctness defined the same way as in recall measure). More precisely, the recall and precision are, using classification terminology, defined as

$$\text{recall} = \frac{tp}{tp + fn} \tag{2.4}$$

$$\text{precision} = \frac{tp}{tp + fp}, \tag{2.5}$$

where *tp*, *fn* and *fp* are number of true positive, false negative and false positive detections respectively. The F-score [KK11, SCC+13b, KMM12] was used to combine the recall with precision. The F-score, a harmonic mean of recall and precision, is defined as

$$\text{F-score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{2.6}$$

The measure based on a thresholded overlap depends heavily on the selected threshold, which is usually set to 0.5. In many cases, this is a very strict threshold for tracking task and may not represent accurately the failure cases of a tracker (see examples in Figure 2.1). However, compared to the localization error, it is bounded (failed trackers have the same overlap) and it takes into account target transformations, such as scale and rotation.

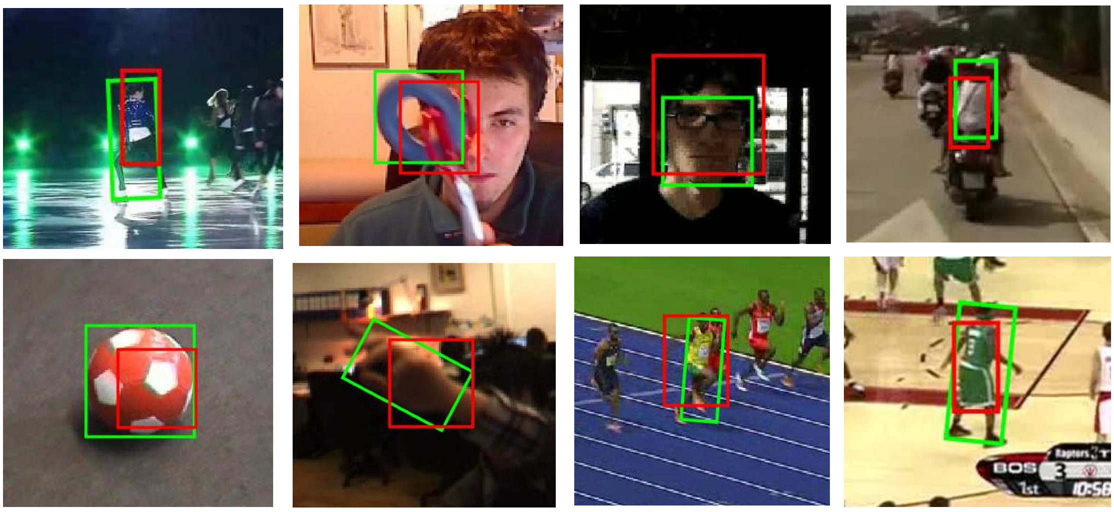


Figure 2.1: Examples of bounding boxes (red) at 0.5 overlap with the ground truth (green). Notice that the rectangles still fit the objects quite well. Image courtesy of Kristan et al. [KML^{+}16].

There are also more sophisticated measures, such as Combined Tracking Performance Score (CoTPS) [NC13, CCCR12], which combines several measures. The CoTPS incorporates a tracking accuracy, computed as an thresholded overlap $o(\tau)$ (overlap computed according to Eq. 2.3) and integrated over τ and normalized to range $[0, 1]$, and tracking failures λ_0 defined as the number of frames where the tracker did not produce bounding box and target was present or when bounding box was produced but the target was not present in the frame. Linear combination is used to produce a single score:

$$\text{CoTPS} = \beta \frac{1}{C} \int_{\tau} o(\tau) + (1 - \beta)\lambda_0 \tag{2.7}$$

The merit of the combined measures is the single score representation of the tracker quality that encompasses multiple characteristics of the tracking performance (e.g. accuracy and failure rate). A downside of the CoTPS is the limited insight into trackers performance on a finer level which reduces their interpretability, e.g. answering a question such as “which tracker is more accurate in the case of illumination changes?”.

2.1.2 Performance Visualization

The measures described so far are usually reported as a single average number per dataset or per sequence in a table. Recently, a novel trend appeared that compares the tracking performance visually by so called summarization plots. There are several types:

- Aggregate all sequences (frames) together and plot the measure-threshold dependence, i.e. a performance measure as a function of a threshold, see Fig. 2.2(A). Examples of these types are the precision plots [YJS06, BYB11, WLY13], which measure the object location accuracy in terms of a center error, or the success plot [WLY13] where the region overlap is used instead.

- Plot dependency between two performance measures (which are averages per sequence or per dataset), see Fig. 2.2(B). A tracker performance is then represented as a single point in a 2D graph, e.g. A-R plot presented in Čehovin et al. [ČKL14, ČLK15] where the trackers are compared in terms of accuracy and robustness (using a zero threshold on an overlap and a tracker reinitialization in the case of a zero overlap). The drawback of this plots is that they typically become cluttered and their clarity degrades when multiple trackers are in the same plot.

- Survival curve (proposed by Smeulders et al. [SCC$^+$13b]), which shows sorted per-sequence performance of a tracker. Trackers are then compared on the entire dataset visually by their survival curves. These graphs show a clear comparison of trackers on entire datasets; however, they do not allow per-sequence comparison since the sequence ordering differs for each tracker. Similar approach was used by Vojir et al. [VNM14] when comparing one particular method against many (e.g. proposed method compared with state-of-the-art), where the sequences are ordered by the performance of the selected method, see Fig. 2.2(D). This plot preserves the clean performance decreasing curve only for the selected method, however, it allows direct per-sequence performance comparison with the other methods.

- Challenge plots used where single measure is computed over whole dataset and tracking methods are ordered by this one measure (e.g. EAO plot in visual object tracking challenge (VOT) [KML$^+$15, KML$^+$16]), see Fig. 2.2(E).

- Performance with respect to different visual attributes, e.g. radar plot in [NC13, LVC$^+$17], see Fig. 2.2(C).

2.1.3 Single Target Tracking Benchmarks

There are three popular single target tracking benchmarks [WLY13, SCC$^+$13b, KML$^+$16], which provide a dataset, evaluation methodology and results for state-of-the-art methods so a new tracking method can be easily evaluated and compared to them.

The "Amsterdam Library of Ordinary Videos" (ALOV) benchmark proposed by Smeulders et al. [SCC$^+$13b] contain 315 video sequences annotated by axis-align rectangles every 5th frame and linearly interpolated between the key frames. The dataset is organized into 14 different categories based on the most dominant attribute (e.g. the first category contains sequences with an illumination change as a dominant characteristic). The tracker evaluation uses two performance measures, F-measure with overlap threshold set to 0.5 and the centroid normalized distance. Tracker results are visualized by the survival curve for each measure. The final ranking is based on the average F-measure over all sequences. One of the best performing trackers are detection-based methods TLD [KMM12] and STRUCK [HST11].

The OTB benchmark [WLY13] uses a dataset of 50 video sequences and evaluates 29 trackers (the revised version [WLY15] contains 100 sequences and 31 trackers). The dataset has a per-frame ground truth annotation by axis-aligned rectangles and 11 attributes per sequence. There are three types of experiments: (i) Baseline one-pass

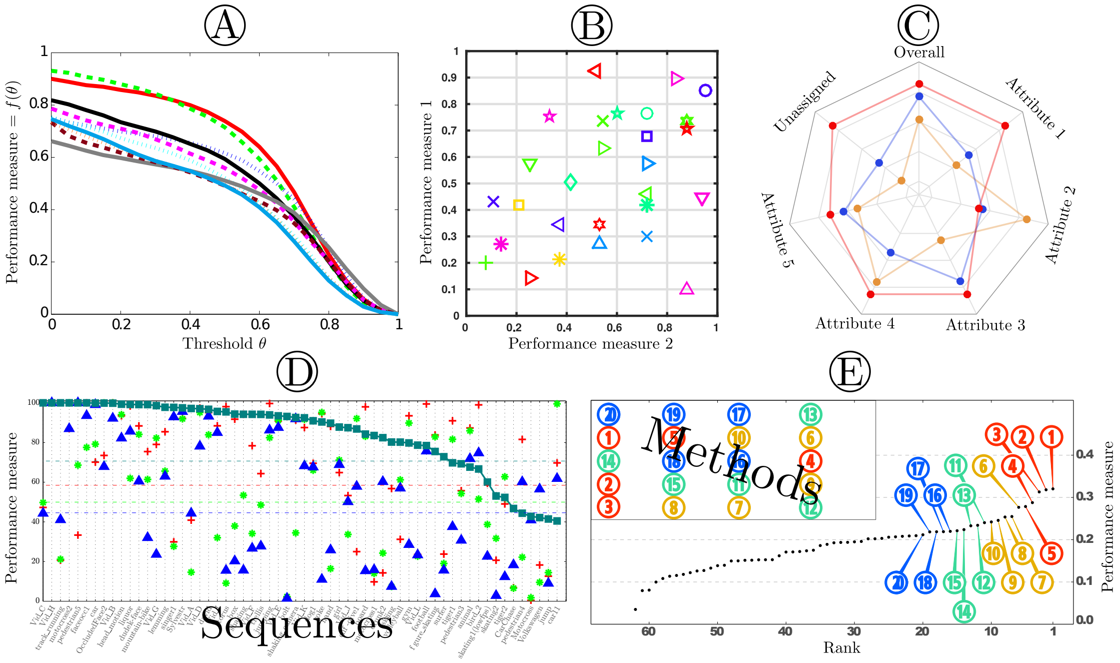


Figure 2.2: Different types of tracker performance visualization. A) performance as a function of a acceptance threshold, B) performance w.r.t. two complementary performance measures, C) performance for different visual attributes, D) survival curve with fixed sequence ordering based on a highlighted method and E) ranking plot w.r.t. a single performance measure. For further details, see text in Section 2.1.2.

experiment runs a tracker initialized from the first frame to the end of a sequence. (ii) Temporal perturbation experiment starts the tracker from frames uniformly sampled from the sequence and averages the performance from these runs. (iii) Spatial perturbation experiment starts the tracking from the first frame but augmented ground truth bounding box (perturbation in center and scale) and averages results from each run. The results of each experiment are visualized by a precision and success plots. In the end, trackers are ranked by the area under curve (AUC) value. Similarly to the previous benchmark, the best performing trackers are TLD [KMM12], STRUCK [HST11] and SCM [ZLY12].

There are two major weaknesses of the methodology shared by ALOV and OTB benchmarks. Firstly, the evaluation initializes trackers at the beginning of a sequence and lets them run until the end after which the performance measure is computed. This approach does not fully utilize the sequences since, after the tracker fails, the rest of the sequence is rendered useless. The exception is the second experiment of the OTB, which is, however, designed to evaluate a tracker sensitivity to the initialization frame rather than exploit the whole sequence. Secondly, the benchmarks use unsuitable long-term measures (recall, precision) on (mostly) short-term sequences, a brittle performance measures like center-based or overlap with a questionable threshold 0.5 (see Figure 2.1). Moreover, these benchmarks do not consider a tracker equivalence problem that is inherently present because of errors in the ground truth annotations. Therefore, ranking trackers based on one performance measure, e.g. using only the average overlap (as used in OTB), does not answer the question “If the average overlap for T1 is 0.6 and T2 is 0.61, can we say that T2 is better than T1?".

Additionally, besides the methodology, arguably the most important thing in benchmarks are data. The proposed datasets (even though they may be enormous) are picked without a strategy how to balance and control the distribution of visual attributes and object properties. Many of the used sequences are conceptually similar, which makes results biased toward some particular types of visual phenomena (e.g. lighting conditions). The data also does not provide a complete ground truth annotation and visual attributes are per sequence only. The lack of a per frame annotation does not allow detailed and deep analysis of tracking results since the tracker failure in the sequence with multiple visual attributes can be caused by any of the attribute or any of their combinations. Another common problem of tracking datasets is that the objects of interest, in some cases, are not well defined or that the sequences are artificially created, e.g. movie scene cuts or ambiguously defined targets such as fireworks exploding to multiple particles. All the issues mentioned above make these datasets inappropriate for the short-term tracking evaluation.

Lastly, the baseline tracking results of published methods, which define the state-of-the-art results of the benchmarks are not kept up to date and no new trackers are periodically included to advance the performance standard. This makes it hard to monitor the progress of the state-of-the art-results through the years.

To address the aforementioned benchmarks issues a working group was formed, in collaboration with Kristan et al. and collective, and together we proposed a novel benchmark [KML$^+$16] focused on the single target visual tracking. It provides a novel evaluation methodology, a new systematically selected and carefully annotated dataset, a deep result analysis capability and continuously evolving state-of-the-art results along with the growing supporting community. The benchmark is presented in Chapter 3.

2.2 Tracking Methods

This section presents an overview of influential basics of tracking algorithms to the current state-of-the-art methods. The following sub-sections describe the most commonly applied underlying principles of tracking algorithm. Nevertheless, some tracking methods take advantage of multiple different aspects and could be placed in several sub-sections, nevertheless, they will be described in the sub-section of the tracking principle that is the most prominent for the particular method.

2.2.1 Gradient-based Tracking

To the best of my knowledge, the first gradient-based tracking method was proposed by Kanade, Lucas, and Tomasi (*KLT*) [LK81, TK91, ST94]. In their work, brightness constancy between consecutive images is assumed and tracking task is formulated as a template matching using the sum of squared differences as a cost function. The cost function computes the difference between the target template and a patch related by a geometric transformation (usually only translation is considered). By linearizing the effect of the transformation on the image a gradient descent algorithm may be used efficiently to find a local maximum of the cost function. The size of the template (object of interest) defines a trade off for multiple aspects of the method: (i) speed of the optimization, (ii) basin of attraction, i.e. maximum possible motion between consecutive frames that the algorithm is able to estimate and (iii) estimation robustness. For small templates, speed and robustness are high, but the possible motion is small. On the other hand, for large templates (e.g. face) the allowed motion is large but the speed is lower and robustness suffers from the violation of the brightness constancy assumption. Furthermore, additional errors in the motion estimation are introduced due to non-planar objects or perspective transformation (this is partially mitigated for smaller patches since they can be locally well approximated by affine patches).

Multiple methods exploit the simple and fast KLT tracker and build complex algorithm around the KLT method. The Kölsch et al. [KT04] propose an idea of exploiting a collection of KLT trackers in a method called the Flock of Features for a fast hand tracking. It is using KLT as local trackers sampled in a location with a high probability of being the target. The probability is computed for each pixel as probability ratio of belonging to histograms of a foreground and background, which are learned in the first frame and updated online. The method enforces ”flock behavior” [Rey87] to detect failing local trackers and replenish them on places of high probability. The output of the tracker is the median position of the local trackers, which manifests the flocking behavior. Kalal et al. proposed a similar approach called median-flow tracker [KMM10b] that places local trackers (KLT) on a regular grid, i.e. the local trackers cover the object uniformly. Object motion, which is assumed to be well modeled by translation and scaling, is estimated by the median of a subset of local tracker displacement estimates (translation) and the median of the relative change of distance between positions of local tracker pairs (scale). The subset of trackers is select by reliability predictors. Two standard filtering predictors are used, namely the normalized cross-correlation (or sum of squared differences) of the corresponding patches of local trackers and the consistency of the forward-backward procedure, which tracked local trackers in forward and backward direction (i.e. from frame $t-1$ to t and backward) and threshold position error.

Mean-shift (MS) [Che95] was another popular technique used in gradient-based tracking algorithms. The MS maximizes a similarity of target template and candidate template via a gradient descent optimization. The first tracking approach that utilized the MS procedure was proposed by Comaniciu et al. [CRM00]. The method represent the target by a color histogram that approximates the pdf of pixel colors belonging to the target. The pixels values are weighted during histogram extraction by a spatial kernel (e.g. Epanechnikov kernel), which adds more weight to the pixels in a center of the region of interest and smooths the histogram w.r.t. position of the region. The Hellinger distance is employed as a similarity measure between the template and candidate histograms, which well approximates the chi-squared test for histograms, i.e. the goodness of fit for frequency distributions. An efficient gradient descent is used on the similarity measure w.r.t. the spatial parameter to iteratively move the position of the candidate window to the location that maximizes its similarity to the template histogram. This method is very simple to implement and achieves a real-time performance, which leads to large amount of following works that improve the original idea in various aspects, e.g. a scale estimation [PP06, LHJ$^+$07, ZKR08, NZZW12b], a new similarity measure [YDD05] which is more discriminative and allows for more general motion models, incorporating a background information [NZZW12a] in a form of novel pixel weighting scheme in the MS procedure or including texture features [ZLQ08, NZZW09, BDS12] as a complementary source of information for a more precise localization.

2.2.2 Part-based Tracking

Part-based tracking methods models a target by splitting it to smaller parts, which are either smaller ”independent” parts [KT04, ARS06, KL09, KMM10b, ČKL13] with a voting scheme to estimate a single pose of the target or the target model is a combination of multiple global or spatially dependent visual features [SNHY08, FSW12, ČKL13, LHMB16].

Adam et al. [ARS06] introduced the FragTrack tracker, which represents object by multiple patches. Each patch is described by a color histogram and by a relative offset from the center of the target region. During tracking, each patch votes for an object pose by exhaustively comparing its histogram to patches in a given radius. Robust statistics technique is then used to combine heat map votes obtained from target patches. Nejhum et al. [SNHY08] combines global description (a histogram over the whole object) and a number of small rectangular blocks (weighted histograms) placed on the target. To determinate the most probable object location the global model is used on the whole image in a sliding window fashion, after that the local models are used to refine the position. Lastly, an approximate boundary contour is then extracted using graph-cut segmentation and block positions and weights are then updated. Cehovin et al. proposed the LGT [ČKL13] tracker that couples a local and a global visual models. The local model is a set of local patches that geometrically constrains the appearance of the target. The local patches evolve in a probabilistic manner to cope with the target deformation and are managed (added or removed) by the global model. The global model is a probabilistic representation of the target properties such as color, shape and motion. The global model is updated from stable local patches. The mutual learning between the local and global models allows an appearance adaptation while maintaining robustness to model drifting. Position of the object is computed as the average of the local patches positions.

2.2.3 Tracking by Detection

The detection based method gained popularity after Viola and Jones [VJ01] introduced a real-time face detection framework using a sliding window approach with a cascaded classifier. Structure of such a tracking-by-detection framework is shown in Figure 2.3. Since then, many approaches were introduced using either a sliding window technique (e.g. [SLS$^+$10, KMM12, DVM11]) or a local exhaustive search with fast classifiers (e.g. [Avi07, HST11, ZMS14]). The two most popular classifiers, adopted from a machine learning field, were AdaBoost [FS97] (or WaldBoost [SM05] variation) and Support Vector Machine (SVM) [CST00]. On top of those classifiers several popular methods were built.

Avidan [Avi04, Avi07] proposed two tracking by detection approaches. The first one is based on the combination of SVM and optical flow (SVT) [Avi04]. The optical flow is reformulated to optimize SVM score instead of intensity differences. The other method, the Ensemble Tracker [Avi07], combines a large number of weak classifiers

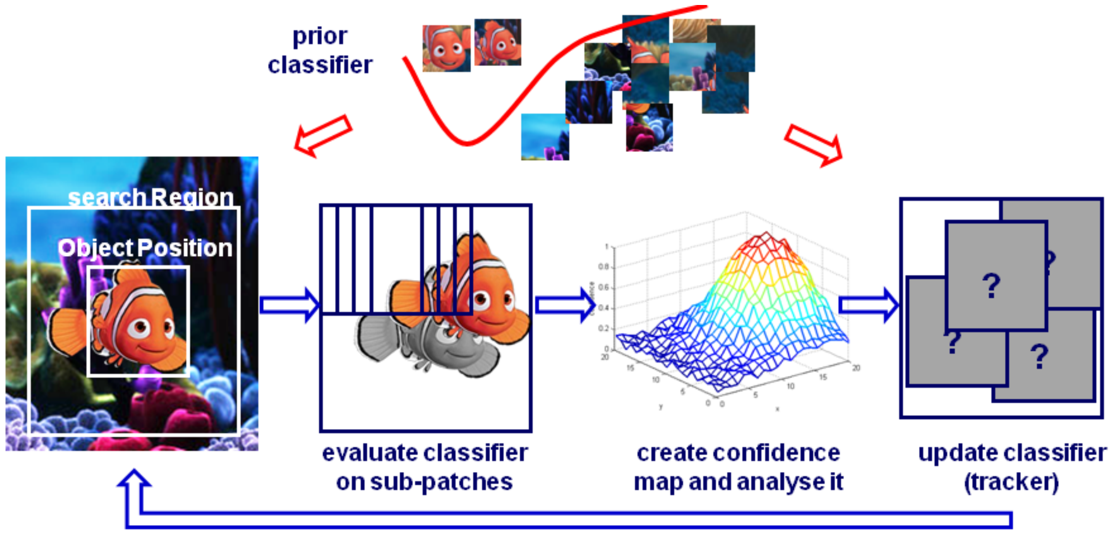

prior classifier
search Region
Object Position
evaluate classifier on sub-patches
create confidence map and analyse it
update classifier (tracker)
?
?
?
?

Figure 2.3: The tracking-by-detection pipeline with model updates. The sliding window approach is used in an arbitrarily sized search region to generate sub-patches for a classifier. The response map is generated from a individual responses of a classifier on sub-patches. From the response map a new position is inferred, e.g. by taking position of the sub-patch with the maximum response. The classifier is then updated by collecting new positive and negative examples, which are labeled based on the previously inferred new position. Image courtesy of Grabner et al. [GLB08].

by the AdaBoost to one strong classifier. The weak classifiers are learned online on 11-dimensional vector of features (i.e. concatenation of local gradient histogram and color). In each frame, every pixel is classified as either target or background resulting in a confidence map on which a mean-shift procedure is run to find a location of the highest density. The weak classifiers are updated (added or removed) using the new position as a positive example and the rest as negative (background) examples.

Hare et al. [HST11] proposed a STRUCK tracker using a kernelized structured output support vector machine (SVM). The classifier is trained directly on the examples of the desired transformation (e.g. translation) given a location from which features are extracted; therefore there is no direct binarization to target/background. This alleviates an issue with a precise separation of training data to the positive and negative group during online learning. In each frame, tentative target windows are generated in the neighbourhood of the previous frame position and are evaluated by the SVM classifier. The window with the highest SVM response is selected and all the evaluated windows are used to retrain the classifier. Zhang et al. [ZMS14](MEEM) keep a pool of multiple SVM classifiers learned from different time spans and chose the one that maximizes an entropy-based cost function. The tracking scheme is similar to previous methods, the classifiers are evaluated exhaustively in the local area around the previous position and the final bounding box is selected based on SVM responses and the entropy-based cost function. Babenko et al. [BYB09] incorporate the multiple instance learning paradigm (MIL) for learning of weak classifiers and combine them similarly as AdaBoost. The MIL assigns the training data to groups, where each group has a label (foreground or background) by majority voting of labels of individual examples in that particular group. This formulation allows to cope with a model drifting during online updates from a self-annotated training samples.

The sliding window approach can also be used globally on the whole image, rendering the methods robust to large motion, scene cuts, an object disappearance and reappearance in an arbitrary location (e.g. during full occlusion or leaving the field of view) as a trade of for increased performance demand and background clutter. Santner et al. [SLS$^+$10] introduce the PROST tracker, where three tracking methods with different rates of appearance adaptation are combined in a cascade to prevent drift due to incorrect model updates. The intermediate stage of the tracker pipeline (modest adaptability) is a sliding window detector using a randomized forest [Bre01] as a classifier. The other two trackers, a completely adaptive optical flow based tracker and non-adaptive NCC tracker, are used to generate training examples for the detector and validate the detector model in a case of drift, respectively. The approach uses simple, hard-coded rules how the final position is selected from the different outputs of the three approaches. Kalal et al. [KMM12] proposed a TLD tracker which combines a sliding window with randomized forest detector with a short-term tracker (median-flow tracker [KMM10b]) that generates so-called P-N events to a learn new object appearance. The output is defined either by the detector or the tracker based on visual similarity to the learned object model. Adaptation of the TLD was proposed by Dinh et al. [DVM11] in their "Context tracker". They use a TLD as a baseline tracker and build a higher logic on top of it, that models an appearance of background that helps with the target recognition (supporters) and background that distracts the tracker (distractors).

The sliding window approaches show promising results, but have two main disadvantages: (i) The sliding window technique is usually used only to cope with translation and scale. The number of search windows increases extensively for more complex object transformations (e.g. rotation), thus limiting the real-time applications if more complex transformations are required. (ii) Every window needs to be evaluated and either accepted or rejected by the classifier, so there is a demand for a fast classifier to keep the high performance at the expense of discriminative power. To address this issues, several methods are proposed which utilize a technique of feature matching that allows more complex transformation and the computational complexity can be controlled by the number of extracted features. Pernici et al. [PDB14] introduce the ALIEN tracker, which extracts SIFT [Low04] features from a whole image and matches them to the model (collection of foreground and background features) to established tentative correspondences between features in the image and the model. Using the RANSAC [FB81] procedure a transformation model is found to map features from the model to the new location in the image. Moreover, a novel scheme for updating the feature model by spatially oversampling the features and filtering of unstable features using a transitive property of feature matches in consecutive frames was proposed. This alleviates the drifting problem and allows for long-term tracking.

Lebeda et al. [LHMB16] introduced the LT-FLO tracker, where new edge-based features are proposed. The tracker is divided to the short-term and long-term parts. The short-term part works locally, extracting features in each frame assuming they are close to the previous position, matching them to the model to established tentative correspondences and using RANSAC to estimate a target transformation. The long-term part of the method monitors the quality of the tracking and in a case of a failure (drop in confidence) it starts a re-detection process which reinitializes the short-term part to previously stored ”correct locations” and the most confident one is used.

2.2.4 Tracking by Segmentation

Segmentation based trackers are characterized by the output in the form of the pixel-wise labeling to object and background. The following three methods [GRB11, DG13, FSW12] combine detection-based approach with a segmentation process which produces the pixel-wise output labeling and simultaneously provide a more precise (then bounding box) feedback to update the detector.

Godec et al. [GRB11] proposed a HoughTrack which utilizes online learned Hough Forest classifier, which is a combination of hough voting and randomized forests [Bre01], and GrabCut [RKB04] segmentation. The Hough Forest is trained by example triplets consisting of extracted features, label (foreground or background) and the displacement vector to the target center (in the case of foreground label). During tracking, a classifier is evaluated in the neighbourhood of the previous position and the voting map of center displacement is used to find the most probable target position. After that, a back-projection of votes that voted for the selected center is used to generate a sparse support of points that are used as a seed to the GrabCut algorithm. The resulting segmentation is outputted and is also used to update the Hough Forest. Similarly to HoughTrack, the Duffner et al. [DG13] proposed a PixelTrack which also relies on hough voting coupled with segmentation. PixelTrack quantizes the color space and learns for each quantization the most probable center location which is used in hough voting during tracking. The backprojection of pixels that contribute to the most frequent vote are used as a seeding for the segmentation algorithm. The segmentation is formulated in a probabilistic manner using recursive Bayesian formulation that incorporates a segmentation from the previous frame and the current observation (the seeds from the hough voting). The final segmentation is used to update the hough voting model.

Fan et al. [FSW12] introduced the Scribble tracker, which combines two main components: (i) Tracking using short-term salient points, discriminative colors and long-term bags-of-patches. (ii) Matting to segment the object. First, short-term salient points (SIFT [Low04]) of the object and background (near the target) are matched, or, if they do not have a good match, tracked by the KLT tracker. Then the rest of the pixels in the area, denoted by the object salient points, are labeled as the object or a background based on their likelihood to belong more likely to the object or background. The pixels labeled as object and object salient points are used as scribbles for the matting segmentation, which gives a final result. In a case of occlusion or object reappearance, the long-term bags-of-patches are used to match learned local patches to the image and find the most likely object pose w.r.t. local patches matching scores and their relative position to each other. An online update is performed by resampling the salient points and updating the discriminative color distributions and bags-of-patches from the area of the precise segmentation from the matting step.

The tracking by segmentation may be also formulated as an estimation of the object outer closed contour, therefore by defining the boundary of the object the segmentation is estimated implicitly. When talking about contour tracking there are two components that needs to be defined, i.e. a contour representation and a energy (cost) of a

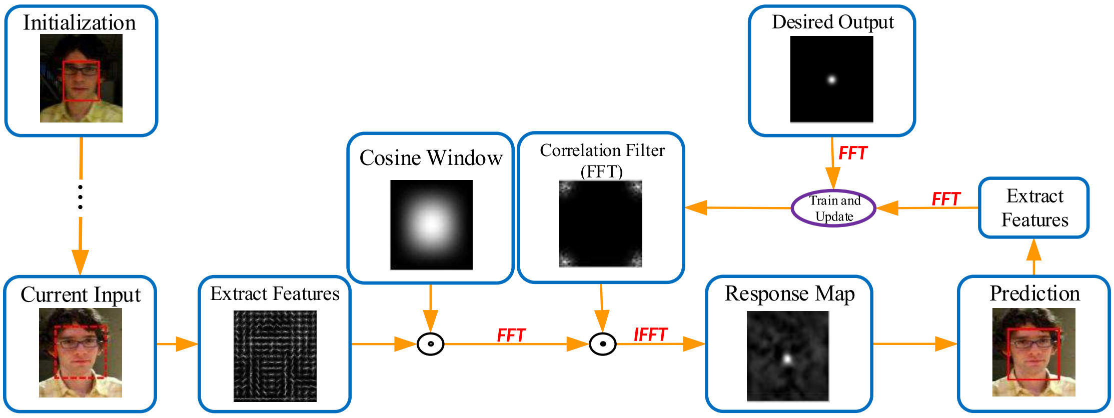


Figure 2.4: The basic correlation tracker pipeline. The tracker is initialized in the first image (red bbox) and a correlation filter is learned by minimizing the difference between filter response and desired output (i.g. Gaussian peak) for the first image. Then, for each consecutive image, features are extracted from a search region (denoted as the whole visible image patch in the figure) and cosine window is applied. The transformed features are element-wise multiplied in the Fourier domain with the learnt filter. The result is converted back to real domain using inverse FFT to produce the response map. The new position, i.e. max peak in response map, is used to update the filter. Image courtesy of Chen et al. [CHT15].

given contour. A contour can be represented as a parametric curve (such as splines) or implicitly by level sets [OS88]. There is number of ways how to define the energy functional which defines the cost of the given contour. The most common functionals are snakes [KWT88], geodesic active contour [CKS97], Mumford–Shah functional [MS89] or region competition [ZLY95]. A variational approach is than used to derive equations for the time evolution of the curves. Examples of tracking approaches using level sets are Paragios et al. [PD00] and Shi et al. [SK05]. Cremers et al. [CTWS02, Cre06] use a spline representation and introduced a prior probabilistic shape model into the energy functional. Rathi et al. [RVTY07] exploit particle filters inside a geometric active contour framework for probabilistic curve evolution.

2.2.5 Correlation Tracking

The correlation trackers originates with the MOSSE tracker, proposed by the Bolme et al. [BBDL10]. There had been several earlier works (UMACE [MVKC87], ASEF [BDB09]) that influenced the MOSSE tracker, but none of them had a significant impact on the tracking community. The MOSSE tracker pose the tracking task as a finding of a filter that when convolved with the image patch produces high response in the target location and, ideally, zero response elsewhere. The tracker is initialized from one image where a search region is chosen around a target which defines the size of the initial filter. The initial filter is learned from the single patch extracted from the search window and the filter is continuously updated during tracking. The learning of the filter is an optimization task minimizing the sum of squared differences between the actual output of the correlation (i.e. correlation between example patch and a filter) and the desired output of the correlation (modeled as compact 2D Gaussian shaped peak). Solving of this task is possible in close form for a single pair (example, response). However, learning from a single example leads to overfitting, because the minimization produces an exact filter with zero error which does not generalize well and almost always leads to tracking failure in consecutive images; therefore, multiple examples are generated by affine warping of the initial region to produce multiple filters that by taking their average render more generalizing filter. Moreover, a regularizer term is added to prevent zero division in cases of low amplitude in some frequencies (i.e. homogeneous areas) to robustify the filter computation. The tracking is performed as a correlation of input image search region, usually extracted from previous target position, and the filter in the Fourier domain representation using a Fast Fourier Transform (FFT) [PTVF92], where the matrix multiplication and division become element-wise operations resulting in extremely fast evaluation of search region subwindows. The output location is found as a peak value of the filter correlation responses of the subwindows. A convex combination is used to incorporate the new filter, learned from the current location and the previously obtained filter to reflect the target appearance changes. The simplified pipeline of correlation tracking methods (which is common for most method) is illustrated in Figure 2.4.

The MOSSE tracker has two main drawbacks: (i) lack of training examples (limited to affine warps) and (ii) only linear combination of the filter and image can be used (by the correlation), which does not allow to use more complex features or multiple channels. Henriques et al. proposed the Circular Structure Kernel (CSK) [HCMB12] tracker which addresses these drawbacks of the MOSSE tracker. Firstly, CSK tracker introduced a circular matrix structure which clarifies the underlaying property of the learning process in Fourier domain. This structure allows to represent all cyclical shifts of the training example (i.e. all shifted subwindows of a search region) at the same time using a matrix representation where each row is a training example represented as a shifted original. Secondly, the circular structure of the data enables the use of the kernel trick [SS02] in the formulation of the optimization task, i.e. use of non-linear kernels in regularized least squares to obtain a filter. Furthermore, a theoretical connection between the MOSSE tracker and newly proposed CSK tracker was shown via a ridge regression framework. In [HCMB15], Henriques et al. extended the previous CSK tracker and introduced a KCF tracker. It newly supports multi-channel features (demonstrated on 32 channels HOG [DT05] features) and provides simpler and more intuitive derivation of learning equations by a diagonal representation of training samples. Since the introduction of KCF, which achieved state-of-the-art performance with processing speed multiple time faster then real-time, a large number of its modification appeared. Most notably DSST [DHSKF14] and SAMF [LZ15] trackers which extended the KCF for scale estimation and incorporated other types of features. Danelljan et al. [DHSKF15](SRDCF tracker) introduced a spatial regularization in the learning process, which limits a boundary effect and penalizes filter coefficients depending on their spatial location. This extension allows to use much larger search region; therefore the tracker is able to track faster objects and is more discriminative to the background since it is trained on a large set of negative examples.

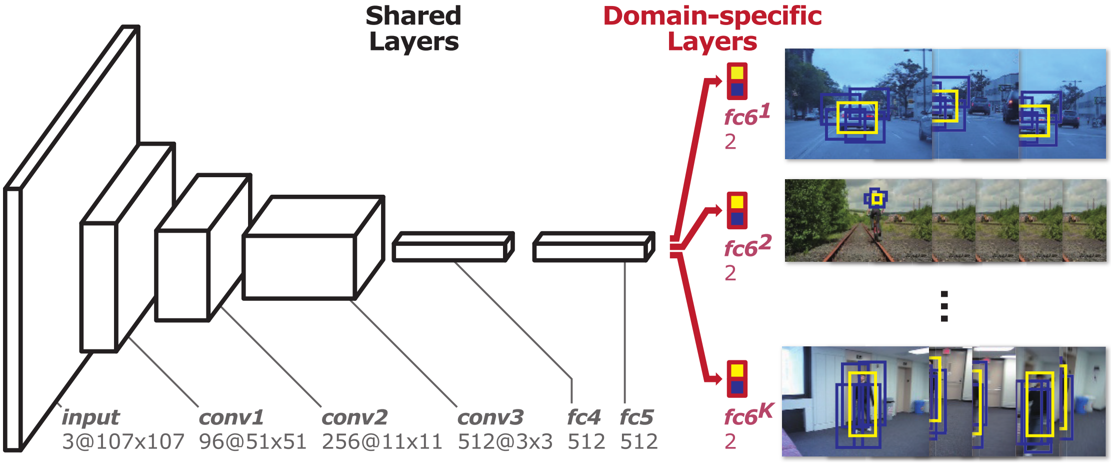


Figure 2.5: The architecture of the MDNet [NH15] tracker. There are three convolutional layers with max pooling which reduce the dimensionality of the data (denoted by y-z size of the blocks) and provide robustness to a spatial perturbation of a input data. The convolution layers are followed by two fully connected layers and Domain-specific (DS) layers which are trained for different object types and combined in evaluation phase. During tracking, the network computes scores for randomly sampled bounding boxes around the previous position. The bounding box with the highest score is selected as the new location of the object and the remaining bounding boxes are classified as background, denoted by yellow and blue color respectively. Image courtesy of Nam et al. [NH15].

2.2.6 Deep Neural Networks

The next category of trackers is based on deep neural networks, especially convolution neural networks (CNN). In the most recent years, there has been a surge of tracking methods based on the deep neural networks and most of them achieves stellar performance (disregarding the processing speed factor). The high performance of CNN based trackers comes from very discriminative and context focused features that are computed in the convolutional layers. A concrete example of direct performance gain by introducing features extracted from convolution layers is the SRDCF [DHSKF15] tracker in the VOT2015 challenge [KML$^+$15] with its ”deep” variation *deepSRDCF* which achieved 10% and 15% performance increase in expected average overlap and robustness respectively.

One of the fully end-to-end trained CNN tracker was proposed by Nam et al., the MDNet [NH15] tracker uses three convolution layers and two fully connected layers followed by a newly proposed Domain-specific (DS) layers, where each DS layer is trained for different domain (an object type) from large sets of tracking video sequences. During tracking the pre-trained DS layers are combined with a new binary classification layer trained online in the current sequence. The MDNet is evaluated in randomly sampled locations around the previous location and the most confident location is used for a bounding box regression that produces the final output. The architecture of the MDNet is shown in Figure 2.5.

Ma et al. [MHYY15] proposed a hierarchical CNN features for tracking using the VGG-Net [SZ14] pre-trained on ImageNet dataset [DDS$^+$09]. They extract features from third, fourth and fifth convolutional layers and learn for each of them a linear correlation filter. These filters are then used to generate response maps and coarse-to-fine approach is used to find the location with the maximum response. Similarly, Qi et al. [QZQ$^+$16] learn ”weak” correlation trackers on features from different layers of VGG-Net and then weighted sum of theirs outputs is the final position. An adaptive Hedge algorithm [CFH09] was introduced to update the weights of the correlation trackers online.

Wang et al. [WOWL15] propose two CNNs, denoted as GNet and SNet, both with the same architecture, i.e. two convolutional layers, on top of a pretrained VGG-Net convolutional layers. The GNet is general network and it captures the category information of the object. The SNet is trained to discriminate between the object and background. Both networks produce a foreground heat map from which an object pose is estimated by locating the maximum peak.

Hong et al. [HYKH15] use pre-trained CNN [GDDM14] in the forward direction for features extraction from sampled position. Each sample is classified by an online trained SVM classifier to label the sample as either object or background. For positively labeled samples the relevant object features, which are identified by the SVM (i.e. positive SVM model weighs), are back-projected through the pre-trained CNN to obtain a saliency maps. The target-specific saliency map is obtained by sequential Bayesian filtering using the saliency maps as observations and is convolved with online learned generative appearance model to generate a dense likelihood map, where the peak denote the target new position.

Chapter 3

Visual Object Tracking Evaluation Methodology

Evaluation and comparison of different tracking approaches is important task in visual object tracking. Every year, tens to hundreds of novel tracking approaches are proposed and the necessity to objectively compare these methods becomes an essential problem. In many cases, the results of published methods cannot be directly compared for two reasons i) each method may have used different set of testing video sequences and ii) different evaluation metric was employed. A unified framework for evaluation and comparison of tracking methods is imperative to alleviate these issues and, consequently, continuous advancement of the state-of-the-art.

This section describes the latest core components of the Visual Object Tracking challenge (VOT) evaluation methodology [KPL$^+$13, KPL$^+$14, KML$^+$15, KML$^+$16]. The core components consist of the performance measures, selection and annotation of the data and the experiment setup. The VOT was formed in 2013 as an attempt to unify the evaluation of the tracking method and to help the advance of the state-of-the-art by annually organizing the visual object tracking challenge as a workshop on a major conference, where the novel tracking methods are compared to each other and the previous state-of-the-art methods. For all resources related to the VOT (datasets, publication, presentations, evaluation toolkit) and information about future workshops and tracking challenges please visit the official website[1].

The rest of the chapter describes the VOT 2015 methodology (the theoretical and experimental validation of the proposed aspects of the performance measure) and on the automatic sequence selection. Please note that the methodology is still evolving by incorporating comments from the CV community, therefore it may differ, in some aspects, from previous/future VOT methodology.

3.1 VOT Methodology[2]

Based on the recent analysis of widely-used performance measures [ČKL14, ČLK15] two weakly-correlated and easily interpretable measures were chosen: (i) accuracy and

[1] `http://www.votchallenge.net/`

[2] Figures in this subsection are courtesy of Matej Kristan.

(ii) robustness. The accuracy at a time-step t is measured as an overlap with the tracker predicted bounding box and the ground truth using the standard definition (described in Section 2.1). The robustness is the number of times the tracker failed, i.e., drifted from the target (= zero overlap with ground truth), and had to be reinitialized. A re-initialization is triggered when the overlap drops to zero.

The reinitialization technique has several advantages, i.e. (i) fully use of all sequence frames, allowed by reinitializing the tracker after failure (ii) provides a theoretically more robust estimation of accuracy (shown in Section 3.3.1 and experimentally validated in Section 3.4.5) (iii) reinitialization-based measures can be used. One of the drawbacks is an introduction of bias into the performance estimation caused by the non-random character of the frames where the tracker is re-initialized. The non-randomness comes from the fact that trackers usually fail during non-standard situations such as illumination changes, an abrupt motion or occlusion and re-initialization of the tracker during such condition will likely result in consequent failures. Therefore, to address this bias, the tracker is re-initialized $N_{\text{skip}} = 5$ frames after the failure. This value was set experimentally from a study of this parameter described in the Section 3.4.3. A similar issue occurs in the accuracy measure, where the measured values right after a re-initialization are artificially higher than during a regular course of tracking. The accuracy stabilizes after a few frames, called the burn-in period. Frames within the burn-in period are labeled as invalid and are not used in the performance estimation. The burn-in period is set to $N_{\text{burnin}} = 10$ frames and the effect of this parameter is studied in Section 3.4.4.

The VOT2015 runs a single type of experiment, where the tracker is initialized in the first frame and then is let run using the reinitialization principle. To accommodate for a stochastic trackers each sequence is tracked 15 times to obtain a better statistics on performance measures. The per-frame accuracy is computed as an average over these runs. Averaging per-frame accuracies gives the per-sequence accuracy, while the per-sequence robustness is computed by averaging failure rates over different runs. To provide a more in-depth result analysis, the performance measures are also computed w.r.t. per-frame attributes, i.e. performance for a given attribute is estimated only from the frames where the attribute is present. The benefits of the per-frame attribute analysis are discussed in Section 3.3.2 and experimentally validated in Section 3.4.6.

In the VOT, the trackers are compared in two ways: (i) by raw values of performance measures and (ii) with each other using a ranking-based methodology akin to [EEVG$^+$14, GJP$^+$12]. The ranking-based methodology was extended to introduce a concept of equally-ranked trackers. The tracker ranking goes as follows; for the computation of trackers ranks two statistical techniques are adopted to made the ranking more stable and robust. Firstly, the individual rank for each tracker is computed using the raw values and greater (or less) operator. A statistical test of the performance differences is employed to modify the ranks to address the subsets of trackers that might be performing equally well and; therefore, they should have the same rank. Since the VOT2015, the minimal rank is used for the subset of equally performing trackers. The measured accuracy is available per frame for each tracker; therefore, a statistical test is applied to determine if the difference in accuracies for each frame is statistically significant. From the preliminary tests, the accuracies do not always follow a normal distribution, hence the Wilcoxon Signed-Rank test as in [Dem06] is applied that tests a null hypothesis

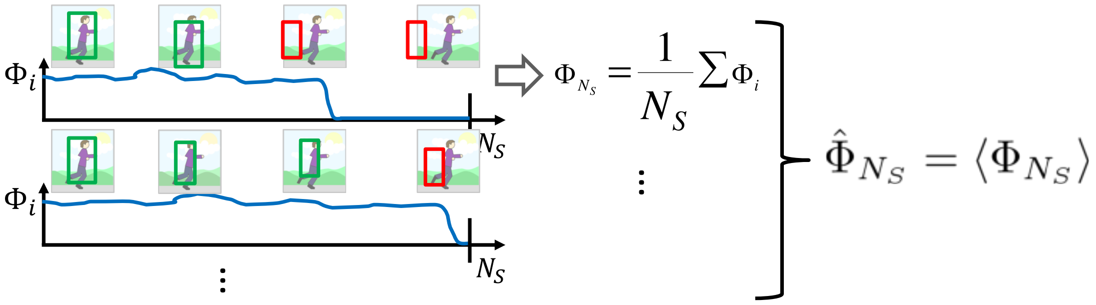

Φ_i
N_S
Φ_i
N_S
$\Phi_{N_s} = \frac{1}{N_S} \sum \Phi_i$
$\hat{\Phi}_{N_S} = \langle \Phi_{N_S} \rangle$

Figure 3.1: The figure illustrates the computation of the expected average overlap from sequences of length N_s.

that differences come from a symmetric distribution with a zero median (see [Nav11] for further details). For the robustness measure, one measure per-sequence per-run (for stochastic trackers) is obtained and they cannot be paired, therefore the Wilcoxon Rank-Sum (also known as Mann-Whitney U-test) [Dem06] is used instead to test the difference in the average number of failures. This is a two-sided rank test which tests the null hypothesis that the number of failures of two independent trackers follows the same distribution (see [Nav11] for further details). Furthermore, the practical difference test (introduced in [KPL$^+$14]) is used to take into account a noisy ground truth annotation and a bias caused by different annotators. This practical test studies if the difference in trackers performance is in a range of the annotation noise. The level of the annotation ambiguity under which the tracker's performance difference is considered negligible is called the practical difference threshold. The estimation of these thresholds for the VOT2015 is described in Section 3.4.2. If both of these tests are positive, then the pair of trackers is considered equally performing.

One of the novelty in the 2015 challenge was introduced to the final ranking of the trackers. Instead of averaging the tracker ranks for robustness and accuracy, which lack an intuitive interpretability, a novel single value measures with more clear interpretation was proposed. The measure can be described as ”expected average overlap on N_s frames long sequence” $\hat{\Phi}_{N_s}$ and is seamlessly computed from the measured values, see Figure 3.1. Normally, to calculate the $\hat{\Phi}_{N_s}$ a large number of sequences of the same (N_s) length would be required, however, this can be efficiently approximated using the VOT reinitialization methodology (illustrated in Figure 3.2). Each tracklet (tracking results from initialization to detected failure) approximates tracking results on sequence of length N_s such that if the tracklet is longer than N_s frames, then the first N_s frames are used, otherwise, the tracklet is padded with zero overlap after failure. Using this approach, lot of tracklets are collected for different N_s and $\hat{\Phi}_{N_s}$ is computed more robustly.

For the final ranking $\hat{\Phi}$ is an average of $\hat{\Phi}_{N_s}$ on interval (N_{lo}, N_{hi}). The interval is inferred from the data and depends on the sequences length distribution of the dataset. The $\hat{\Phi}_{N_s}$ is visualized by the expected average overlap curve, see Figure 3.3 where this curve is illustrated together with the estimation of the interval (N_{lo}, N_{hi}) from the data using a kernel density estimation method.

Additionally to the presented performance measures, the VOT deems the speed as an

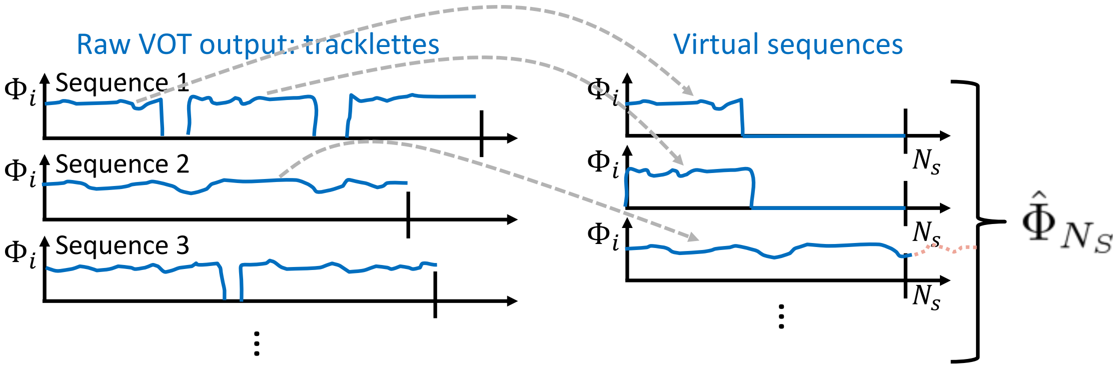
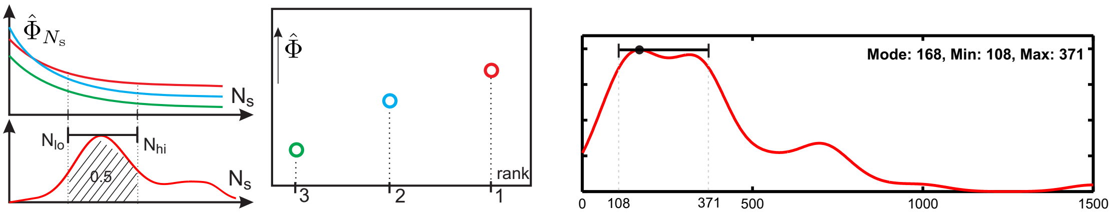


Figure 3.2: The figure illustrates the approximated computation of the expected average overlap using the VOT reinitialization methodology for sequences of length N_s.



(a) (b)

Figure 3.3: The figure (a) illustrates the expected average overlap curve $\hat{\Phi}_{N_s}$ for different values of N_s and respective ranking (right subplot) for interval (N_{lo}, N_{hi}). The figure (b) shows the distribution of the sequence length of the VOT2015 dataset and the estimated interval (N_{lo}, N_{hi}).

important property of a tracking algorithm, therefore, it is also measured and reported by the evaluation toolkit. To reduce a bias caused by different hardware parameters of evaluating machines the speed is reported in a novel equivalent filter operations (EFO) speed unit that was introduced in the VOT2014 [KPL$^+$14]. This unit normalizes the speed in terms of predefined filtering operation that are automatically computed on each machine prior a tracker evaluation.

Results in the VOT challenges are visualized by the accuracy-robustness (AR) plots proposed by [ČKL14], where each tracker is represented as a point in the 2D space and by the proposed expected average overlap curve. These plots are used for a raw-valued accuracy and robustness and also for trackers with respect to their accuracy, robustness and the final $\hat{\Phi}$ ranks. The raw values of robustness are converted to the (0,1) range representing the probability of tracker failure after S frames. The parameter S does not change the ordering of the trackers but only affects the scaling (stretching of a point in robustness axis). The results for the VOT2015 challenge, using the described methodology, are shown in Section 3.5.

3.2 Automatic Dataset Construction

The VOT2013 [KPL$^+$13] and VOT2014 [KPL$^+$14] introduced a semi-automatic sequence selection methodology to construct a dataset rich in visual attributes but small enough to keep the time for performing experiments reasonably low. In the VOT2015, the methodology is extended such that the sequence selection is fully automated and that the selection process focuses on sequences that are likely challenging to track.

The dataset was prepared as follows. The initial pool of sequences was created by combining the sequences from existing datasets OTB [WLY13] (51 sequences), ALOV [SCC$^+$13b] (315 sequences), PTR [VNM13] and over 30 additional sequences from other sources summing to a set of 443 sequences. After a removal of duplicate sequences, grayscale sequences and sequences that contain objects with area smaller than 400 pixels, we obtained 356 sequences. The new automatic sequence selection protocol requires approximate annotation of targets in all sequences by bounding boxes. For most sequences, the annotations already existed and we annotated the targets with axis-aligned bounding boxes for the sequences with missing annotations. Next, the sequences were automatically clustered according to their similarity in terms of the following globally calculated sequence visual attributes:

1. *Illumination change* is defined as the average of the absolute differences between the object intensity in the first and remaining frames.

2. *Object size change* is the sum of averaged local size changes, where the local size change at frame t is defined as the average of absolute differences between the bounding box area in frame t and past fifteen frame.

3. *Object motion* is the average of absolute differences between ground truth center positions in consecutive frames.

4. *Clutter* is the average of per-frame distances between two color histograms: one extracted from within the ground truth bounding box and one from an enlarged area (by factor 1.5) outside of the bounding box.

5. *Camera motion* is defined as the average of translation vector lengths estimated by key-point-based RANSAC between consecutive frames.

6. *Blur* was measured by the Bayes-spectral-entropy camera focus measure [KPPK05].

7. *Aspect-ratio change* is defined as the average of per-frame aspect ratio changes. The aspect ratio change at frame t is calculated as the ratio of the bounding box width and height in frame t divided by the ratio of the bounding box width and height in the first frame.

8. *Object color change* defined as the change of the average hue value inside the bounding box.

9. *Deformation* is calculated by dividing the images into 8×8 grid of cells and computing the sum of squared differences of averaged pixel intensity over the cells in current and first frame.

10. *Scene complexity* represents the level of randomness (entropy) in the frames and it was calculated as $e = \sum_{i=0}^{255} b_i \log b_i$, where b_i is the number of pixels with value equal to i.

11. *Absolute motion* is the median of the absolute motion difference of the bounding box center points of the first frame and current one.

Note that the first ten attributes are taken from the VOT2014 [KPL$^+$14, KML$^+$16], with the attributes *object size* and *object motion* redefined to make their calculation more robust. The eleventh attribute (*absolute motion*) was newly introduced in [KML$^+$15].

To reduce the influence of the varied scales of the attributes a binarization procedure was applied. A k-means clustering with $k = 2$ was applied to all values of a given attribute, and thus each value was assigned a value, either zero or one. In this way each sequence was encoded as an 11D binary feature vector and the sequences were clustered by the Affinity propagation (AP) [FD07] using the Hamming distance. The only parameter in AP is the exemplar prior value p, which was set according to the rule-of-thumb proposed in [FD07]. In particular, we have set $p = 1.25\alpha_{\text{sim}}$, where α_{sim} is the average of the similarity values among all pairs of sequences. This resulted in $K = 28$ sequence clusters, where each cluster k contained a different number of sequences N_k. The clustering stability was verified by varying the scaling value in the range 1.2 to 1.3. The number of clusters varied in the range of ± 3 clusters, indicating a stable clustering at the chosen parameter value.

The goal of the sequence selection is to obtain a dataset of size M in which the following five visual attributes specified in VOT2014 are sufficiently well represented: (i) occlusion, (ii) illumination change, (iii) motion change, (iv) size change, (v) camera motion. The binary attributes were concatenated to form a feature vector $\mathbf{f}_i$ for each sequence i. The global presence of four of these attributes, except occlusion, is indicated by the automatically calculated binarized values that were used for clustering. All sequences were manually inspected and occlusion was indicated if the target was at least partially occluded at any frame in the sequence. To estimate the sequence tracking difficulty, three well performing but conceptually different trackers (FoT [VM14], ASMS [VNM13], KCF [HCMB15]) were evaluated using the VOT2014 methodology on the approximately annotated bounding boxes. In particular, the raw accuracy (average overlap) and raw robustness (number of failures per sequence) were computed for each tracker on each sequence and quantized into ten levels (i.e., into the interval [0,9]). The quantized robustness was calculated by clipping the raw robustness at nine failures and the quantized accuracy was computed by $9 - \lfloor 10\Phi \rfloor$, where Φ is the VOT accuracy. The final tracking difficulty measure was obtained as the average of the quantized accuracy and robustness.

With the $n = 5$ global attributes and tracking difficulty estimated for each sequence, the automatic sequence selection algorithm proceeded as follows. First, the most difficult sequence from each cluster is selected as an initial pool of sequences and a maximum number of samples $\{S_k\}_{k=1}^{K}$ for each cluster k is calculated. From the selected pool of sequences, the weighted balance vector $\mathbf{b}^0$ is computed and normalized afterward. The balance vector controls the attribute representation inside the pool of selected sequences. We use weights to account for the unbalance distribution of the attributes in the dataset and compute them as follows $\mathbf{w} = N_s / \sum_i \mathbf{f}_i$, where N_s denotes the number of sequences. The vector $\mathbf{w}$ have lower values for to the attributes that are most common, therefore would always over-represented and the sequence without this attribute would be selected most of the time (e.g. object motion attribute). After initialization, the algorithm iterates until the number of selected sequences reaches the desired number M ($M = 60$ in VOT2015). In each iteration, the algorithm computes the attributes that are least represented, **aw**, using a small hysteresis so that multiple attributes can be chosen. Then, the Hamming distance between the desired attributes **aw** and all sequences is computed, excluding the sequences already selected and the sequences that belong to a cluster, which has already S_k sequences selected in the pool. From the set of most attribute-wise similar sequences, the most difficult one is selected and added to the pool. In the end, the balance vector is recomputed and the algorithm iterates again. The sequence selection algorithm is summarized in Algorithm 1.

Algorithm 1: Sequence sampling algorithm

```
Input  : $N_s$, $M$, $K$, $\{N_k\}_{k=1}^{K}$, $\{\mathbf{f}_i\}_{i=1}^{N_s}$, $\mathbf{w}$
Output: ids
Initialize, $t = 0$
    $\{S_k\}_{k=1}^{K}$, $S_k = \lfloor \frac{N_k M}{N_s} \rfloor$
    select the most difficult sequence from each cluster $\mathbf{ids}^0 = \{\mathrm{id}_1, \dots, \mathrm{id}_K\}$
    $\mathbf{b}^0 = \mathbf{w} \sum_{i \in \mathbf{ids}} \mathbf{f}_i$, $\mathbf{b}^0 = \mathbf{b}^0 / |\mathbf{b}^0|$
Iterate, $t = t + 1$
while $|ids| < M$ do
    $\mathbf{aw}= (\mathbf{h} < \min(\mathbf{h}) + \frac{0.1}{n})$, where $\mathbf{h} = \frac{\mathbf{b}^{t-1}}{\max(\mathbf{b}^{t-1})}$
    $\{\mathrm{id}_1, \dots\} = \mathrm{argmin}_i \; \mathrm{dist}(\mathbf{f}_i, \mathbf{aw})$
    s.t. if $i \in$ cluster $k$ then $|$cluster $k \cap \mathbf{ids}^{t-1}| < S_k$
    select the most difficult sequence $\mathrm{id}^* \in \{\mathrm{id}_1, \dots\}$
    $\mathbf{ids}^t = \mathbf{ids}^{t-1} \cup \{\mathrm{id}^*\}$
    $\mathbf{b}^t = \mathbf{w} \sum_{i \in \mathbf{ids}} \mathbf{f}_i$, $\mathbf{b}^t = \mathbf{b}^t / |\mathbf{b}^t|$
end
```

As in the VOT2014, we have manually or semi-automatically labeled each frame in each selected sequence with five visual attributes: (i) occlusion, (ii) illumination change, (iii) motion change, (iv) size change, (v) camera motion. In case a particular frame did not correspond to any of the five attributes, we denoted it as (vi) unassigned. To ensure the annotation quality, all frames were annotated by an expert and then verified by another expert. Note that these labels are not mutually exclusive. For example, most frames in the dataset contain a camera motion.

The relevant objects in all sequences were manually re-annotated by rotated bounding boxes. The annotation guidelines were predefined and distributed among the annotators. The bounding boxes were placed such that they approximated the target well, with a large percentage of pixels within the bounding box (at least $> 60\%$) belonging to the target. Each annotation was verified by two experts and corrected if necessary. The resulting annotations were then processed by approximating the rotated bounding boxes by axis-aligned bounding boxes if the ratio between the shortest and largest box edge was higher than 0.95 since the rotation is ambiguous for approximately round objects. The processed bounding boxes were again verified by an expert. The preview of the selected sequences is shown in the Fig. 3.4. The automatic sequence attribute annotation, clustering and the final dataset selection is a part of the VOT toolkit and the source code

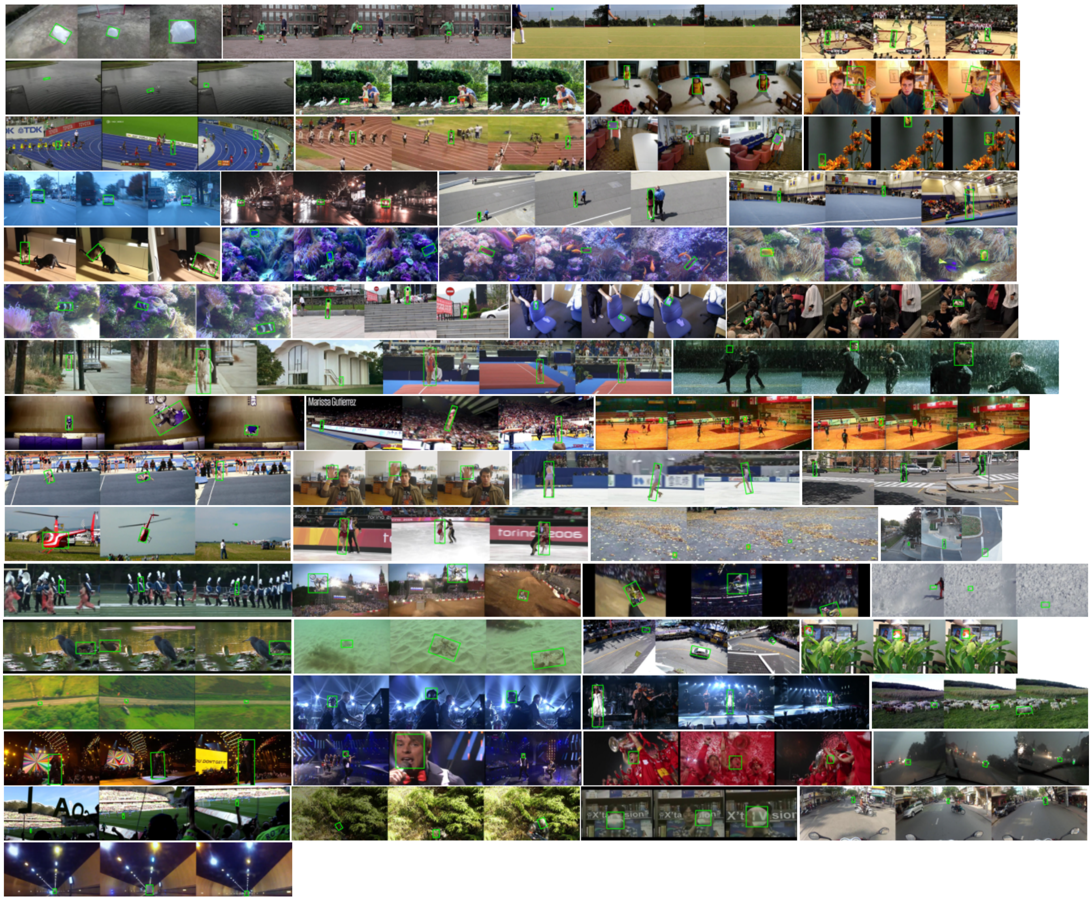


Figure 3.4: Preview of the automatically selected sequences for the VOT2015 dataset. Green box outlines the objects of interest.

is available at `https://github.com/votchallenge/vot-toolkit/`.

3.3 Theoretical Background of the Methodology[3]

In this section, the key parts of the VOT methodology are theoretically described and the benefits of the proposed evaluation methodology compared to the existing ones are discussed. The theory is experimentally validated in the next section. There are two main differences in the evaluation methodology between the VOT and the other two main benchmarks [WLY13, SCC$^+$13b]; the VOT employs an automatic reinitialization technique in case a tracker failure is detected and, secondly, the VOT use a per-frame attribute annotations. The following two major differences are theoretically defined and their effect is discussed in the following sub-sections.

[3] This section was mainly written by Matej Kristan in the collaborative work [KML$^+$16] and was included here to make the thesis self-contained and to emphasized the advantages of the VOT design over standardly used techniques.

3.3.1 The importance of re-initialization

To establish some theoretical results on performance evaluation with or without applying re-initializations, the following thought experiment is considered. Assume a tracker is tested on a set of N sequences, each N_s frames long. A sequence j contains a critical point at the frame $\alpha_j N_\mathrm{s}$, where a tracker fails with probability p, i.e., it drifts and remains off the target for the remaining part of the sequence. During a successful period of tracking, the per-frame overlaps are sampled from a distribution with mean μ_A and variance σ_A^2. After the failure, the overlaps fall to zero, i.e., they are sampled from a distribution with $\mu_b = 0$ and $\sigma_B^2 = 0$. A critical point can occur anywhere in the sequence with equal probability, meaning that these points are distributed uniformly along the sequence, i.e., $\alpha_j \sim \mathcal{U}(0,1)$. A tracker is run on each sequence and a set of N per-sequence average overlaps $\{M_j\}_{j=1:N}$ is calculated. The final performance is reported as the average over the sequences, i.e., an overall average overlap $M = \frac{1}{N}\sum_{j=1:N} M_j$. The aim of the estimator (evaluation methodology) is to recover the hidden average performance μ_A. In the following we will study the expected value and the variance of the output M depending on whether the tracker is re-initialized at failure (WIR) or the failure is ignored (NOR).

The NOR-based methodologies ([WLY13, SCC$^+$13b]) do not detect the failures and the overlaps after the failure affect the estimate of the actual overlap μ_A. Alternatively, the WIR-based methodology (our approach) detects a failure, skips Δ frames and re-initializes the tracker. It can be shown that the expected value $\langle M_\mathrm{NOR}\rangle$ and the variance $var(M_\mathrm{NOR})$ of the overall overlap M_NOR estimated without re-initialization on the theoretical tracking experiment are

$$\langle M_\mathrm{NOR}\rangle = \mu_A(1-\frac{p}{2}), \tag{3.1}$$

$$\mathrm{var}(M_\mathrm{NOR}) = \frac{(2-p)\sigma_A^2}{2NN_\mathrm{s}} + \frac{p(4-3p)\mu_A^2}{12N}, \tag{3.2}$$

while the expected values and variance for the overall overlap estimated by WIR, i.e., M_WIR, are

$$\langle M_\mathrm{WIR}\rangle = \mu_A, \tag{3.3}$$

$$\mathrm{var}(M_\mathrm{WIR}) = \sigma_A^2 \frac{N_\mathrm{s} - \Delta(1-p)}{NN_\mathrm{s}(N_\mathrm{s}-\Delta)} \leq \mathrm{var}(M_\mathrm{NOR}). \tag{3.4}$$

Please see the outline of derivation in appendix of [KML$^+$16].

The following observations can be deduced from Eqs. (3.1-3.4). The NOR estimator is biased increasingly with the probability of failing at a critical point. If critical points always cause a failure, i.e., $p = 1$, then the overall average estimated by the NOR is half the true overlap. On the other hand, the WIR estimator is unbiased, recovers the true hidden overlap, and the mean does not depend on the critical points. The variance of the NOR estimator depends both on the variance of overlaps during successful track as well as the hidden overlap μ_A. This leads to a large variance for trackers that track at high overlap and fail at critical points. On the other hand, the variance of the WIR does not show this effect and is always lower than for NOR, i.e., $\mathrm{var}(M_\mathrm{WIR}) \leq \mathrm{var}(M_\mathrm{NOR})$.

The asymptotic properties of the estimators are visualized in Figure 3.5 w.r.t. the number of test sequences N for parameters $\mu_A = 0.63$, $\sigma_A = 0.4$, $N_\mathrm{s} = 150$, $p = 0.5$,

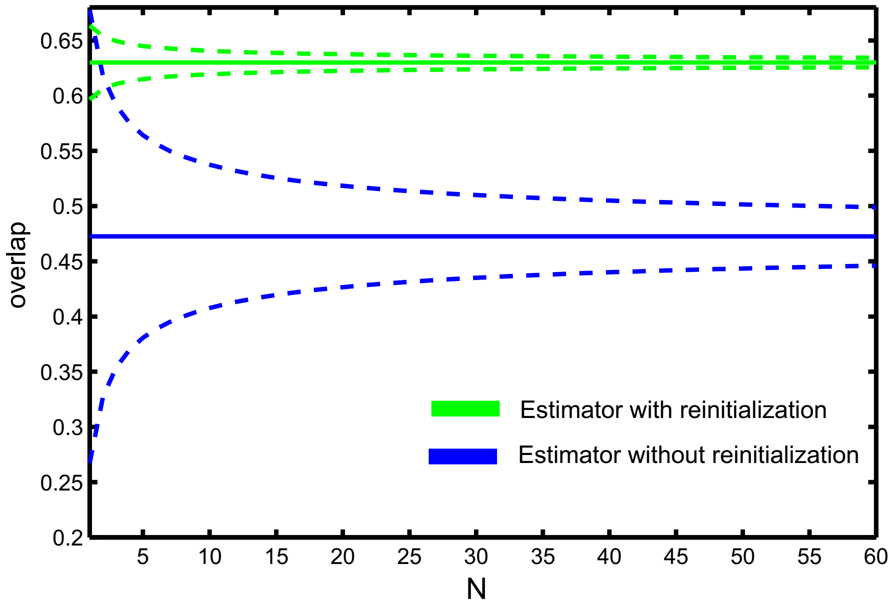


Figure 3.5: Effects of re-initialization in performance estimators. The expected values and standard deviations of the estimators are shown in solid and dashed lines, respectively.

$\Delta = 15$. Note that the WIR estimator is indeed asymptotically unbiased, while the NOR is biased toward a lower overlap values. Furthermore, the variance of the WIR is significantly smaller than that of NOR and decreases faster than for WIR, which is primarily due to the second term in $\mathrm{var}(M_{\mathrm{NOR}})$ (3.1), i.e., lack of re-initializations in NOR. A practical implication is that the methodologies like [WLY13, SCC$^+$13b] require much more sequences than our methodology to produce a similarly small variance of the estimate and their estimate will always be much more biased than ours when failures occur. Note that our theoretical model assumes sequences of equal length. If this constrained was further relaxed such that some sequences were allowed to be significantly longer than the others, it would not affect the WIR estimator, but would substantially increase the variance of the NOR even further.

3.3.2 The importance of per-frame annotation

To study the impact of visual property annotation strategies, we will assume running a tracker on a dataset in which N sequences contain a particular attribute, e.g., an illumination change. The aim is to estimate tracking performance on this visual attribute. A tracker is thus run on each of N sequences, recovering the set of per-sequence overlaps $\{M_j\}_{j=1:N}$, and the average of these is reported as an overall performance, i.e., $M = \frac{1}{N}\sum_{j=1:N} M_j$. For ease of exposition assume that each sequence contains N_{A} frames with illumination change and the remaining $N_{\mathrm{B}} = \eta N_{\mathrm{A}}$ frames contain the other attributes. Thus the per-frame overlaps during the N_{A} frames can be described as samples from a distribution with mean μ_{A} and variance σ_A^2, while the per-frame overlaps in the remaining N_{B} frames are governed by a distribution with mean μ_{B} and variance σ_B^2. For clarity of the analysis, we will assume that there are no critical points in any sequence, i.e., a tracker never fails during tracking, and that the variances σ_A^2 and σ_B^2

are equal.

A global visual property annotation strategy (GLA) (e.g., [WLY13, SCC$^+$13b]) calculates overall per-visual property performance M_{GLA} using all the frames in sequences that contain at least one frame with the considered visual property. Alternatively, the per-frame annotation strategy (PFA) (our approach) considers only frames annotated with a particular visual attribute to estimate the performance M_{PFA}. Note, however, that some frames may be incorrectly annotated. From the perspective of bias in state estimation, the most critical frames are those that are incorrectly annotated as the considered attribute. Let's assume, that in each sequence, a set of βN_{A} are added as false annotations to the correctly annotated N_{A} frames. With these definitions, it is easy to show that the mean and variance of the M_{GLA} estimator are

$$\langle M_{\mathrm{GLA}} \rangle = \frac{1}{1+\eta}\mu_A + \frac{\eta}{1+\eta}\mu_B, \tag{3.5}$$

$$\mathrm{var}(M_{\mathrm{GLA}}) = \frac{1}{N N_{\mathrm{A}}(1+\eta)}\sigma_A^2, \tag{3.6}$$

while the mean and variance for the and M_{PFA} estimator are,

$$\langle M_{\mathrm{PFA}} \rangle = \frac{1}{1+\beta}\mu_A + \frac{\beta}{1+\beta}\mu_B, \tag{3.7}$$

$$\mathrm{var}(M_{\mathrm{PFA}}) = \frac{1}{N N_{\mathrm{A}}(1+\beta)}\sigma_A^2. \tag{3.8}$$

According to equations (3.5,3.8) both estimators are biased, but the bias in M_{GLA} is much greater than the bias in M_{PFA}. For example, assuming sequence lengths $N_{\mathrm{S}} = 150$, with $N_{\mathrm{A}} = 50$ properly labelled frames and five frames per sequence mislabelled, results in $\eta = 2$ and $\beta = 0.1$. This means that M_{GLA} is biased with $0.67\mu_B$, while the bias of M_{PFA} is only $0.09\mu_B$. In fact, since typical sequences contain only small subsets of frames with particular visual attribute, (3.5) shows that the M_{GLA} estimator reflects performance that is dominated by the other visual attributes, thus significantly skewing the per-visual attribute performance evaluation. Note that the variance of the M_{GLA} is lower than that of M_{PFA} by a constant $\frac{1+\beta}{1+\eta}$ since it applies more frames. Nevertheless, the variances of both estimators decrease linearly with factor $N N_{\mathrm{A}}$. A practical implication of these results is that per-frame annotation of moderately-sized dataset (our approach), even with a reasonable number of mislabelled frames, provides a much better estimate of true per-visual attribute performance than a per-sequence labeled large dataset (methodologies used in [WLY13, SCC$^+$13b]).

3.4 Experimental Validation of the Methodology[4]

The VOT methodology introduced several techniques and parameters that effect the performance measures. In this section, these techniques are experimentally validated, the impacts of individual parameter choices are studied and reasoning for particular choices is provided.

[4] The results of individual experiments are taken from [KML$^+$16], where the author of this thesis was responsible for conducting most of the experiments. The summaries were written mainly by Matej Kristan.

3.4.1 Influence of difference tests

The proposed methodology applies tests of performance equivalence by testing statistical and practical differences in tracker performance. In the absence of these tests, trackers that perform slightly differently in average values of performance measures would be assigned different ranks even tough the difference in performance might not be statistically significant or below the annotation noise level (practical difference). To quantify the variations in ranks, we sampled 50 random subsets of 15 sequences from VOT2014 dataset, ranked DSST, KCF, SAMF, CT, FRT and Struck on all subsets and computed the average of the rank variances over all trackers. Table 3.1 reports the rank variations for sequence-pooled and attribute-normalized ranking. The difference tests consistently reduce the variance in both setups.

	accuracy				robustness			
	Seq. pool.		Att. norm.		Seq. pool.		Att. norm.	
	T	N	T	N	T	N	T	N
var	0.1	0.11	0.26	0.31	0.07	0.1	0.09	0.34

Table 3.1: Rank variance (var) with (T) and without (N) difference tests for accuracy and robustness computed for sequence-pooled (Seq. pool.) and attribute-normalized (Att. norm.) setting.

3.4.2 Estimation of practical difference thresholds

The per sequence practical difference thresholds [KPL+14] were estimated using independent (four) annotators each annotating the same selected (five) frames in each sequence by the axis-aligned bounding boxes. This annotation repeats three times. By computing overlaps among all bounding boxes per frame, a set of 3300 samples of differences was obtained per sequence and used to calculate the practical difference thresholds. Figure 3.6 shows boxplots of difference distributions w.r.t. sequences alongside with examples of the annotations.

3.4.3 Influence of the re-initialization frame skipping

The effect of the N_{skip} values was quantified by re-running several of the state-of-the-art trackers using the VOT2014 dataset. The number of failures and robustness ranks w.r.t. the skipping values N_{skip} are shown in Table 3.2. The number of failures most significantly changes between one to three skipped frames and remains stable with increasing N_{skip}. The relative changes are consistent across trackers. This is confirmed by the ranking, which remains stable.

3.4.4 Estimation of the burn-in period

A study was designed to estimate the burn-in period. Seven trackers were run with re-initialization on the VOT2013 dataset. After each re-initialization, we recorded the per-frame overlaps with the ground truth (an overlap sequence). Using this protocol we

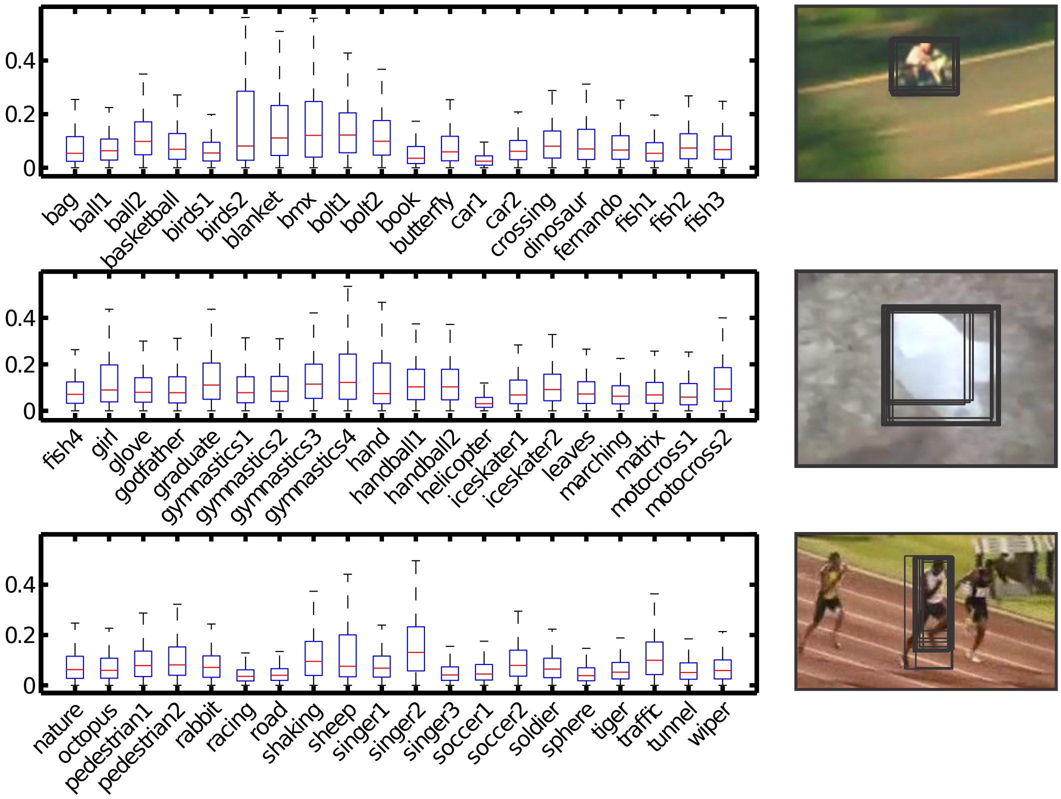


Figure 3.6: Box plots of per sequence ground truth practical differences and examples of annotation variation.

N_{skip}	R	DSST	KCF	SAMF	CT	FRT	Struck
1	raw	1.32	2.04	1.44	1.93	3.76	3.28
3	raw	1.12	1.84	1.56	1.90	3.68	2.76
5	raw	1.16	1.44	1.36	1.57	3.36	2.72
7	raw	1.16	1.56	1.36	1.55	3.48	2.36
9	raw	1.00	1.52	1.16	1.54	2.96	2.28
1	rank	2.58	3.32	2.74	3.40	5.06	3.86
3	rank	2.44	3.18	3.02	3.28	5.26	3.82
5	rank	2.64	3.02	2.94	3.34	5.16	3.90
7	rank	2.70	3.12	2.86	3.20	5.38	3.74
9	rank	2.60	3.32	2.82	3.36	4.94	3.96

Table 3.2: Robustness raw and rank values for different values of frames skipped N_{skip}.

obtained 3249 overlap sequences, which were averaged into a single average overlap sequence shown in Figure 3.7. The rate of temporal change in the overlap is characterized by the derivative of this sequence (also shown in Figure 3.7). It is apparent that the rate of overlap change stabilizes at ten frames after re-initialization. The burn-in period was therefore set to $N_{\text{burnin}} = 10$ frames in the VOT methodology.

To verify the selected burn-in period a quantitative analysis was performed on the VOT2014 dataset using several state-of-the-art trackers. Table 3.3 shows the average

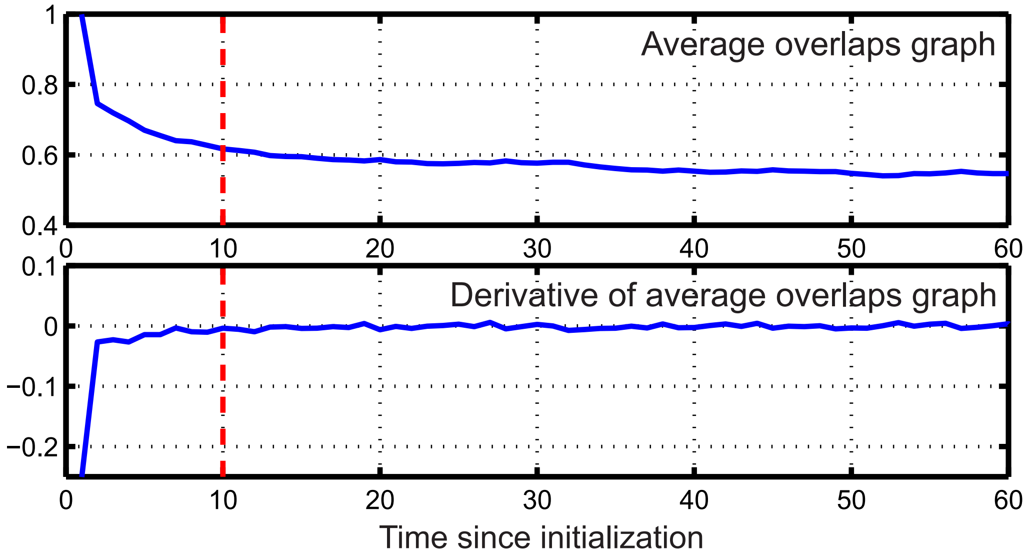


Figure 3.7: Overlaps after re-initialization averaged over a large number of trackers and many re-initializations (top) and the derivative of this graph with respect to time (bottom). The derivative becomes negligible after 10 frames.

accuracy for different values of the burn-in period. The average accuracy is, as expected, slightly reduced when excluding the frames from the burn-in period. The extent of the drop in accuracy is larger for trackers that fail more often.

N_{burnin}	**DSST**	**KCF**	**SAMF**	**CT**	**FRT**	**Struck**	**Average**
0	0.6293	0.6386	0.6213	0.4273	0.4871	0.5167	0.5534
2	0.6285	0.6378	0.6205	0.4248	0.4838	0.5143	0.5516
4	0.6273	0.6369	0.6195	0.4209	0.4786	0.5103	0.5489
6	0.6264	0.6370	0.6191	0.4183	0.4749	0.5071	0.5471
8	0.6258	0.6376	0.6192	0.4165	0.4726	0.5047	0.5461
10	0.6256	0.6385	0.6198	0.4149	0.4711	0.5029	0.5455

Table 3.3: Influence of different burn-in values on raw accuracy.

3.4.5 Effects of re-initialization

The theoretical comparison of estimators (Section 3.3.1) that apply re-initialization, (M_{WIR}), and those that do not, M_{NOR}, was evaluated experimentally. Each tracker was run on all sequences in the VOT2014 dataset once with re-initializations and once without. A set of K sequences was randomly sampled and the average overlap was computed on this set for each estimator. The process was repeated thousand times for $K < 24$ to estimate the mean and variance of the average overlap. For $K = 24$ there are only 25 possible different combinations of sequences; therefore the mean and variance were computed only on these. Table 3.4 shows results for varying K. Due to sampling with replacement, sequences were repeated across the sets, which means that the variance was underestimated, especially for $K = 24$. The actual variances of the average accuracy are expected to be higher. Nevertheless, the relative trends are as

K	R	DSST	KCF	SAMF	CT	FRT	Struck
5	N	0.49(.14)	0.49(.15)	0.50(.14)	0.23(.09)	0.24(.09)	0.35(.11)
	Y	0.63(.09)	0.64(.08)	0.63(.08)	0.43(.07)	0.49(.06)	0.52(.08)
10	N	0.50(.10)	0.49(.10)	0.51(.10)	0.24(.06)	0.24(.06)	0.36(.08)
	Y	0.63(.06)	0.64(.06)	0.63(.06)	0.43(.05)	0.48(.04)	0.52(.06)
15	N	0.50(.08)	0.49(.08)	0.51(.08)	0.24(.05)	0.25(.05)	0.36(.06)
	Y	0.63(.05)	0.64(.05)	0.63(.05)	0.43(.04)	0.49(.04)	0.52(.05)
20	N	0.50(.07)	0.50(.07)	0.52(.07)	0.24(.04)	0.24(.04)	0.36(.06)
	Y	0.63(.04)	0.64(.04)	0.63(.04)	0.43(.03)	0.49(.03)	0.52(.04)
24	N	0.50(.06)	0.50(.07)	0.52(.06)	0.24(.04)	0.24(.04)	0.36(.05)
	Y	0.63(.04)	0.64(.04)	0.63(.04)	0.43(.03)	0.49(.03)	0.52(.04)

Table 3.4: Performance of estimators with re-initialization, $\mathrm{Y(WIR)}$, and without re-initialization, $\mathrm{N(NOR)}$ indicated in the column denoted by R. Average overlap is shown for each tracker and the standard deviation is shown in brackets.

predicted by the theoretical model. The means of M_{NOR} are consistently lower than for M_{WIR}, which is especially evident for trackers that frequently fail, e.g., FRT and CT. Moreover, the variance of M_{NOR} is consistently higher than for M_{WIR} across all trackers. The Wilcoxon paired tests showed that both types of differences are not equal with statistical significance $p < 0.01$.

3.4.6 Importance of the per-frame annotation

The properties of estimators that apply a per-frame visual attribute annotation, M_{GLA}, and the estimators that apply only a per-sequence annotation, M_{PFA}, were estimated using a similar experiment as in the previous section. For a fair comparison, re-initialization was utilized in all experiments. The results for $K = 24$ sequences are visualized in Figure 3.8 and confirm the predictions from our theoretical model. The variance of per-attribute M_{GLA} is generally slightly smaller than M_{PFA} since M_{GLA} uses more frames in estimation, of which many might not contain the attribute in question, making the M_{GLA} estimator strongly biased toward the global mean. This bias is also reflected in the dispersion of per-attribute values around their global mean, which is greater for M_{PFA} than for M_{GLA}. This means that M_{GLA} is much weaker at making predictions regarding the per-visual attribute performance evaluation. For example, consider the trackers DSST, KCF and SAMF. These are highly similar trackers by design, which is reflected in the trends of per-attribute values in Figure 3.8. Nevertheless, the M_{GLA} cannot distinguish performance with respect to attributes motion change, scale change and occlusion, while the performance difference is clear from M_{PFA}. A Wilcoxon paired test on pairs with varying $K = 15 : 24$ showed that the variance of M_{PFA} is lower than that of M_{GLA} at level $p < 0.01$ and an F-test on dispersion showed a difference at significance $p < 0.05$.

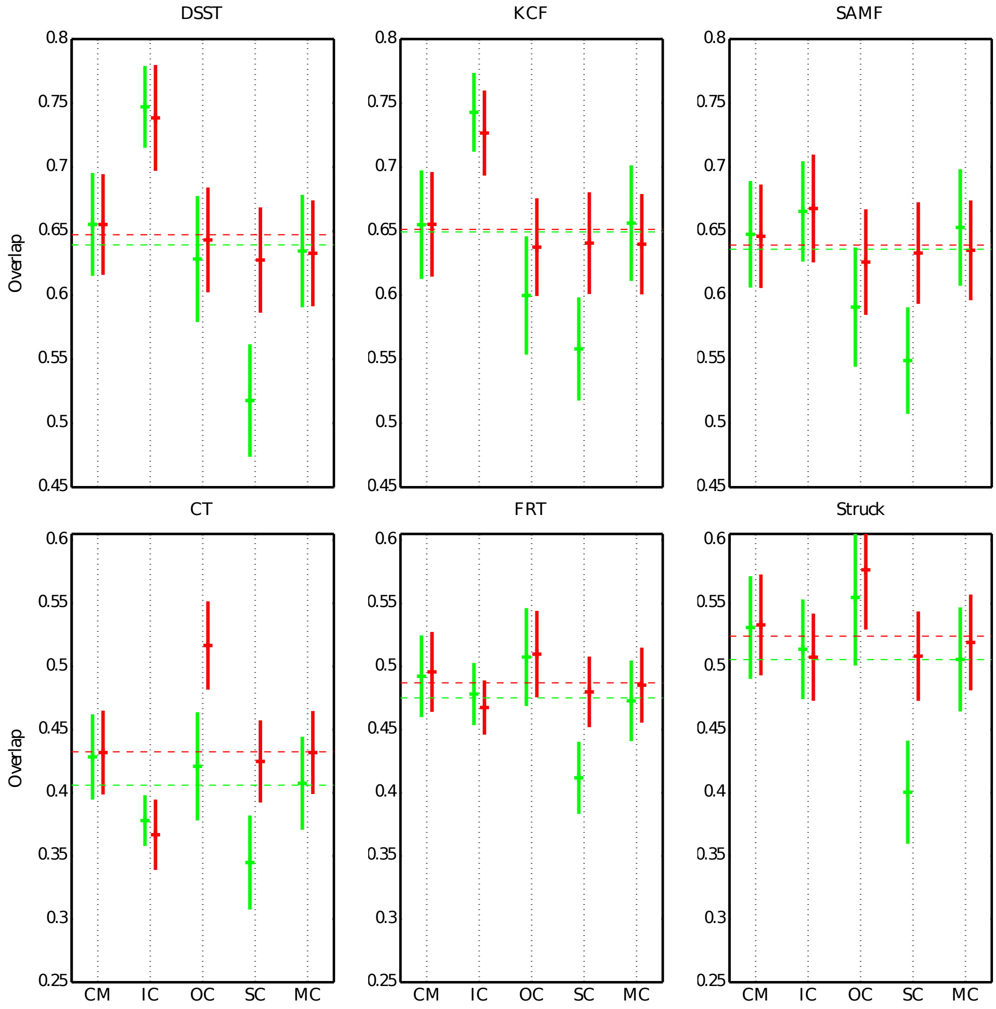


Figure 3.8: The mean and variance of estimators for $K = 24$ sequences that apply per-frame (green) and per-sequence (red) visual attribute annotation. The dashed lines show average performance on the dataset. The abbreviations CM, IC, OC, SC, and MC are used for camera motion, illumination change, occlusion, scale change and motion change, respectively.

3.5 Results of VOT2015 Challenge[5]

This section presents the latest results from the VOT 2015 Challenge that were presented in the workshop on ICCV 2015 conference. In this challenge, a 62 tracking approaches were evaluated, from which 41 were submitted by authors and 21 baseline methods by the VOT committee. The tracking methods represented a diverse set of different approaches such as a deep convolutional neural network (MDNet, DeepSRDCF, SO-DLT), object proposal based (EBT, KCFDP,SPST), part-based (LDP, TRIC-track, G2T, AOG-track, LGT, HoughTrack, MatFlow, CMT, LT-FLO, THANG, FoT, BDF, FCT, FragTrack), generative model-based (ASMS, SumShift, S3Tracker, PKLTF, DFT,

[5] The result plots (AR, an expected average overlap and EFO speed) were created by Luka Čehovin for [KML$^+$15].

IVT, CT, L1APG, DAT), discriminative model-based (OAB, MIL, MCT, CMIL), discriminative regression-based (Struck, RobStruck, SRAT, TGPR, HRP, ACT, KCFv2, DSST, SAMF, SRDCF, PTZ-MOSSE, NSAMF, RAJSSC, OACF, sKCF, LOFT-lite, STC, MKCF+, MTSA-KCF, MvCFT) and combination of multiple trackers (HMM-TxD, MEEM, SCEBT, MUSTer, SME). Please note that some trackers have the same name as their previous version (that have lower performance) such as Struck, which here represents an improved version of the [HST11] method. For more detailed tracker descriptions see [KML^{+}15]. Each tracker had assigned a legend mark that is used in all graphs in this section and is illustrated in Figure 3.9.

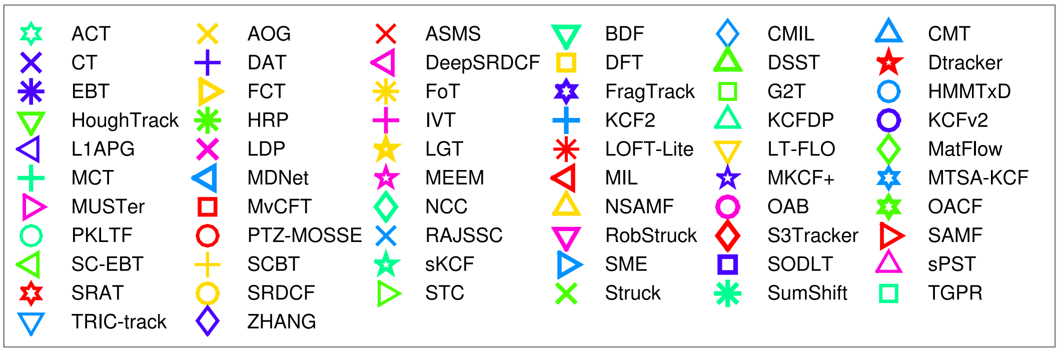


Figure 3.9: Legend (mark symbol with name) of the trackers evaluated in the VOT2015 Challenge. For further details on the method, readers are referred to [KML^{+}15].

The results of the VOT are summarized in sequence pooled and attribute normalized AR rank and AR raw plots in Figure 3.10. The sequence pooling ranking concatenates the result from all sequences and then computes the single ranking, whereas the attribute normalization ranking ranks the trackers for each attribute separately and then averaging the individual ranks.

In both types of the AR plots, there are few methods that clearly outperform the rest in robust or accuracy. The high accurate methods that were significantly more accurate than the other algorithms are MDNet, RAJSSC, DeepSRDCF, SRDCF, sPST, SC-EBT listed in order from the most accurate. The results are more cluttered in the accuracy dimension, which indicates that there is a lot of short-term accurate tracking methods which, however, may be not very robust. An example is OACF, which is a correlation-based tracker and performs exceptionally well in terms of accuracy, but in robustness it has more than twice the number of failures than MDNet. However, focusing the strength of the trackers, the OACF tracker may be useful in application where the accuracy is more important at the expense of the robustness given that OACF is more then two times faster on a CPU then MDNet on a GPU. In terms of robustness, the MDNet, EBT, DeepSRDCF and SRDCF were clearly most robust methods, which is well illustrated in the AR raw plots (right plots in Fig. 3.10) where these methods form a loose cluster that is separated from the rest on the robustness axis.

The expected average overlap was introduced in VOT 2015 Challenge to rank the trackers by one single score. Figure 3.11 shows the expected average overlap curves, with the gray area denoting the used sequence length interval (the top plot), and the respective ranking based on the $\hat{\Phi}_{N_s}$ (the bottom plot). By this score, a winner of the VOT2015 Challenge was the MDNet method, which outperforms the other methods by

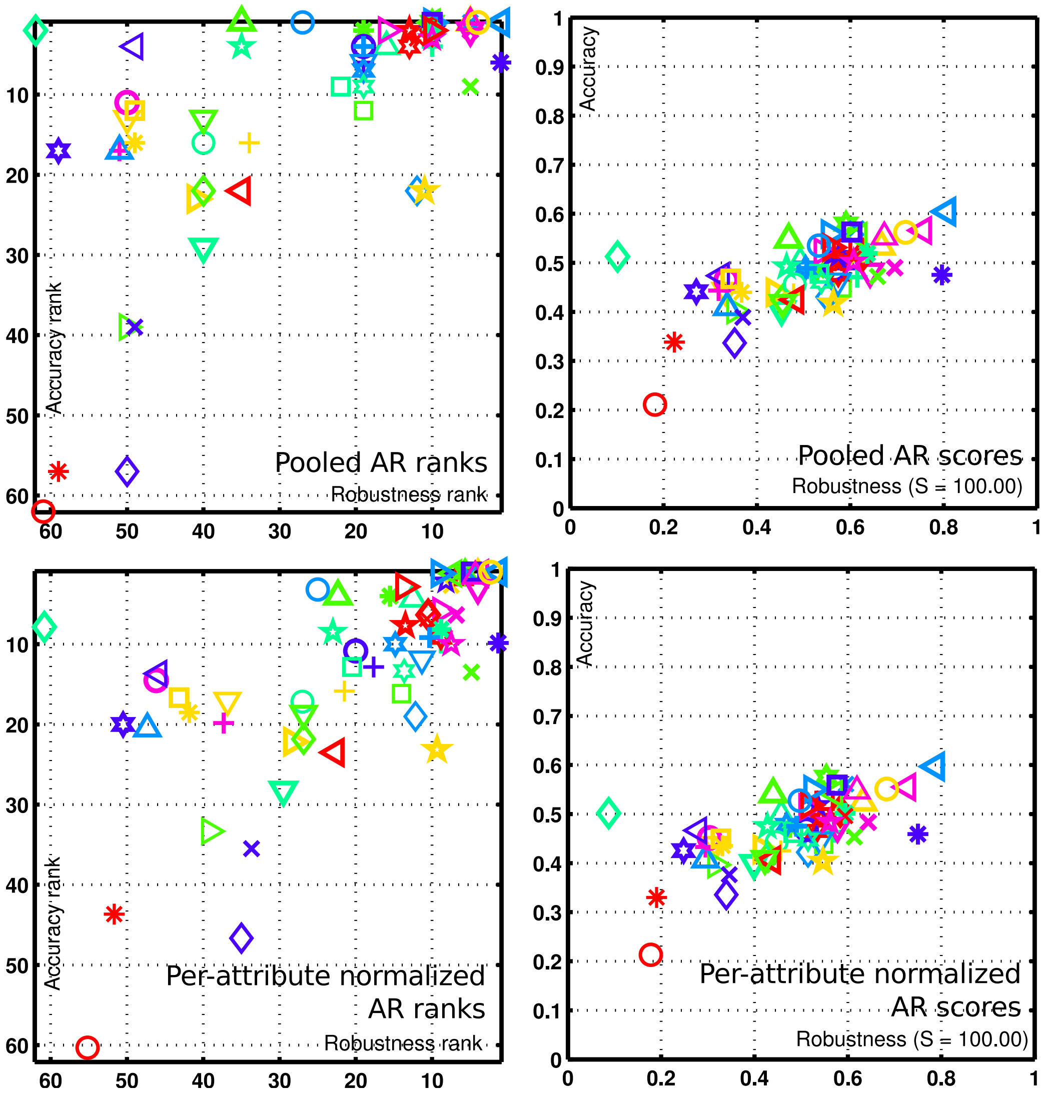


Figure 3.10: The AR rank plots and AR raw plots generated by sequence pooling (upper) and by attribute normalization (below).

a large margin. The bottom plot also highlights the runner-up methods which performs significantly better than the rest. The gray horizontal line denotes the state-of-the-art bound, which is an average performance from the methods published at ICCV, ECCV, CVPR, ICML or BMVC in 2014/2015 (nine papers from 2015 and six from 2014). The trackers used for computation of the bound are marked by gray circles in the plot. This bound shows that more than 40% of the submissions exceed the average performance of the state-of-the-art methods. Furthermore, over 60% of the submissions outperformed the winner of the VOT2014 Challenge.

The other important aspect of tracking algorithms is the speed performance, which may be crucial for some task or hardware specifics. Figure 3.12 shows the performance as the function of the normalized speed measured in the EFO units. The estimated real-time boundary is illustrated by the gray vertical line, which corresponds to the 20 EFO units. There are only a few real-time methods submitted to the VOT, from which the mean-shift tracker ASMS shows the best performance at around 100 EFO units

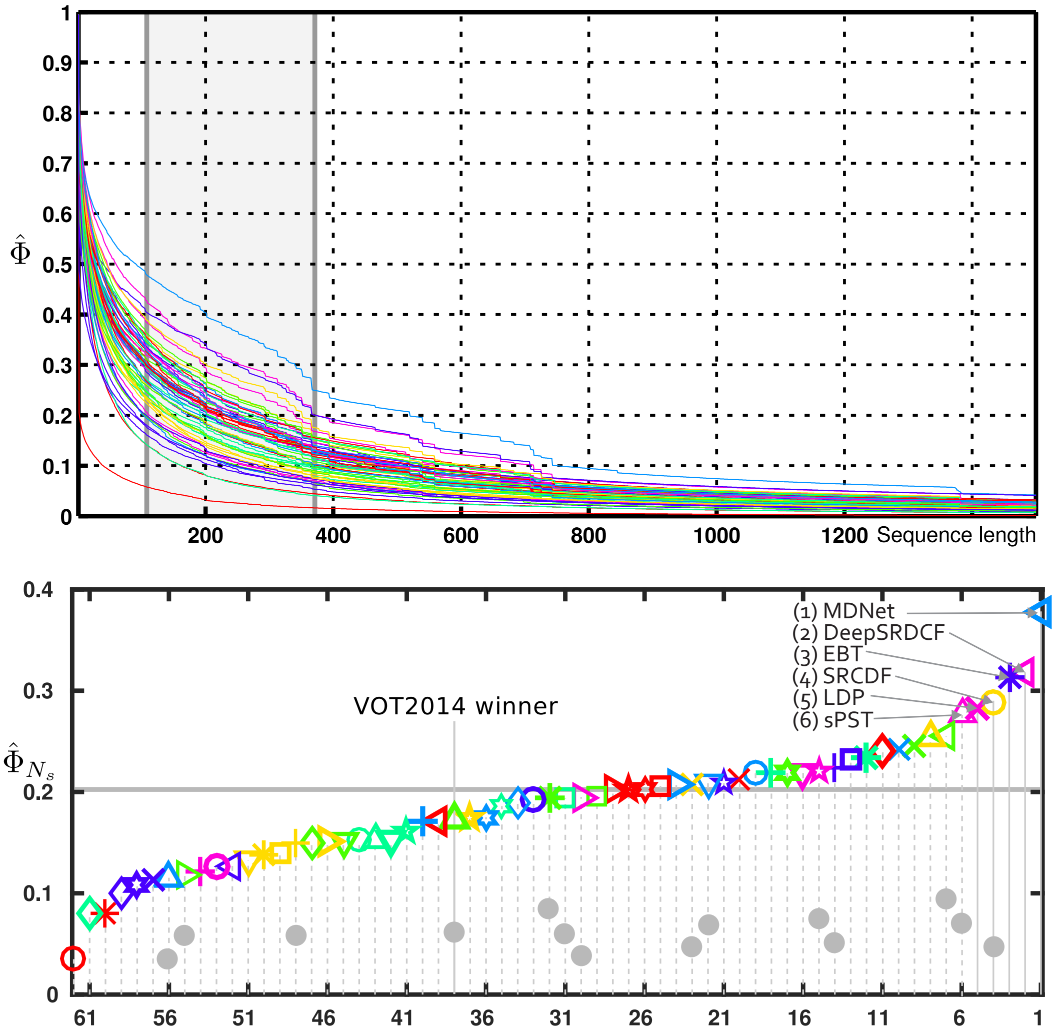


Figure 3.11: The top graph shows the expected average overlap curves. The sequence length interval for the computation of the $\hat{\Phi}_{N_s}$ is denoted by the gray area. The bottom graph show the ranking of the trackers w.r.t. $\hat{\Phi}_{N_s}$ and highlights the names and ranks of the top performing trackers. The horizontal line shows the state-of-the-art bound computed from the 15 methods (marked by the gray circles in the plot) published on major conferences in 2014/1015.

on single CPU with score 0.21 whereas the MDNet runs at 1 EFO on GPU with the score 0.38. Based on this property, the ASMS tracker may find practical use in speed demanding applications such as robots or vision in integrated devices (such as mobile phones).

All the results are summarized in Table 3.5 where the raw accuracy, average number of failures, expected average overlap, tracking speed and implementation details are shown.

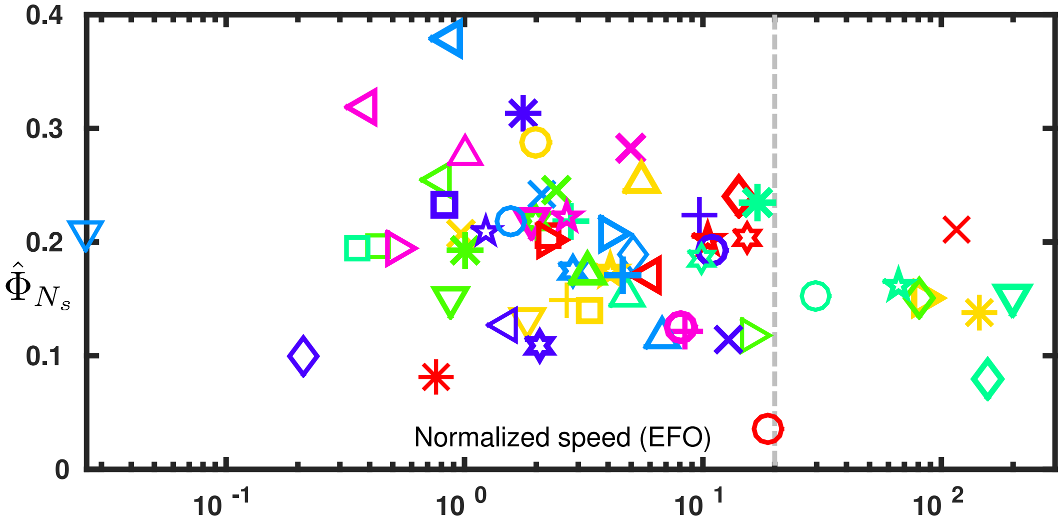


Figure 3.12: Plot of expected overlap score $\hat{\Phi}$ in a relation to EFO speed. The dashed vertical line denotes the estimated real-time performance threshold of 20 EFO units. See Figure 3.9 for legend.

3.6 Conclusions

This Chapter presents the VOT methodology and the VOT2015 challenge results. The methodology measure two weakly correlated performance measures, accuracy and failure rate (i.e. number of tracker failures in the sequence). To measure these scores, the evaluation methodology uses the tracker reinitialization technique, which was theoretically analyzed and compared to other standard benchmarks that do not use reinitialization and its benefits were shown and experimentally validated. The methodology proposes to use per-frame attribute annotation, which allows more detailed and precise analysis of tracker results. The per-frame attribute annotation was theoretically compared to per-sequence annotation, employed by other standard benchmarks, and the ability to display more precise analysis was demonstrated by theory and by experiments. In both proposed novelties (i.e. reinitialization technique and per-frame attribute annotation) the bias and variance of the performance estimators are smaller than in the standardly used per-sequence annotation and no reinitialization (as done in [SCC$^+$13a, WLY13, WLY15]).

The results from the VOT2015 Challenge show that the state-of-the-art is constantly improving and the tracking methods perform significantly better in just a one year span. The current trend mostly exploits the deep neural networks and correlation based filter, however, it is not limited to these techniques and there is a variety of different approaches in top 20 methods (which all perform better than the estimated state-of-the-art bound).

Tracker	A	R	$\hat{\Phi}$	Speed	Impl.	Tracker	A	R	$\hat{\Phi}$	Speed	Impl.
MDNet*	**0.60**	**0.69**	**0.38**	0.87	M C G	**HRP**	0.48	2.39	0.19	1.01	M C
DeepSRDCF*	0.56	1.05	*0.32*	0.38	M C	**KCFv2**	0.48	1.95	0.19	10.90	M
EBT	0.47	*1.02*	0.31	1.76	M C	**CMIL**	0.43	2.47	0.19	5.14	C
SRDCF*	0.56	1.24	0.29	1.99	M C	**ACT***	0.46	2.05	0.19	9.84	M
LDP*	0.51	1.84	0.28	4.36	M C	**MTSA-KCF**	0.49	2.29	0.18	2.83	M
sPST*	0.55	1.48	0.28	1.01	M C	**LGT***	0.42	2.21	0.17	4.12	M C
SC-EBT	0.55	1.86	0.25	0.80	M C	**DSST***	0.54	2.56	0.17	3.29	M C
NSAMF*	0.53	1.29	0.25	5.47	M	**MIL***	0.42	3.11	0.17	5.99	C
Struck*	0.47	1.61	0.25	2.44	C	**KCF2***	0.48	2.17	0.17	4.60	M
RAJSSC	0.57	1.63	0.24	2.12	M	**sKCF**	0.48	2.68	0.16	66.22	C
S3Tracker	0.52	1.77	0.24	14.27	C	**BDF**	0.40	3.11	0.15	**200.24**	C
SumShift	0.52	1.68	0.23	16.78	C	**KCFDP**	0.49	2.34	0.15	4.80	M
SODLT	0.56	1.78	0.23	0.83	M C G	**PKLTF**	0.45	2.72	0.15	29.93	C
DAT	0.49	2.26	0.22	9.61	M	**HoughTrack***	0.42	3.61	0.15	0.87	C
MEEM*	0.50	1.85	0.22	2.70	M	**FCT**	0.43	3.34	0.15	83.37	C
RobStruck	0.48	1.47	0.22	1.89	C	**MatFlow**	0.42	3.12	0.15	81.34	C
OACF	*0.58*	1.81	0.22	2.00	M C	**SCBT**	0.43	2.56	0.15	2.68	C
MCT	0.47	1.76	0.22	2.77	C	**DFT***	0.46	4.32	0.14	3.33	M
HMMTxD*	0.53	2.48	0.22	1.57	C	**FoT***	0.43	4.36	0.14	143.62	C
ASMS*	0.51	1.85	0.21	115.09	C	**LT-FLO**	0.44	4.44	0.13	1.83	M C
MKCF+	0.52	1.83	0.21	1.23	M C	**L1APG***	0.47	4.65	0.13	1.51	M C
TRIC-track	0.46	2.34	0.21	0.03	M C	**OAB***	0.45	4.19	0.13	8.00	C
AOG	0.51	1.67	0.21	0.97	*binary*	**IVT***	0.44	4.33	0.12	8.38	M
SME	0.55	1.98	0.21	4.09	M C	**STC***	0.40	3.75	0.12	16.00	M
MvCFT	0.52	1.72	0.21	2.24	*binary*	**CMT***	0.40	4.09	0.12	6.72	C
SRAT	0.47	2.13	0.20	15.23	M C	**CT***	0.39	4.09	0.11	12.90	M
Dtracker	0.50	2.08	0.20	10.43	C	**FragTrack***	0.43	4.85	0.11	2.08	C
SAMF*	0.53	1.94	0.20	2.25	M	**ZHANG**	0.33	3.59	0.10	0.21	M
G2T	0.45	2.13	0.20	0.43	M C	**LOFT-Lite**	0.34	6.35	0.08	0.75	M
MUSTer	0.52	2.00	0.19	0.52	M C	**NCC***	0.50	11.34	0.08	*154.98*	C
TGPR*	0.48	2.31	0.19	0.35	M C	**PTZ-MOSSE**	0.20	7.27	0.03	18.73	C

Table 3.5: The table shows raw accuracy and the average number of failures, expected average overlap, tracking speed (in EFO) and implementation details (M is Matlab, C is C or C++, G is GPU). Trackers marked with * have been verified by the VOT2015 committee.

Chapter 4

Short-Term Tracking Beyond the Real-Time

A short-term frame-to-frame tracking is the most commonly required task in applications, which employ visual tracking algorithms. Prominent examples of the fastest short-term tracking methods are the Lucas-Kanade [LK81], mean-shift [CRM00] and correlation [BBDL10, HCMB12] based trackers. The popularity of short-term trackers stem from their simplicity and, consequently, high speed and applicability in a wide range of conditions.

A short-term frame-to-frame tracking is formulated as a sequential casual estimation of the pose of an object in the next frame given the pose in the current frame. Short term trackers do not consider the problems of object re-detection after occlusion or disappearance - some pose parameters are always outputted, regardless of the fact that the tracked entity is no more visible in the field of view. This does not exclude tracking-by-detection or correlation methods since their region of interest is usually a small area around the target and does not perform a global detection of the target, therefore relies on sequential detections to keep the track of the target motion.

The following sections presents two new short-term tracking methods whose design is focused on simple, fast and robust tracking. Section 4.1 presents the Flock of Trackers (FoT) approach where the object motion is estimated from the displacements, or, more generally, transformation estimates, of many independent local trackers covering the object. Each local tracker is attached to a particular area specified in the object coordinate frame. The local trackers are not robust and assume that the tracked area is visible in all images and that it all undergoes a simple motion, e.g. translation. The Flock of Trackers object motion estimate is robust if it is obtained by a combination of local tracker motions which is insensitive to failures. This approach allows to estimate more complex transformation (mostly for a rigid object) such as affine transformation robustly while running at hundreds of frames per second, moreover, it is also robust to partial occlusions or deformations by its design.

The scale adaptive mean-shift tracker (ASMS) is proposed in Section 4.2. The ASMS is based on the mean-shift tracker [CRM00] and addresses the general problem of a scale adaptation in the mean-shift tracking formulation, proposes a pixel weighing to exploit the context information (i.e. take into account a background color during mean-shift iteration) and introduces a forward-backward validation and scale regularization for a robust motion and scale estimation. By combining all the above contributions the resulting novel ASMS tracker achieves the state-of-the-art results while running in hundred of frames per second.

4.1 Flock of Tracker

This Section presents a tracking method called the Flock of Trackers (FoT). The name and its design are inspired by the nature [Rey87] and by early methods by Kolsch and Turk [KT04] and Kalal et al. [KMM10b] that exploit a similar design pattern.

The following section describes the method and presents contribution in two main directions. First, methods of placement of local trackers on the object are revisited and the novel idea of ”good points to track are those the tracker drifted to” is proposed. Second, several contributions to the robustness of a pose estimation are proposed: (i) new reliability predictors for the local trackers (displacements) - the Neighbourhood consistency predictor and the Markov predictor, (ii) new rules for combining the predictions and (iii) more robust pose estimator.

The FoT was extensively tested on the larger number of sequences that are standardly used in the literature and outperforms the reference methods. The FoT is also evaluated on the standard VOT benchmarks, where in the VOT2013 [KPL+13] achieves state-of-the-art results.

4.1.1 Introduction

The FoT is a very attractive tracker design. In comparison to many recently published methods, it is relatively simple and transparent and yet its performance is competitive with the state-of-the-art [KPL+13]. Its internal structure allows handling heavy partial occlusion and local non-rigid changes and it makes the pose estimation robust since it does not depend on a single global property of the object but rather on a composition of many local (weak) features.

Recently, Kolsch and Turk [KT04] and Kalal et al. [KMM10b] have shown that a very robust short-term tracker is obtained if a collection (a ”flock”) of local short-term trackers covering the object is run in parallel and the object motion is estimated from the displacements or, more generally, from transformation estimates of the local trackers. Each local tracker is attached to a particular area specified in the object coordinate frame. These two methods use different approaches to placement of local trackers. The Kolsch and Turk method uses the good features to track approach [ST94] where ”suitable” positions are detected and local trackers are placed at these locations. Kalal et al. place the local trackers to the regular grid within the object rectangle. In Section 4.1.2 a new method for handling the local tracker placements is proposed that takes the best of both approaches. As the mentioned methods, we also adopted the Lucas-Kanade [LK81] algorithm for local trackers.

The block structure of the Flock of Trackers is illustrated in Figure 4.1. In its simplest form, the FoT requires only two components: a local short-term tracker, multiple instances of which are run on different areas of the object and provide image-to-image correspondence, and a global object motion estimation module that is robustly combining the local estimates.

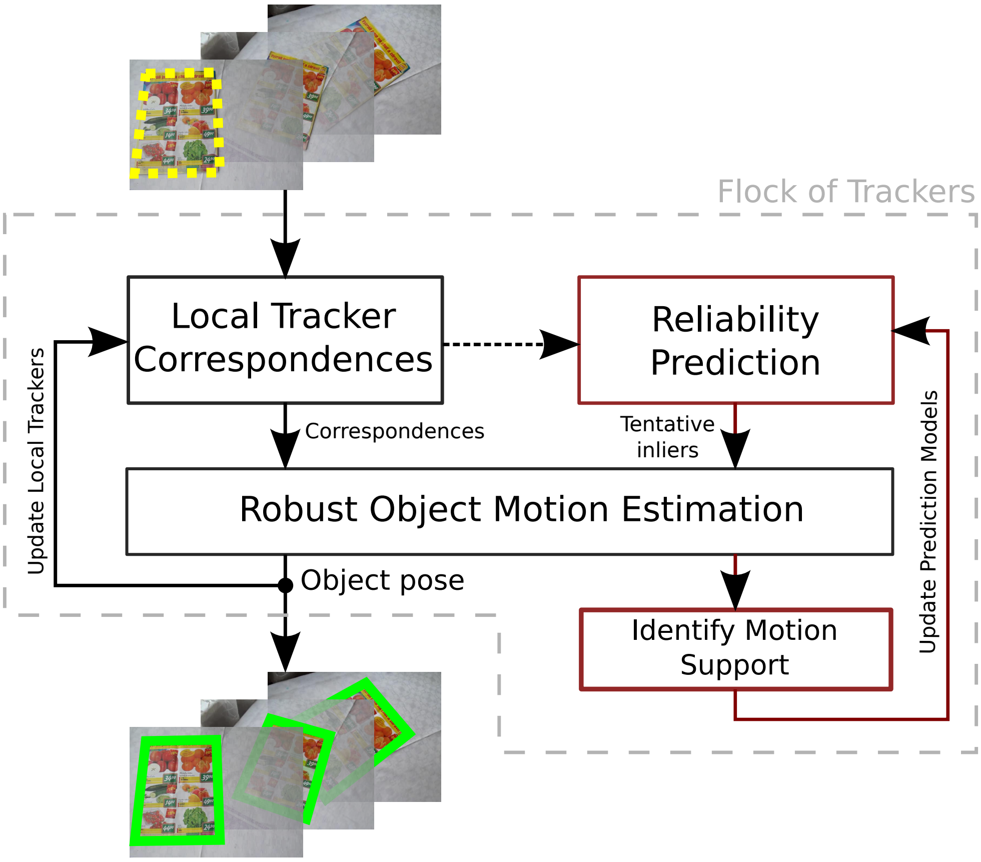


Figure 4.1: Block structure of the Flock of Trackers (FoT). Correspondences (motion estimates) between two images, given the previous object pose and two consecutive images, are produced by local trackers. Simultaneously, reliability is estimated for each motion estimate. The object pose in the next frame is robustly estimated from a subset of most reliable motion estimates called tentative inliers.

The local tracker reliability predictors, presented in Section 4.1.3, fall into two groups. The first group contains methods that are applicable to any short-term tracker and includes estimators based on the apparent magnitude of the intra-frame appearance change like the sum of squared intensity differences (SSD), the normalized cross-correlation (NCC) and the forward-backward procedure (FB). The forward-backward procedure runs the Lucas-Kanade tracker [LK81] twice, once in the forward direction time, as in a standard implementation, and then a second extra run is made in the reverse direction. The probability of being an outlier, i.e. tracker failure, is a function of the distance of the initial position and the position reached by the FB procedure.

The second group of local tracker reliability predictors includes two estimators applicable only to trackers running multiple local trackers, such as the FoT. One, a new predictor based on the consistency of motion estimates in a local neighbourhood (P_N), exploits the fact that it is unlikely for a local motion estimate to be correct if it differs significantly from other motion estimates in its neighbourhood. The second new predictor reflects past performance of the local tracker. If a local tracker motion estimate has often been an outlier in the recent past, i.e. it was inconsistent with the global motion estimate, it is not likely to be correct in the current frame. This occurs for instance when the area covered by the local tracker is occluded or because the area is not suitable for tracking (e.g. it has near constant intensity). This local predictor of tracker reliability is called the Markov predictor (P_M), since it models the sequence of predicted states, either inlier or outlier, as a Markov chain.

The Markov predictor uses the global object motion estimates as ground truth in judging the correctness of local tracker motion. Even though the global motion estimate may be incorrect, this case does not need to be addressed since if the FoT global motion estimation fails the tracker needs to be reinitialized anyway and the Markov predictor model is reset.

Combination of predictors. With the exception of the forward-backward procedure, the evaluation of the reliability prediction is fast in comparison with the time it takes to calculate the local motion estimate. It is therefore reasonable to combine all fast predictors to achieve a high accuracy and avoid, if possible, the FB procedure.

We show that the Markov and Neighbourhood predictors, both on their own and when combined with the normalized cross-correlation predictor P_ρ, are more reliable than the normalized cross-correlation predictor combined with the FB procedure used in the reference method [KMM10b]. The new predictors are computed efficiently at a cost of about 10% of the complete FoT procedure whereas the forward-backward procedure slows down tracking approximately by a factor of two, since the most time consuming part of the process, the Lucas-Kanade local optimization, is run twice. With the proposed combination of reliability predictors, a FoT with much higher robustness to local tracker problems is obtained with a negligible extra computational cost.

Two *predictor combination* schemes are introduced: a predictor combination method approximating a likelihood-based decision, denoted as $\mathcal{P}_\Theta$, and a simple ad-hoc predictor combination denoted as $\mathcal{P}_\wedge$ combination. The ad-hoc combination sets a reliability threshold for each predictor (i.e. P_ρ, P_M, P_N) and the local tracker has to satisfy all of the conditions to be used for pose estimation. The likelihood-based method orders the local trackers based on their likelihood of being correct. It allows choosing either the n best local trackers or a variable size subset that on average maintains a certain level of the inlier ratio for a robust object pose estimation. In experiments, we set the operating point of the $\mathcal{P}_\Theta$ combination so that the number of the local trackers in the predicted inlier set (i.e. points, from which the object pose is estimated) is the same in each frame for the $\mathcal{P}_\wedge$ and the $\mathcal{P}_\Theta$ methods. The methods are evaluated by inlier prediction precision and by how many true inliers were in a predicted set.

Finally, a *robust object motion estimation* that takes as input the local motion estimates equipped with their reliability predictions is examined. The reference method is the Median-Flow (MF) [KMM10b] tracker which was shown to be comparable to the state-of-the-art where an object motion, which is assumed to be well modeled by translation and scaling, is estimated by the median of a subset of local tracker responses. Theoretically, the median with the breakdown point 0.5 is robust up to 50% of corrupted data. Since a single displacement vector gives an estimate of the translation, the median as a translation estimator is robust up to 50% of incorrect local trackers. For a scale estimation a ratio of pairwise distances of local trackers is used as an estimate of scale change, therefore a median is robust up to $100 \times (1 - \sqrt{0.5})\% \doteq 29\%$ of incorrect local trackers for a scale estimation step.

In practice, the outlier tolerance is often lower since the outliers ”conspire”. The outlier motion estimates originate from occluded or background areas. All local motion estimates in such areas are typically consistent with a motion of the occluding object or the background, i.e. they are higher or lower than the tracked object motion and bias the median based estimate. In challenging tracking scenarios presented in Section 4.1.6, the inlier percentage was often not sufficient for the median-based estimation of global motion and it failed when used without local tracker reliability prediction.

Section 4.1.5 shows that the proposed RANSAC [CMK03, FB81] followed by least squares fitting of inliers (LS) as the model estimator is a preferable alternative to the median estimator. There are three key benefits of using the RANSAC+LS estimator: the model is estimated consistently (i.e. a translation estimation is not separated from a scale estimation), the motion model is not constrained to translation, scale and rotation; an affine transformation or a homography requires only to change the sample size and it handles higher outlier percentages.

The rest of this FoT chapter is structured as follows. Section 4.1.2 discussed the local tracker's placement strategies and introduced a novel strategy. Section 4.1.3 proposes two new predictors of local tracker failure and discusses the predictor parameters selection. Section 4.1.4 discusses predictor combinations. Section 4.1.5 introduces RANSAC as a model estimator. Finally, Section 4.1.6 evaluates the proposed improvements. Conclusions are given in Section 4.1.7.

4.1.2 Local tracker placement in FoT

In the FoT, and in similar methods by design, the task of object tracking is usually decomposed into two steps. First, interesting points to track are found (e.g. ”the good features to track” [ST94]). Next, the selected points are tracked. Kalal et al. [KMM10b] omit the first step and choose the points to cover evenly the object of interest - the local trackers are laid out on a regular grid, whereas the Kolsch and Turk [KT04] use the good features to track approach. It is clear that placing local trackers in a regular grid, not all will be put at a location suitable for tracking. The poorly placed local trackers drift away from the original position on the grid. On the other hand, detecting the good features to track does not guarantee that the local trackers evenly cover the object. This may decrease robustness in some cases, e.g. when the part of the object with the majority of good features is occluded or during object deformation, where a particular part of the object may lose its suitable properties for tracking. Moreover, detecting the good features to track is more computationally demanding than just placing the point onto a regular grid. Since we work with the processing speed in terms of milliseconds, the computational burden of localizing the good features may be significant.

In [KMM10b], after estimating the global object motion all local trackers are reset to their original place in the grid. We argue that this is a suboptimal approach because local trackers reinitialized to the same position unsuitable for tracking will drift again. Kolsch et al. [KT04] replenish the ”unreliable” local trackers by running the good features to track detection and choosing new points in a defined area. Instead, we propose the cell structure of placement, where each local tracker is allowed to ”find” a suitable offset from its default position in the grid, see Figure 4.3. The local trackers are forced to stay in their cells, and thus guaranteeing to cover the tracked object evenly, but within the cell, the local trackers are let to assume the best position for tracking.

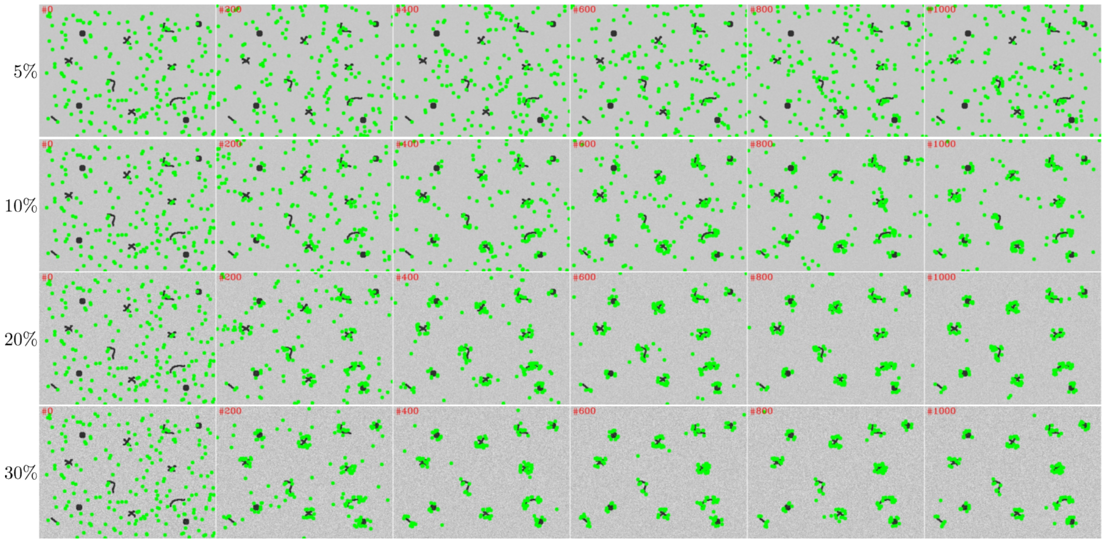


Figure 4.2: Synthetic test of drifting of the 200 KLT trackers. This experiments show that in certain number of iteration (top left red number) the local trackers (green dots) autonomously drift to the corner-like structures (black shapes). The left numbers denote the amount of random uniform noise which is added to the original image in each frame to produce a pair of images for the KLT local trackers.

This can be viewed as a combination of grid placement and good features to track approaches and represents the idea ”good points to track are those the tracker drifted to”. This drifting behavior is demonstrated in the experiment with synthetic data shown in Figure 4.2. The plot shows that the local trackers converge to the corner-like structures given enough iterations. The rate of the convergence is affected by the noise of the image. By increasing the noise ($5, 10, 20, 30\%$ of intensity range, 5% correspond to ± 6 intensity value) the ”exploratory” power of the local trackers increases and they converge much faster. The synthetic image resolution was 480x360 and the parameters for the KLT algorithm were the same as in the FoT. Note that the experiment was demonstrated on much larger images than the usual size of a target bounding box and also it is completely homogeneous except the sparse structure, therefore the number of iterations needed to converge is proportionally larger.

To avoid tracking points that are near each other, which could make the local trackers dependent and likely to fail simultaneously, the cells within which a local tracker must stay may not completely cover the object, as depicted in Figure4.3. However, the preliminary experiments show that slight changes of the cell parameters c_w and c_h from the tight settings (cells share its borders) do not change results significantly, therefore, to keep the method as simple as possible, they are set to the grid resolution, i.e. the local trackers are allowed to assume any position on the object (the cells cover the object). The experimental comparison of grid and cell approach is provided in Section 4.1.6.

4.1.3 Tracker reliability prediction methods

In this section, two novel methods for the local tracker reliability prediction are presented: Section 4.1.3 describes the Neighbourhood consistency reliability predictor and

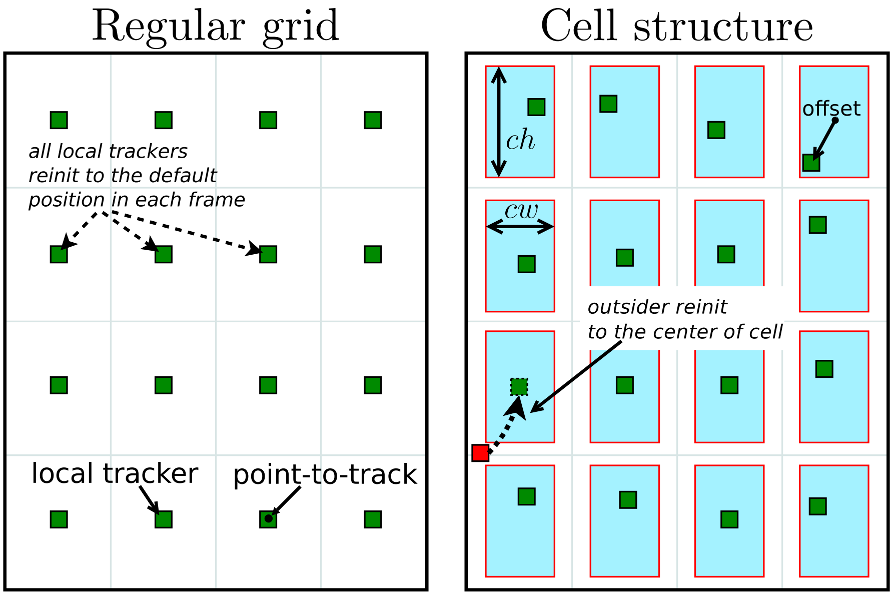


Figure 4.3: A comparison of the regular grid and the cell structure. In the regular grid, in every frame, local trackers are placed on a regular grid. In the cell structure, after the first frame where the local trackers are placed to the regular grid, the location the local tracker drifts to is tracked - the offset with respect to the grid positions is stored. Only local trackers that drifts away form their cells are reset.

Section 4.1.3 presents the Markov predictor based on the long-term behavior of the local tracker. Before that, two predictors used in the literature are described: the reliability predictor P$_\rho$ based on normalized cross-correlation of the corresponding patches in consecutive frames (Section 4.1.3) and the forward-backward predictor (Section 4.1.3

NCC reliability predictor P$_\rho$

The first step of the predictor is to calculate for each local tracker the normalized cross-correlation NCC, eq. 4.1 between the patches T_1 and T_2 at corresponding positions and size (w, h) given by the motion estimate:

$$\begin{aligned} T_1'(x,y) &= T_1(x,y) - 1/(w \cdot h) \cdot \textstyle\sum_{x',y'} T_1(x',y') \\ T_2'(x,y) &= T_2(x,y) - 1/(w \cdot h) \cdot \textstyle\sum_{x',y'} T_2(x',y') \\ \text{NCC} &= \frac{\sum_{x,y}(T_1'(x,y) \cdot T_2'(x,y))}{\sqrt{\sum_{x,y} T_1'(x,y)^2 \cdot \sum_{x,y} T_2'(x,y)^2}} \end{aligned} \tag{4.1}$$

The P$_\rho$ predictor, introduced in [KMM10b] works as a ranking filter. It is hard to find a general function linking the NCC to tracker reliability since NCC values for all local trackers may change dramatically from frame to frame due to e.g. an illumination change, shadows or small drifts. The local trackers are thus only sorted by NCC and their rank is used as a predictor.

The top 50% of the local trackers are predicted to be inliers (correct motion estimate), the rest as outliers (incorrect motion estimate). The threshold was selected empirically. Figure 4.4 shows the histogram of ranks for both inliers and outliers and

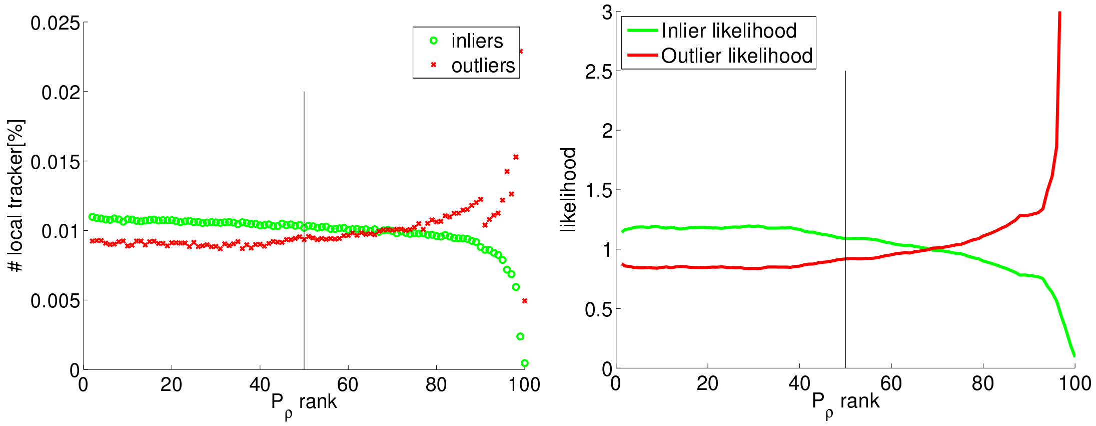


Figure 4.4: Properties of the P_ρ predictor averaged over a subset of the test sequences and all frames. (a) The histogram of NCC ranks ρ for local trackers with correct motion estimates (green) and incorrect motion estimates (red). (b) The correct/incorrect motion estimate ratio as a function of NCC rank ρ (green), the reciprocal value in red.

supports the choice to filter 40%-50% of the worst local trackers, as the probability of being an inlier in the bottom half of the ranks is smaller than the probability of being an outlier. This is illustrated in Figure 4.4 in terms of the likelihood ratio of being an inlier/outlier. Another interesting fact is that probability of being an outlier slightly rises around the 1-5 rank. This is caused by local trackers that are placed on the background (due to the bounding box representation of object or tracker drift) where a zero motion is estimated. The NCC values are very high on the static background.

Experimentally we observed that the P_ρ predictor is sensitive to local tracking precision of the model and candidate patch - small misalignment may induce arbitrarily large similarity difference. This often happens for articulated or non-rigid objects.

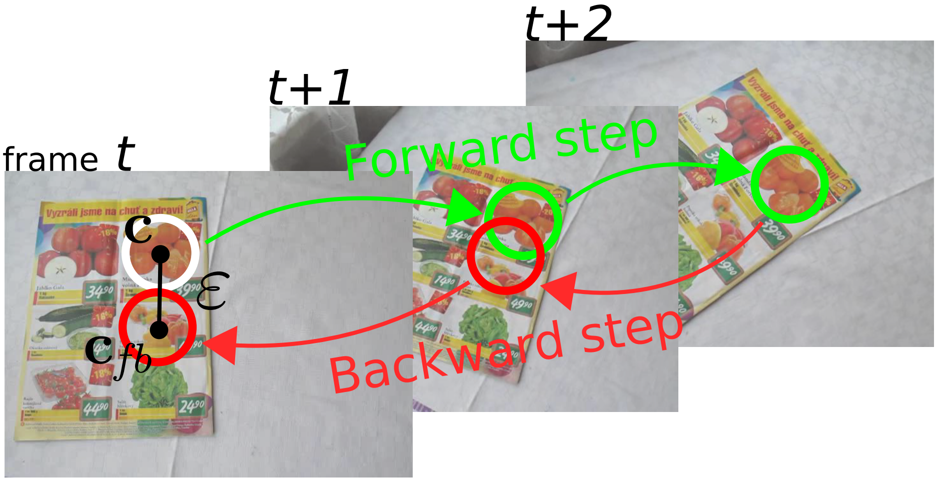


Figure 4.5: A reference point of a regions of interest is tracked forward in time (from frame $t \rightarrow t+1 \rightarrow t+2$) and then backward. The positional forward-backward error $\varepsilon = \| \mathbf{c} - \mathbf{c}_{fb} \|^2$ is then used as a measure of the tracker reliability.

Forward-Backward reliability predictor $\mathrm{P_{FB}}$

The underlying idea behind the forward-backward predictor is that the process of motion estimation between two images is independent of the order of the images. In an error-free situation, tracking an image region using Lucas and Kanade [LK81] gradient optimization from frame $1 \rightarrow 2$ and then the resulting image region from $2 \rightarrow 1$ will end up in the original position in frame 1.

When the deviation from the original position in frame 1 is large, then at least one of the two motion estimates is inaccurate. It is not unreasonable to assume that reliability of the motion estimate is a monotonic function of the distance of the original position and the position reached by the forward-backward procedure. The process may be generalized and the forward and backward direction tracking computed over a larger number of frames. This is illustrated in Figure 4.5.

Figure 4.6 shows the histogram of FB distance ranks for correct and incorrect motion estimates and supports the choice to filter $30\% - 50\%$ of the worst local trackers, as the probability of being an inlier in the bottom half of the ranks is smaller than the probability of being an outlier. Figure 4.6 depicts the ratio of being an inlier or outlier respectively as a function of the rank. Similarly to P_ρ predictor, the probability of being an outlier rises around the 1-5 rank. This is also caused by local trackers that are on the background and thus are consistent with FB procedure.

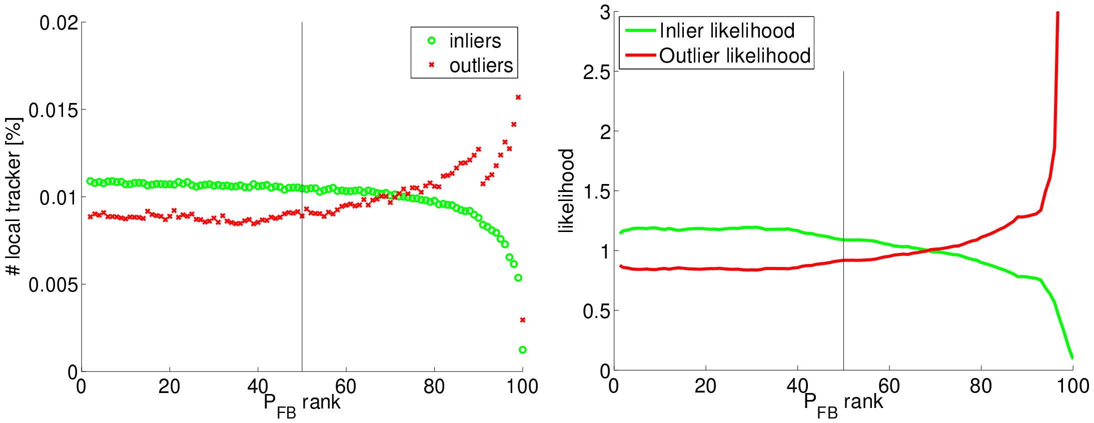


Figure 4.6: Properties of the $\mathrm{P_{FB}}$ predictor averaged over a subset of the test sequences and all frames. (a) The histogram of FB ranks for local trackers with correct motion estimates (green) and incorrect motion estimates (red). (b) The correct/incorrect motion estimate ratio as a function of the FB rank (green), the reciprocal value in red.

Neighbourhood consistency predictor P_N

The assumption behind the neighbourhood consistency predictor is that the motion of neighbouring local trackers is often very similar, whereas a failing local tracker returns a random displacement.

The P_N predictor is implemented as follows. For each local tracker i, a set of neighbouring local trackers N_i is defined. In all experiments, N_i included the four nearest neighbours of i. The neighbourhood consistency score S_i^N, the number of the neigh-

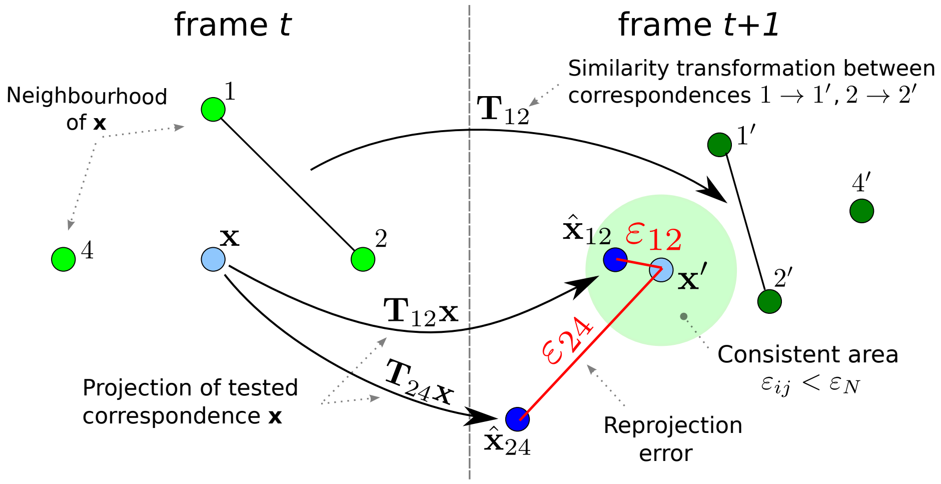

frame t
frame t+1
Similarity transformation between correspondences $1 \rightarrow 1', 2 \rightarrow 2'$
Neighbourhood of **x**
1
2
4
$\mathbf{x}$
$\mathbf{T}_{12}$
$1'$
$4'$
$2'$
$\hat{\mathbf{x}}_{12}$
ε_{12}
$\mathbf{x}'$
$\mathbf{T}_{12}\mathbf{x}$
ε_{24}
$\mathbf{T}_{24}\mathbf{x}$
$\hat{\mathbf{x}}_{24}$
Consistent area $\varepsilon_{ij} < \varepsilon_N$
Projection of tested correspondence **x**
Reprojection error

Figure 4.7: Neighbourhood score computation for two pairs of correspondences. Each unique pair of correspondences (green) $i, j \in 1, 2, 3, 4$ generate a similarity transformation $\mathbf{T}_{ij}$. The tested (blue) correspondence $\mathbf{x}$ is transformed by the estimated similarities and the reprojection error $\varepsilon_{ij} = \| \hat{\mathbf{x}}_{ij} - \mathbf{x}' \|^2$ is computed. The final score is the number of $\varepsilon_{ij} < \varepsilon_N$ (number of $\hat{\mathbf{x}}_{ij}$ points inside green circle around $\mathbf{x}'$).

bourhood local trackers that have a similar displacement. The process is visualized in Figure 4.7.

We proposed and tested two definitions of the scoring functions given in Eq. 4.2 and Eq. 4.3. The first scoring function considers only single point correspondence and tests the points in neighbourhood for max angle and magnitude deviation. It is inspired by the flocking behavior [Rey87], where the direction and velocity of the neighbours in the flock is similar. This score function is easy to compute but suffers problems with zooming scenarios or larger rotations. The second definition works with pairs of correspondences from which a similarity transformation is estimated and a reprojection error for other neighbourhood points is used as a score. By using the similarity transformation it naturally handles the previously mentioned issues; however, the computation is more involved. From the preliminary experiments the latter score function has superior performance and the computational cost of this more complex score function is comparable with the computational cost of the first one, and therefore the latter function was adopted.

$$\mathrm{S'}_i^N = \frac{1}{\mathrm{Z}} \sum_{j \in N_i} \left[|\angle_{ij}| < \varepsilon_\angle \quad \& \quad \frac{\|\Delta_j\|}{\|\Delta_i\|} \in (\varepsilon_l, \varepsilon_h) \right]$$
$$\text{where} \quad [expression] = \begin{cases} 1 & \text{if } \textit{expression} \text{ is true} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

and where $\varepsilon_\angle$ is the maximum angle threshold, $(\varepsilon_l, \varepsilon_h)$ is a bounding range for the ratio of displacement magnitudes, Δ_i is the displacement of the local tracker *i* and $\mathrm{Z} = \frac{4}{N_i}$ is normalization to 4-neighbourhood (to account for corners and sides of bounding box). A local tracker is defined to be consistent if $\mathrm{S}_i^N \geq \theta$, where θ is a threshold for this predictor.

$$\mathrm{S}_i^N = \frac{1}{\mathrm{Z}} \sum_{\substack{j,k \in N_i \\ j \neq k}} [\| T_{jk}\mathbf{x}_i - \mathbf{x'}_i \|^2 < \varepsilon_N] \tag{4.3}$$

$$\text{where} \quad [expression] = \begin{cases} 1 & \text{if } \textit{expression} \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Scoring function S_i^N counts the number of triplets of consistent local trackers. The transformation T_{jk} calculated from motion estimates of trackers j and k is applied on the reference point $\mathbf{x}$ of tracker i. If the transformed position $T_{jk}\mathbf{x}_i$ is within ε_N of its corresponding point $\mathbf{x'}_i$, one is added to the score. In experiments, ε_N was set to 2.

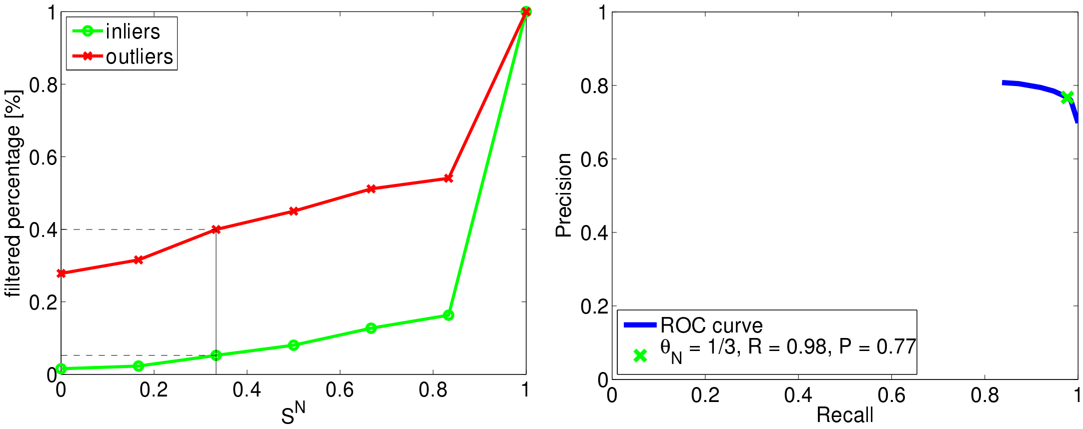


Figure 4.8: Properties of the P_N predictor averaged over a subset of test sequences and all frames, (a) The normalized cumulative histogram of the local tracker state for S^N, (b) The Precision-Recall curve for P_N predictor.

When used as a decision function which is required in one of the predictors combination methods described in the next section, there is a finite number of possible thresholds depending on the number of neighbourhood local trackers.

Figure 4.8 shows a normalized cumulative histogram of the local tracker state for values of S^N normalized to range $< 0, 1 >$. Threshold $\theta_N = 1/6$ is chosen (i.e. S^N greater or equal to $1/3$ to predict an inlier state) as a good trade-off between the ratio of filtered outliers and the false negative rate. Figure 4.8 shows the operating point of this threshold on the Precision-Recall curve by the dashed lines.

Markov reliability predictor $\mathbf{P}_M$

The Markov reliability predictor P_M is based on the model of the past performance of a local tracker bounded to a region specified by an object coordinate frame. The model is in the form of a Markov chain with two states, $s_t \in \{0, 1\}$, depicted in Figure 4.9.

The predicted state, i.e. being correct (inlier) or incorrect (outlier), of the local tracker depends on its state in the previous time instance and on the transition probabilities. The behaviour of each local tracker *i* at time *t* is modeled by transition matrix $\mathbf{T}_t^i$ described in Eq. 4.4, where s_t is the current state of the local tracker, whose columns sum to 1.

$$\mathbf{T}_t^i = \begin{bmatrix} p^i(s_{t+1} = 1 \mid s_t = 1) & p^i(s_{t+1} = 1 \mid s_t = 0) \\ p^i(s_{t+1} = 0 \mid s_t = 1) & p^i(s_{t+1} = 0 \mid s_t = 0) \end{bmatrix} \tag{4.4}$$

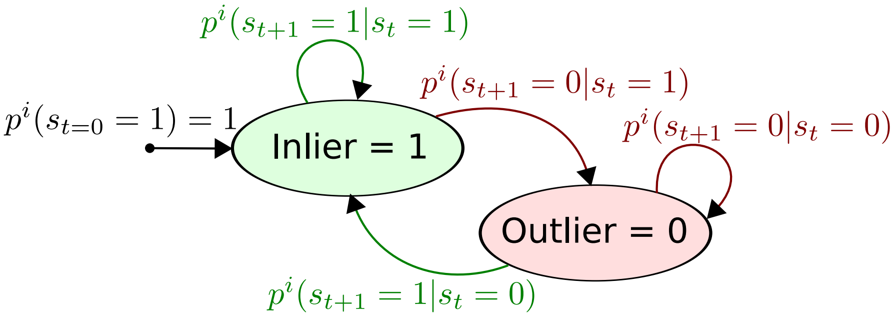

$p^i(s_{t+1} = 1|s_t = 1)$
$p^i(s_{t+1} = 0|s_t = 1)$
$p^i(s_{t=0} = 1) = 1$
Inlier = 1
$p^i(s_{t+1} = 0|s_t = 0)$
Outlier = 0
$p^i(s_{t+1} = 1|s_t = 0)$

Figure 4.9: The state diagram of the Markov chain for the local tracker in the form of a two-state probabilistic automaton with transition probabilities p^i, where *i* identifies the local tracker and the initial state $s_{t=0} = 1$.

The prediction that certain local tracker would be tentative inlier (or an outlier) is done according to Eq. 4.5.

$$\begin{bmatrix} p^i(s_{t+1} = 1) \\ p^i(s_{t+1} = 0) \end{bmatrix} = \mathbf{T}_t^i \cdot \begin{bmatrix} p^i(s_t = 1) \\ p^i(s_t = 0) \end{bmatrix} \tag{4.5}$$

where $p^i(s_t = 1) \in \{0, 1\}$ is binary and depends on the previous state (inlier/outlier) of the i^{th} local tracker. The left side of Eq. 4.5 are probabilities that the next state would be an inlier (outlier).

In the update stage, transition probabilities are re-estimated as follows :

$$\begin{aligned} p^i(s_{t+1} = 1 \mid s_t = 1) &= \frac{n_{11}^i}{n_1^i} \\ p^i(s_{t+1} = 1 \mid s_t = 0) &= \frac{n_{01}^i}{n_0^i} \end{aligned} \tag{4.6}$$

where n_1 and n_0 are numbers for the local tracker *i* being an inlier (outlier respectively), and n_{11} and n_{01} are frequencies for event that the local tracker *i* was an inlier (outlier respectively) in time t and inlier in time $t+1$, for $t \in (0, t\rangle$. The current state of the local tracker being an inlier (outlier) is obtained by identifying local trackers that support the estimated global motion model.

When used as a decision function which is required in one of the predictor combination methods described in the next section, the observed characteristics support the natural choice of thresholding the inlier probability at 0.5. Figure 4.10 depicts the normalized cumulative histograms of a local tracker state for the Markov predictor values quantized to 100 bins. It shows how many inliers/outliers would be filtered out for different values of the θ_M threshold. The selected threshold 0.5 filtered out 4% of inliers and more than 35% of outliers. Figure 4.10 shows the operating point for threshold 0.5 on the Precision-Recall curve.

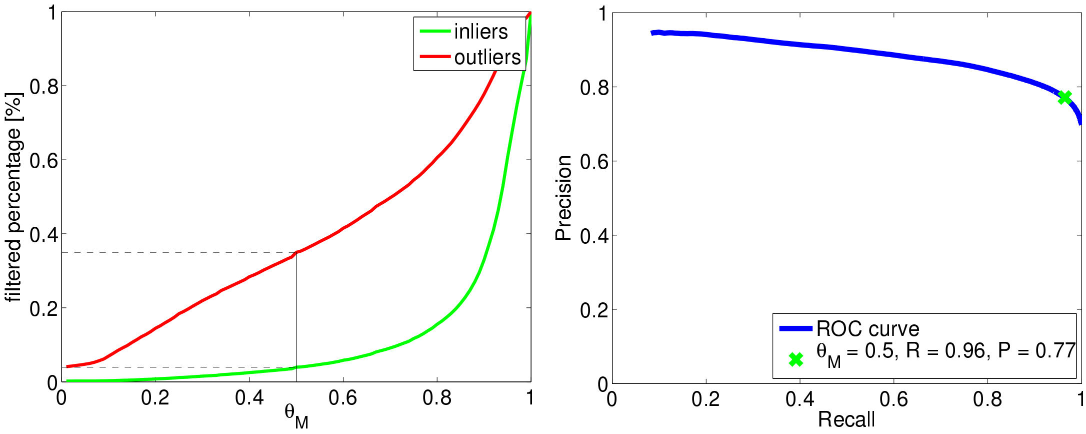


Figure 4.10: Properties of the P_M predictor averaged over a subset of test sequences and all frames, (a) The normalized cumulative histograms of a local tracker state for $p(s_{t+1} = 1)$ values quantized to 100 bins, (b) The Precision-Recall curve for the P_M predictor

4.1.4 Methods for combining tracker reliability predictions.

This section describes two predictor combination methods – $\mathcal{P}_\Theta$ and $\mathcal{P}_\wedge$ and discusses their advantages and disadvantages. The explanation of the combination methods is elaborate for the combination of three predictors P_ρ, P_N, P_M.

$\mathcal{P}_\Theta$ combination method

The $\mathcal{P}_\Theta$ combination estimates the likelihood of a local tracker being an inlier. The local tracker inlier likelihood is a function of three variables (i) P_ρ rank $\in \{1, 2, \dots, 100\}$ quantized equally to 25 bins, $\rho = \lceil \frac{\text{rank}}{25} \rceil$ (ii) The P_N score $\in \{0, 1, 2, 3, 4\}$ in case of four neighbourhood (iii) P_M probability $\in (0, 1)$ quantized equally to 25 bins. In the training phase, an inlier/outlier likelihood ratio is estimated for all the combinations of variables using a Bayesian inference, which results in a table with dimension $25 \times 5 \times 25$. The combination can work in two modes (1) choose the fixed threshold for local trackers inlier/outlier likelihood (2) take the n best local trackers, to form a local trackers subset for object pose estimation.

The advantage of this combination is a possibility to take a quasi-optimal decision (assuming independence of the individual predictors). The problem is formulated as a hypothesis test whether a local tracker is an inlier (outlier) given the likelihood ratio using a standard criterion such as Neyman–Pearson or min-max. The disadvantage is a need for a learning phase to estimate a local tracker inlier likelihood, which may overfit to the training data. In practice, the likelihood estimate generalized well enough to work in various scenarios.

$\mathcal{P}_\wedge$ combination method

The $\mathcal{P}_\wedge$ predictor combination method computes responses of its constituent predictors and makes a binary decision for each of them (reliability below a threshold is interpreted as an outlier and vice versa). The final decision about the local tracker failure is a logical

”and” function:

$$\begin{array}{rcl} f(\mathrm{P}_\rho, \mathrm{P}_N, \mathrm{P}_M) & = & \rho > \mathrm{median}(\rho) \\ & \wedge & \mathrm{S}^N > \theta_N \\ & \wedge & p(s_{t+1} = 1) > \theta_M \end{array} \tag{4.7}$$

The $\mathcal{P}_\wedge$ combination method assumes that since local tracker predictors exploit complementary information (i.e. P_ρ predictor – local appearance, P_M – temporal behaviour, P_N predictor – spatial consistency), parameters and threshold values of the inlier/outlier decision may be set independently.

4.1.5 Pose Estimation

The median estimator is robust and has a breakdown point 0.5. However, as shown in the experimental section, the percentage of correct local motion estimates is lower in many situations. Moreover, the median is biased if the noise is biased, which causes drifting of the tracker. This drifting happens in cases, where the background is static or locally static around the object of interest, e.g. when the bounding box is not a precise representation of the object shape and some local trackers are placed on the background.

We propose to use RANSAC for transformation estimation and show its superiority experimentally. This method has two main advantages over the median: (1) Is more robust to outliers (2) using an unbiased least-square fitting to estimate transformation (up to homography).

4.1.6 Performance evaluation

Test data

The performance of the FoT with combined reliability prediction of local trackers and RANSAC-based object motion estimation was tested on challenging video sequences collected from many recently published papers. The sequences include an object occlusion (or disappearance), illumination changes, fast motion, different object sizes and an object appearance variance. The videos vary in length, contain highly articulated objects and a background clutter; some have poor visual quality. Targets in the sequences exhibit out-of-plane and in-plane rotations and some have homogeneous surfaces almost without texture. For details about the sequences visit `http://cmp.felk.cvut.cz/~vojirtom/dataset`.

Experimental set-up

In all experiments, a frame is considered correctly tracked if its overlap with the ground truth is greater than 0.5, with the exception of experiment 4.1.6 where the influence of the initialization of the tracker was assessed. Since in this case the bounding boxes are randomly generated and may not fully overlap the object, the threshold was lowered to 0.3, see Figure 4.14. The overlap was measured as $o = \frac{area(T \cap G)}{area(T \cup G)}$, where *T* is the object bounding box reported by the tracker and *G* is the ground truth bounding box.

In the experiments, the predictor of neighbourhood consistency (P$_N$) and the Markov predictor (P$_M$) were run as explained in Section 4.1.3. The normalized cross-correlation (P$_\rho$) and the forward-backward procedure rank local trackers and treat the top 50% as inliers. Combinations of two or more predictors use the $\mathcal{P}_\wedge$ approach. Predictors are denoted by the names of their error measure, except for the combination P$_M$ + P$_\rho$ + P$_N$ which is abbreviated to Σ.

Local trackers placement comparison

Two versions of the FoT that differ by local tracker placement — the cell structure and the regular grid — along with the baseline median-flow [KMM10b] that also uses the regular grid were compared on sequences presented in [KMM10b].

The performance was first measured by the number of consecutively correctly tracked frames, i.e. until the first failure (overlap with ground truth less or equal to 0.5). This criterion measures the short-term property of the tracker, which can be interpreted as a robustness of the tracker. It also eliminates the random cases where the object is oscillating around some position and the tracker is periodically on and off the object. According to this criterion, the cell FoT outperforms the grid version as well as the baseline method [KMM10b].

Sequence	FoT-grid	FoT-cell	[KMM10b]
David	453	761	761
Jumping	76	76	36
Pedestrian1	125	140	45
Pedestrian2	153	264	90
Pedestrian3	52	52	52
Car	510	510	510

Table 4.1: A comparison of the grid and cell FoT and the baseline method [KMM10b] in term of the number of consecutively correctly tracked frames, i.e. until the first failure (overlap with groundtruth less or equal to 0.5).

Comparison of $\mathcal{P}_\wedge$ combination vs. $\mathcal{P}_\Theta$ combination

The $\mathcal{P}_\wedge$ predictor combination is compared with the $\mathcal{P}_\Theta$ combination in terms of an inlier prediction precision. To make results comparable, the measurement was done at the operating point of $\mathcal{P}_\wedge$ combination because this method does not guarantee a number of predicted inliers and does not have any means for choosing n-best in contrast to $\mathcal{P}_\Theta$ combination.

The $\mathcal{P}_\Theta$ combination needs to learn likelihoods for the combined likelihood table of three criterion variables. A leave one out cross-validation was used to split the dataset into the training and validation sets. That means that for evaluation on sequence i the table is learned on all sequences except the sequence i. True inliers were extracted by comparing frame-to-frame tracking results with corresponding ground truth positions and criteria variables were recorded. The recorded values (P$_N$ Score, P$_M$ probability, P$_\rho$ rank) were quantized (to 5, 25, 25 bins) and used to compute the inlier - outlier likelihood. Entries of the combined likelihood table are addressed by the quantized criteria values.

Results in Table 4.2 show that the two combination methods perform similarly. The $\mathcal{P}_\wedge$ predictor combination has an advantage that it does not require learning in advance. We choose to use the $\mathcal{P}_\wedge$ predictor combination to keep the tracker as independent as possible of the training data and other external variables (e.g. the precision of the ground truth used for extracting true inliers, the size of the dataset or diversity of dataset).

Seq.	Θ	$\wedge$
gymnastics	0.713±0.132	0.738±0.134
torus	0.875±0.022	0.919±0.021
CarChase	0.894±0.040	0.922±0.043
Motocross	0.857±0.060	0.895±0.058
Panda	0.952±0.029	0.773±0.166
Volkswagen	0.943±0.007	0.965±0.005
car	0.958±0.008	0.977±0.008
david	0.945±0.006	0.966±0.004
jumping	0.680±0.073	0.730±0.068
pedestrian3	0.623±0.053	0.684±0.060
pedestrian4	0.925±0.013	0.945±0.026
pedestrian5	0.967±0.002	0.986±0.001
Sylvestr	0.980±0.006	0.986±0.006
coke	0.924±0.008	0.967±0.006
Mean	0.874±0.033	0.890±0.043

Table 4.2: The comparison of the $\mathcal{P}_\wedge$ predictor combination and the $\mathcal{P}_\Theta$ combination in terms of inlier prediction precision ± variation. Averaged performance over a subset of sequences is reported in the last row. The subset of sequences was selected such that it includes mainly rigid objects; in some sequences also articulated objects (pedestrians) are tracked.

Comparison of the reliability prediction methods

We compared performance of individual predictors and combinations $\mathrm{P}_{FB\circ\rho}$ (reference [KMM10b]), $\mathrm{P}_{N\circ M}$ and P_{Σ}. All parameters for predictors were fixed for all sequences, as described in Section 4.1.4.

The performance was measured by the recall and the number of reinitializations needed to track the whole sequence (reinitializations after object disappearance are not counted). The recall is defined as the ratio of the number of frames where the estimated object rectangle had an overlap with the ground truth rectangle higher than 0.5 and the number of frames where the object is visible. Approximately speaking, recall is the percentage of the frames, with the tracked object visible, where the object was correctly tracked.

The average results are summarized in Tables 4.3a) and b). Both tables have the same structure. The #best line compares the median flow object motion estimator (m, left) and the RANSAC-based estimator (r, right) by counting the number of sequences when median flow outperformed RANSAC (the number before the ":"), where RANSAC dominated (the number after the ":"), the number of "draws" is given in parentheses.

According to both the recall and reinitialization (Table 4.3) criteria, RANSAC performs better for all reliability predictors and their combinations. Results for different predictors and combinations are presented in different columns. The "mean" line of the table compares the mean recall and reinitialization, where RANSAC also performs better.

The "mean" row allows comparison of the reliability predictors, both individually and in combination. The combinations $\mathrm{P}_{N \circ M}$ and P_{Σ} perform the best, clearly better than any individual tracker and slightly better than the forward-backward procedure combined with the NCC. Note that the P_{Σ} and even more $\mathrm{P}_{N \circ M}$ are significantly faster than the FB procedure.

Figure 4.11 visualizes the performance for selected combinations of predictors in a manner facilitating comparison. Two combinations of predictors P_{Σ} and $\mathrm{P}_{N \circ M}$ are clearly the most reliable.

Visualization of predictor performance on selected frames from two challenging sequences are shown in Figs. 4.12 (*motor-bike*) and 4.13 (*woman*). Predictor score is encoded in a "heat map" (red - high score, blue - low score). Green/Red boxes below predictor score encode false positive (red dot with red background), false negative (green dot with red background), true positive (green dot with green background) and true negative (red dot with green background). On the right side of the image, a cut-out shows the outlier (red) and inlier (green) motion estimates. The green-on-black images show the area covered by inlier local trackers.

For the *motor-bike* sequence, it is somewhat surprising that the motion estimates on the biker are small. The biker is tracked by the camera operator and the position of the bike in the image stays roughly the same, the background exhibits fast apparent motion in the opposite direction. The FoT handles a rather large change of appearance of the biker between frames #31 and #77.

The *woman* sequence is more challenging, due to occlusions and changes of appearance due to walking, the number of local trackers providing correct motion estimates is small, as low as 19 out of 90 in frame # 18.

Comparison of the speed of the reliability prediction methods

The FoT tracker is intended for real-time performance and thus the speed of local tracker predictor is important. Speed was measured as an average time needed for frame-to-frame tracking on all available sequences. For results see Table. 4.4. Processing time for I/O operations, including image loading, and other tasks not relevant to tracking were excluded. The P_{Σ} predictor performs 41% faster than $\mathrm{P}_{\mathrm{FB} \circ \rho}$. Most of the additional computation of P_{Σ} over the $\mathrm{P}_{\emptyset}$ lies in computation of normalized cross-correlation. Therefore, the $\mathrm{P}_{N \circ M}$ overhead is negligible compared to reference predictor $\mathrm{P}_{\emptyset}$ (i.e. tracker without any predictor) and is more than two times faster than $\mathrm{P}_{\mathrm{FB} \circ \rho}$.

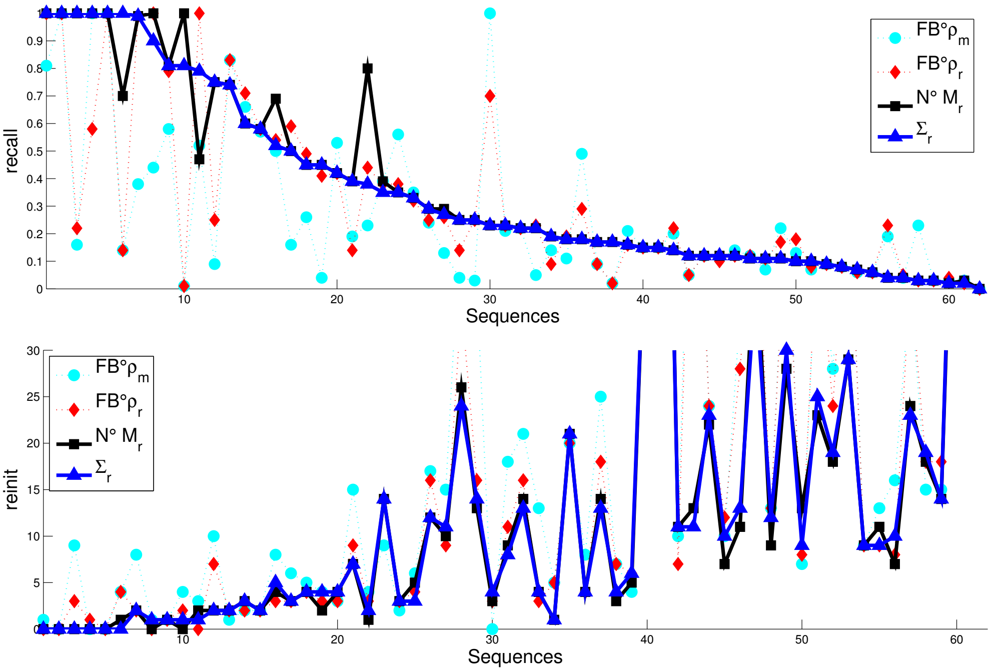


Figure 4.11: Comparison of the best performing predictor combinations and estimators in terms (a) Recall and (b) the number of reinitialization. Sequences (x-axis) are sorted by the recall measure of the P_Σ with RANSAC estimator.

P	∅	ρ	N	FB	M	FB ∘ ρ	$N \circ M$	Σ
	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r
#best	15:36 (11)	18:34 (10)	14:33 (15)	15:28 (19)	19:31 (12)	15:28 (19)	21:30 (11)	21:27 (14)
mean	0.26:0.34	0.26:0.32	0.27:0.35	0.27:0.32	0.30:0.35	0.27:0.32	0.30:**0.36**	0.30:**0.36**

(a) Recall

P	∅	ρ	N	FB	M	FB ∘ ρ	$N \circ M$	Σ
	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r
#best	13:40 (9)	11:36 (15)	19:34 (9)	14:34 (14)	17:37 (8)	14:34 (14)	22:33 (7)	19:35 (8)
mean	20.4:14.9	19.8:14.7	15.8:14.6	18.9:17.1	18.0:13.4	18.9:17.1	15.1:**13.3**	15.1:13.5

(b) Number of reinitialisations

Table 4.3: The tables show the recall and the number of reinitialization for different predictor combinations (top rows) and a pose estimation method (median flow - m, RANSAC - r). The #best line counts sequences where the pose estimation (m or r) outperformed the other and the number of "draws" is given in parentheses. The "mean" line shows the mean values of recall and number of reitinializations.

P / Seq.	∅	ρ	FB	FB ∘ ρ	$N \circ M$	Σ
	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r	m ⋄ r
Time [ms]	1.53 ▶ 1.55	2.44 ▶ 2.87	2.52 ▶ 2.89	3.43 ▶ 3.58	1.58 ▶ 1.72	2.43 ▶ 2.52

Table 4.4: A comparison of the speed of tracking reliability prediction methods. All times are in milliseconds. The values are averaged over all sequences.

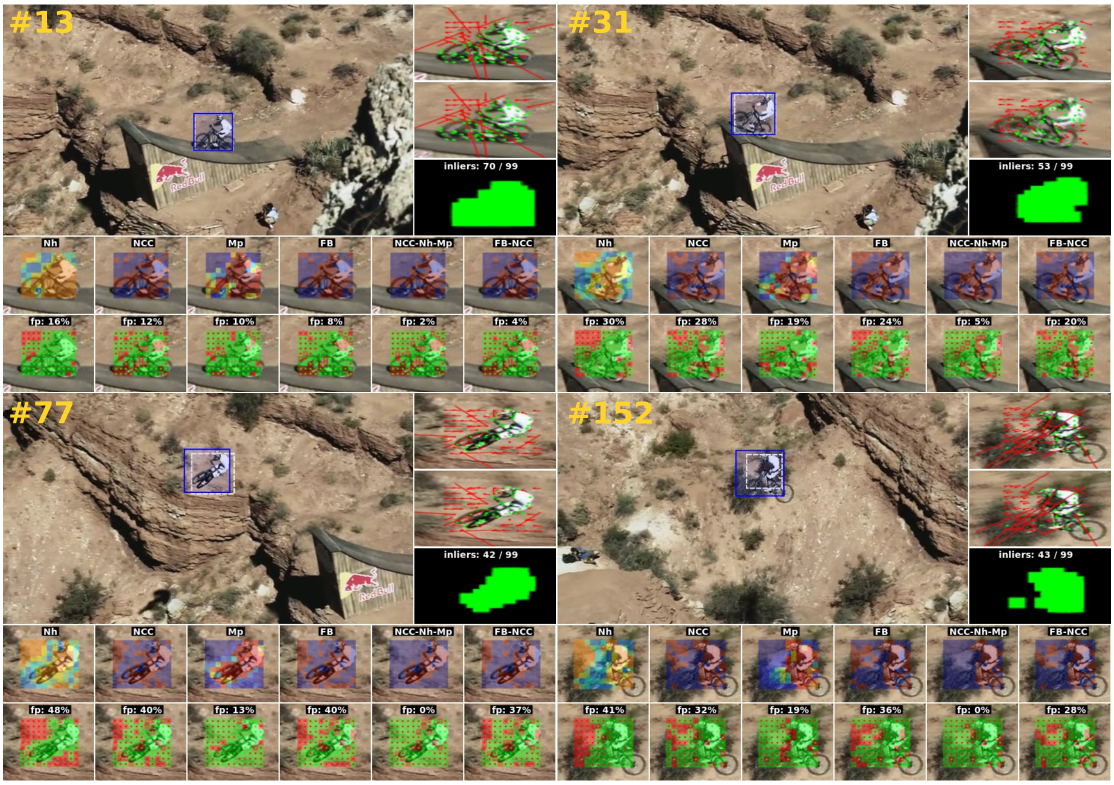


Figure 4.12: Visualization of predictors performance on sequence *mountain-bike*. For details, see text.

Robustness to bounding box initialization

For a tracking algorithm, it is highly desirable not to be sensitive to the initial pose specified by the object bounding box as it is often selected manually, with an unknown precision.

If a part of the bounding box does not cover the object, the P_M predictor soon discovers that the local trackers are consistent with the outlier set. This property can be used to define the object more precisely, e.g. as the set of object parts that are likely to be inliers according to P_M (see Figs. 4.12 and 4.13). Thus, with P_M, the global tracker may be insensitive to the initialization.

This experiment tested the assumption on the challenging sequence Pedestrian 1, where an articulated object is tracked in a sequence containing a background clutter and fast motions, which emphasize the need for correct initialization. We randomly generated 100 initial bounding boxes overlapping the object of interest (Figure 4.14) and counted the correctly tracked frames (Table 4.5).

In the experiment, a frame was declared as correctly tracked if the overlap with the ground truth was greater than 0.3. The tracker with the P_Σ predictor performed about 90% better than the tracker with the $\mathrm{P}_{\mathrm{FB}\circ\rho}$ predictor and it was able to track the object correctly up to frame 84 on average.

Figures 4.15 a) and b) show the histograms of the number of correctly tracked frames for 100 runs with different initialization and Figure 4.15 c) shows the 2D histogram of the number of correctly tracked frames by $\mathrm{P}_{\mathrm{FB}\circ\rho}$ and P_Σ initialized with the

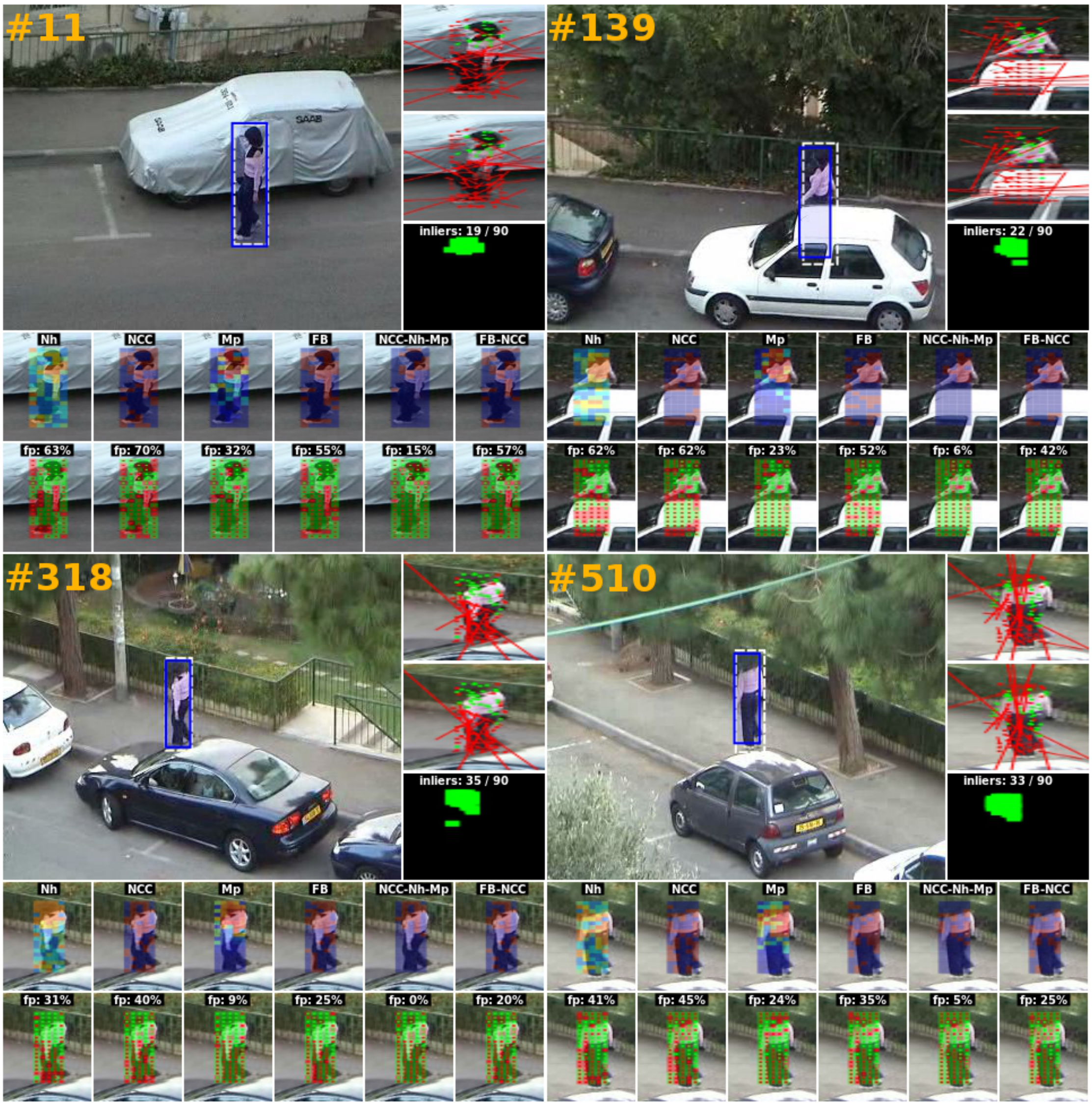


Figure 4.13: Visualization of predictors performance on sequence *woman*. For details, see text.

Method	Score	mean (median)
$\mathrm{P}_{\mathrm{FB}\circ\rho}$ [ref]	4493	45 (21)
P_{Σ}	8438	84.4 (99.5)

Table 4.5: Evaluation of filtering methods in terms of the number of correctly tracked frames with randomly initialized bounding box (see. Figure 4.14). The “score” is the total number of correctly tracked frames, the mean and the median of the same quantity are presented in the right column.

same random bounding box (to compare their performance for an individual random initialization).

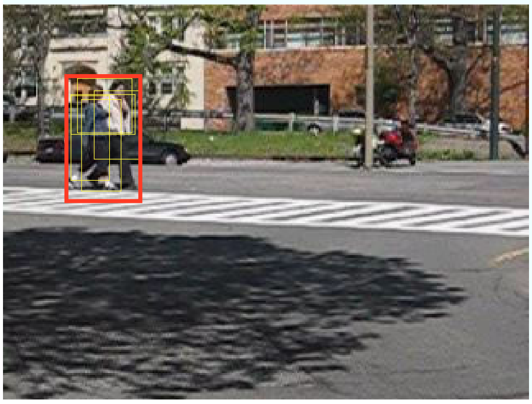
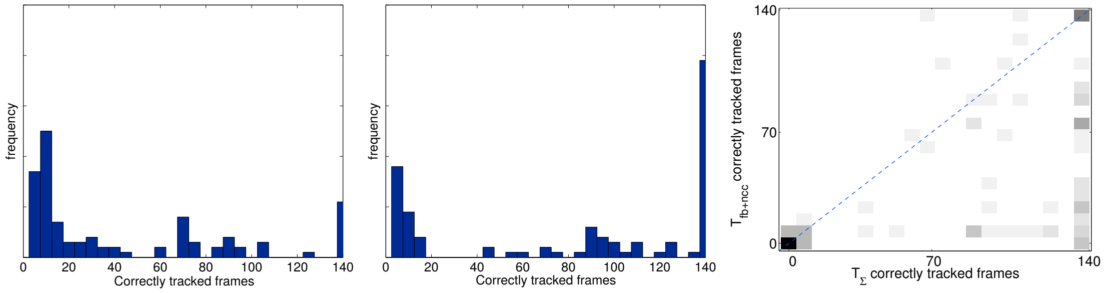


Figure 4.14: Examples of randomly generated initial bounding boxes (yellow) randomly generated within the red rectangle.



(a) (b) (c)

Figure 4.15: Histograms of the number of correctly tracked frames for tracker with (a) $\mathrm{P}_{\mathrm{FB}\circ\rho}$ and (b) P_Σ. (c) The 2D histogram of the number of correctly tracked frames by $\mathrm{P}_{\mathrm{FB}\circ\rho}$ and P_Σ initialized with the same random bounding box for the sequence *Pedestrian3*.

VOT evaluation

The P_Σ combination with RANSAC estimation, i.e. the winning combination of predictors and estimators denoted as FoT, was evaluated in the VOT2013 Challenge (and all following challenges). In the first challenge in 2013, there were three experiments: (i) baseline experiments (same as described in Chapter 3.1), (ii) region noise experiment where the initial bounding box position was perturbed and (iii) same as the baseline but all sequences were converted to grayscale. The results in a form of the rank AR plots are shown in the Figure 4.16. In the overall ranking on VOT2013, the FoT tracker ended up second best and in terms of accuracy the FoT was first in two out of three experiments. In the latest VOT2015 Challenge (results in Section 3.5), the FoT is not very competitive in terms of performance with the most recent tracking algorithms. However, the FoT still has several features that the other methods lack, i.e. (i) is two order of magnitude faster than the majority of the methods and (ii) is able to estimate robustly the object pose up to homography. Thanks to these two features, the FoT is still relevant in a specific application that demands high speed or precise robust estimation of higher order models (such as an affine transformation).

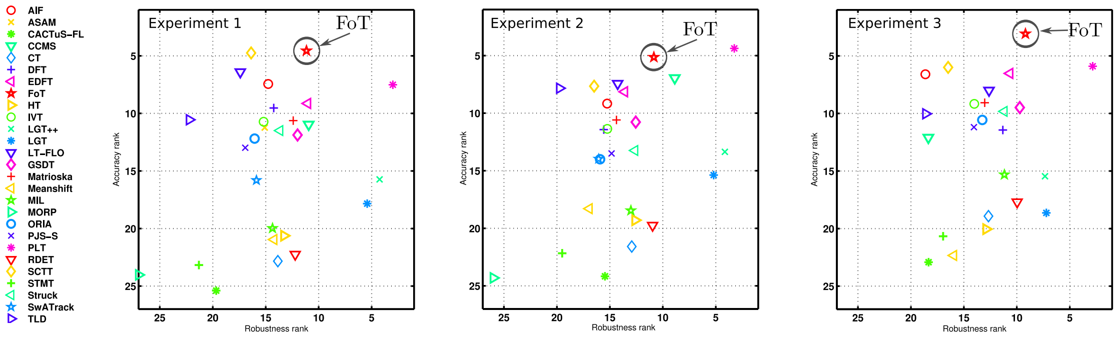


Figure 4.16: The ranking AR plots with respect to three experiments: (1) baseline (2) region noise (3) grayscale, detailed description in the text. Tracker is better if it resides closer to the top-right corner of the plot.

4.1.7 Conclusions

We have presented a set of enhancements of the Flock of Trackers. First, new reliability prediction methods were introduced - the Neighbourhood consistency predictor and the Markov predictor.

Next, two methods for combining predictors, the ad-hoc $\mathcal{P}_\wedge$ and the likelihood thresholding $\mathcal{P}_\Theta$, were proposed and compared and similar performance was achieved. We decided to use $\mathcal{P}_\wedge$, because it is a straightforward approach without a need of learning the relevant statistics in advance.

Combined with the normalized cross-correlation predictor, the new Markov and Neighbourhood consistency predictors form a reliable compound predictor P_Σ. The P_Σ predictor was compared with the published $\mathrm{P}_{\mathrm{FB}\circ\rho}$ predictor and outperformed it in all criteria, i.e. in speed, recall, the number of reinitialization and the robustness to bounding box initialization. The simpler $\mathrm{P}_{N\circ M}$ combination performed almost identically and is faster. Finally, we have shown that the RANSAC-based global object motion estimator outperforms the published median flow algorithm.

The enhanced FoT was extensively tested on a large dataset. Most of the sequences are standard and used in the literature. The improved FoT showed performance superior to the reference method, which competes well with the state-of-the-art [VM11].

For all sequences, the ground truth is available at http://cmp.felk.cvut.cz/~vojirtom/dataset. For some of these sequences, the ground truth has not been in the public domain till now.

4.2 Scale-Adaptive Mean-Shift Tracker

The mean-shift procedure is a popular object tracking algorithm since it is fast, easy to implement and performs well in a range of conditions. This section addresses the problem of scale adaptation in mean-shift tracking and presents a novel theoretically justified scale estimation mechanism which relies solely on the mean-shift procedure for the Hellinger distance.

We also propose two improvements of the mean-shift tracker that make the scale estimation more robust in the presence of a background clutter. The first one is a novel histogram color weighting that exploits the object neighbourhood to help discriminate the target called background ratio weighting (BRW). The BRW improves the performance of MS-like tracking methods in general, which is demonstrated by an experiment with several MS-like methods. The second improvement boosts the performance of the tracker with the proposed scale estimation by the introduction of a forward-backward consistency check and by adopting regularization terms that counter two major problems: a scale expansion caused by background clutter and a scale implosion on self-similar objects.

The proposed mean-shift tracker with the scale selection and BRW is compared with recent state-of-the-art algorithms on a dataset of 77 public sequences. It outperforms the reference algorithms in average recall, processing speed and it achieves the best score for 30% of the sequences - the highest percentage among the reference algorithms. This method is also evaluated on standard VOT benchmark [KML$^+$15], where also achieves state-of-the-art results in performance and also in processing speed.

4.2.1 Introduction

The mean-shift (MS) algorithm by [FH75] is a non-parametric mode-seeking method for density functions. It was introduced to computer vision by [CRM00] who proposed its use for object tracking. The MS algorithm tracks by minimizing the distance between two probability density functions (pdfs) represented by a target and target candidate histograms. Since the histogram distance (or, equivalently, similarity) does not depend on the spatial structure within the search window, the method is suitable for deformable and articulated objects.

The performance of the mean-shift algorithm suffers from the use of a fixed size window if the scale of the target changes. When the projection of the tracked object becomes larger, localization becomes poor since some pixels on the object are not included in the search window and the similarity function often has many local maxima. When the object becomes smaller, the kernel window includes background clutter which often leads to a tracking failure.

The seminal paper by [CRM00] already considered the problem and proposed changing the window size over multiple runs by a constant factor ($\pm 10\%$). The window size maximizing the similarity to the target histogram was chosen. This approach does not cope well with an increase of the object size since the smaller windows usually have higher similarity and therefore the scale is often underestimated.

[Col03] exploited image pyramids and used an additional mean-shift procedure for scale selection after estimating the position. The method works well for objects with a fixed aspect ratio, but this often does not hold for non-rigid or deformable objects. Moreover, the method is significantly slower than the standard MS.

Image moments were used in [Bra98] and [NZZW12b] to determine the scale and orientation of the target. The second moments are computed from an image of weights that are proportional to the probability that a pixel belongs to the target model. [YDD05] introduced a new similarity measure that estimates the scale by comparison of second moments of the target model and the target candidate.

[PP06] assume target rigidity and restrict motion to scaling and translation. The target is first tracked using the mean-shift both in the forward and backward direction to estimate the translation. A scale is then estimated from feature points matched by an M-estimator with outlier rejection. Similarly, [LHJ$^+$07] and [ZKR08] rely on ”support features” for scale estimation after the mean-shift algorithm solves for the position. [LHJ$^+$07] search for the target boundary by correlating the image with four templates. Positions of the boundaries directly determine the scale of the target. [ZKR08] exploit affine structure to recover the target relative scale from feature point correspondences between consecutive frames.

Methods depending on feature matching to estimate the scale robustly, however, they cannot be seamlessly integrated into the mean-shift framework. Moreover, estimating scale from feature correspondences takes times, requires a presence of well-localised features that can be detected with high repeatability and it has difficulties dealing with a non-rigid or deformable object.

We present a theoretically justified scale estimation mechanism which, unlike the method listed above, relies solely on the mean-shift procedure for the Hellinger distance. Furthermore, we propose a formulation for background weighting that exploits the tracked object neighbourhood to help discriminate the object from the background. Additionally, we present two mechanisms that make the scale estimation more robust in a presence of a background clutter and improve tracker performance to a level of the state-of-the-art. The performance is compared to state-of-the-art algorithms on a large tracking dataset and standard benchmark.

4.2.2 Mean-Shift Tracker with Scale Estimation

Standard Kernel-Based Object Tracking

In the standard mean-shift tracking of [CRM00], the target is modelled as an m-bin kernel-estimated histogram in a feature space located at the origin:

$$\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1\dots m} \quad \sum_{u=1}^{m} \hat{q}_u = 1. \tag{4.8}$$

A target candidate at location $\mathbf{y}$ in the subsequent frame is described by its histogram

$$\hat{\mathbf{p}}(\mathbf{y}) = \{\hat{p}_u(\mathbf{y})\}_{u=1\dots m} \quad \sum_{u=1}^{m} \hat{p}_u = 1; \tag{4.9}$$

Let $\mathbf{x}_i$ denote pixel locations, n be the number of pixels of the target and let $\{\mathbf{x}_i^*\}_{i=1\dots n}$ be the pixel locations of the target centered at the origin. Spatially, the target covers a unit circle and an isotropic, convex and monotonically decreasing kernel profile $k(x)$ is used. Function $b : R^2 \to 1 \dots m$ maps the value of the pixel at location $\mathbf{x}_i$ to the index $b(\mathbf{x}_i)$ of the corresponding bin in the feature space. The probability of the feature $u \in \{1, \dots, m\}$ is estimated by the target histogram as follows:

$$\hat{q}_u = C \sum_{i=1}^{n} k\left(\|\mathbf{x}_i^*\|^2\right) \delta[b(\mathbf{x}_i^*) - u], \tag{4.10}$$

where δ is the Kronecker delta and C is a normalization constant so that $\sum_{u=1}^{m} \hat{q}_u = 1$.

Let $\{\mathbf{x}_i\}_{i=1\ldots n_h}$ be pixel locations in the current frame where the target candidate is centered at location $\mathbf{y}$ and n_h be the number of pixels of the target candidate. Using the same kernel profile $k(x)$, but with a scale parameter h , the probability of the feature $u = 1 \ldots m$ in the target candidate is

$$\hat{p}_u(\mathbf{y}) = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right) \delta[b(\mathbf{x}_i) - u], \tag{4.11}$$

where C_h is a normalization constant. The difference between probability distributions $\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1\ldots m}$ and $\{\hat{p}_u(\mathbf{y})\}_{u=1\ldots m}$ is measured by the Hellinger distance of probability measures, which is known to be a metric:

$$H(\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}) = \sqrt{1 - \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}]}\ , \tag{4.12}$$

where

$$\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y})\hat{q}_u} \tag{4.13}$$

is the Bhattacharyya coefficient of $\hat{\mathbf{q}}$ and $\hat{\mathbf{p}}(\mathbf{y})$. Minimizing the Hellinger distance is equivalent to maximizing the Bhattacharyya coefficient $\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}]$. The search for the new target location in the current frame starts at location $\hat{\mathbf{y}}_0$ of the target in the previous frame using gradient ascent with a step size equivalent to the mean-shift method. The kernel is repeatedly moved from the current location $\hat{\mathbf{y}}_0$ to the new location

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h} \mathbf{x}_i w_i g\left(\left\|\frac{(\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{(\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right)}, \tag{4.14}$$

where

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}} \delta[b(\mathbf{x}_i) - u] \tag{4.15}$$

and $g(x) = -k'(x)$ is the derivative of $k(x)$, which is assumed to exist for all $x \geq 0$, except for a finite set of points.

Scale Estimation

Let us assume that the scale changes frame to frame in an isotropic manner[1]. Let $\mathbf{y} = (y^1, y^2)^T, \mathbf{x}_i = (x_i^1, x_i^2)^T$ denote pixel locations and N be the number of pixels in the image. A target is represented by an ellipsoidal region $\frac{(x_i^{*1})^2}{a^2} + \frac{(x_i^{*2})^2}{b^2} < 1$ in the image and an isotropic kernel with profile $k(x)$ as in [CRM00], restricted by a condition $k(x) = 0$ for $x \geq 1$, is used. The probability of the feature $u \in \{1, .., m\}$ is estimated by the target histogram as

$$\hat{q}_u = C \sum_{i=1}^{N} k\left(\frac{(x_i^{*1})^2}{a^2} + \frac{(x_i^{*2})^2}{b^2}\right) \delta[b(\mathbf{x}_i^*) - u], \tag{4.16}$$

[1] Generalization to the anisotropic where $\mathbf{h} = (h^1, h^2)^T$ is straightforward.

where C is a normalization constant. Let $\{\mathbf{x}_i\}_{i=1\ldots N}$ be the pixel locations of the current frame in which the target candidate is centered at location $\mathbf{y}$. Using the same kernel profile $k(x)$, the probability of the feature $u = 1 \ldots m$ in the target candidate is given by

$$\hat{p}_u(\mathbf{y}, h) = C_h \sum_{i=1}^{N} k\left(\frac{(y^1 - x_i^1)^2}{a^2h^2} + \frac{(y^2 - x_i^2)^2}{b^2h^2}\right) \delta[b(\mathbf{x}_i) - u], \tag{4.17}$$

where

$$C_h = \frac{1}{\sum_{i=1}^{N} k\left(\frac{(y^1-x_i^1)^2}{a^2h^2} + \frac{(y^2-x_i^2)^2}{b^2h^2}\right)}. \tag{4.18}$$

The parameter h defines the scale of the target candidate and thus the number of pixels with non-zero values of the kernel function.

For a given kernel and variable h, C_h can be approximated in the following way: Let n_1 be the number of pixels in the ellipsoidal region of the target model, and let n_h be the number of pixels in the ellipsoidal region of the target candidate with a scale h; then $n_h \doteq h^2 n_1$. Using the definition of Riemann integral we obtain:

$$\begin{aligned}
&\frac{1}{C_h}\frac{\pi abh^2}{n_h} = \textstyle\sum_{i=1}^{N} k\left(\frac{(x_i^1)^2}{a^2h^2} + \frac{(x_i^2)^2}{b^2h^2}\right) \frac{\pi abh^2}{n_h} \approx \int\int_{\left\{\frac{(x^1)^2}{a^2h^2}+\frac{(x^2)^2}{b^2h^2}<1\right\}} k\left(\frac{(x^1)^2}{a^2h^2} + \frac{(x^2)^2}{b^2h^2}\right) dx^1dx^2\\
&\frac{1}{C_h}\frac{\pi abh^2}{n_h} \approx h^2ab \int\int_{\|\mathbf{x}\|<1} k(\|\mathbf{x}\|^2)\\
&C_h \approx \frac{\pi}{n_h}\frac{1}{\int\int_{\|\mathbf{x}\|<1} k(\|\mathbf{x}\|^2)} = \frac{\pi}{h^2n_1}\frac{1}{\int\int_{\|\mathbf{x}\|<1} k(\|\mathbf{x}\|^2)}
\end{aligned} \tag{4.19}$$

similarly C can by approximated as $C \approx \frac{\pi}{n_1}\frac{1}{\int\int_{\|\mathbf{x}\|<1} k(\|\mathbf{x}\|^2)}$, therefore $C_h \approx C\frac{1}{h^2}$ and for any two values h_0, h_1 $C_{h_1} \approx C_{h_0}\frac{h_0^2}{h_1^2}$. For justification of the approximation see 4.2.7.

As in [CRM00] the difference between probability distribution $\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1\ldots m}$ and $\{\hat{p}_u(\mathbf{y}, h)\}_{u=1\ldots m}$ is measured by the Hellinger distance. Using the approximations above for C_h in some neighbourhood of h_0 we get

$$\begin{aligned}
&\rho[\hat{\mathbf{p}}(\mathbf{y}, h), \hat{\mathbf{q}}] \approx \hat{\rho}(\mathbf{y}, h) =\\
&= \sum_{u=1}^{m} \sqrt{C_{h_0}\frac{h_0^2}{h^2} \sum_{i=1}^{N} k\left(\frac{(y^1-x_i^1)^2}{a^2h^2} + \frac{(y^2-x_i^2)^2}{b^2h^2}\right) \delta[b(\mathbf{x}_i) - u]\hat{q}_u}
\end{aligned} \tag{4.20}$$

Thus, to minimize the Hellinger distance, function $\hat{\rho}(\mathbf{y}, h)$ is maximized using a gradient method. In the proposed procedure, the kernel with a scale parameter h_0 is iteratively moved from the current location $\hat{\mathbf{y}}_0$ in direction of $\bigtriangledown\hat{\rho}(\hat{y}_0^1, \hat{y}_0^2, h_0)$ to the new location $\hat{\mathbf{y}}_1$, changing its scale to h_1. The basic idea of this procedure is the same as the mean-shift method.

Let us denote

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0, h_0)}} \delta[b(\mathbf{x}_i) - u], \tag{4.21}$$

$$G = \sum_{i=1}^{N} w_i g\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2h_0^2}\right), \tag{4.22}$$

and

$$\mathbf{m}_k(\hat{\mathbf{y}}_0, h_0) = \frac{\sum_{i=1}^{N} \mathbf{x}_i w_i g\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2 h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2 h_0^2}\right)}{G} - \hat{\mathbf{y}}_0, \tag{4.23}$$

where $\mathbf{m}_k(\hat{\mathbf{y}}_0, h_0) = (m_k^1(\hat{\mathbf{y}}_0, h_0), m_k^2(\hat{\mathbf{y}}_0, h_0))^T$. Then we get

$$\frac{\partial \hat{\rho}(\mathbf{y}, h)}{\partial y^1}(\hat{\mathbf{y}}_0, h_0) = \frac{C_{h_0}}{a^2 (h_0)^2} \cdot G \cdot m_k^1(\hat{\mathbf{y}}_0, h_0), \tag{4.24}$$

$$\frac{\partial \hat{\rho}(\mathbf{y}, h)}{\partial y^2}(\hat{\mathbf{y}}_0, h_0) = \frac{C_{h_0}}{b^2 (h_0)^2} \cdot G \cdot m_k^2(\hat{\mathbf{y}}_0, h_0) \tag{4.25}$$

and

$$\begin{aligned} \frac{\partial \hat{\rho}(\mathbf{y},h)}{\partial h}(\hat{\mathbf{y}}_0, h_0) = \frac{C_{h_0}}{(h_0)^2} \cdot G \cdot & \left[\frac{1}{h_0} \frac{\sum_{i=1}^{N} w_i \cdot \left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2}\right) \cdot g\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2 h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2 h_0^2}\right)}{G} \right. \\ & \left. - h_0 \frac{\sum_{i=1}^{N} w_i \cdot k\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2 h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2 h_0^2}\right)}{G} \right]. \end{aligned} \tag{4.26}$$

Finally, the mean-shift update of $\mathbf{y}$ and h is obtained:

$$\hat{y}_1^1 = \frac{1}{a^2} m_k^1(\hat{\mathbf{y}}_0, h_0) + \hat{y}_0^1, \quad \hat{y}_1^2 = \frac{1}{b^2} m_k^2(\hat{\mathbf{y}}_0, h_0) + \hat{y}_0^2 \tag{4.27}$$

$$\begin{aligned} h_1 = & \left[1 - \frac{\sum_{i=1}^{N} w_i \cdot k\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2 h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2 h_0^2}\right)}{G} \right] h_0 \\ & + \frac{1}{h_0} \frac{\sum_{i=1}^{N} w_i \cdot \left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2}\right) \cdot g\left(\frac{(\hat{y}_0^1 - x_i^1)^2}{a^2 h_0^2} + \frac{(\hat{y}_0^2 - x_i^2)^2}{b^2 h_0^2}\right)}{G}. \end{aligned} \tag{4.28}$$

Background Ration Weighting

To incorporate the background information, we can reformulate the maximization of the Bhattacharyya coefficient to coefficient ratio maximization, where the numerator and the denominator are defined as Bhattacharyya coefficients of a target candidate and background respectively. We call this formulation *background ratio weighting* (BRW). Intuitively, this ratio prefers the features (in our case RBG colors), which are more likely to belong to the target and are unlikely in the background. This reformulation ultimately only change the weights of the histogram bins (Eq. 4.30). Background histogram $\hat{\mathbf{bg}}$ is computed over the neighborhood of the target in the first frame (same as $\hat{\mathbf{q}}$, by the same equation only from larger window (larger by half the width of the target window) excluding the pixels in the region of the target) and the ratio is defined as follows:

$$R = \frac{\hat{\rho}[\hat{\mathbf{p}}(\mathbf{y}, h), \hat{\mathbf{q}}]}{\hat{\rho}[\hat{\mathbf{p}}(\mathbf{y}, h), \hat{\mathbf{bg}}]}. \tag{4.29}$$

Using a gradient ascent method for maximization of $log(R)$ we use the following formula with weights w_i changed to weights w_i^{bg}, where

$$\begin{aligned} w_i^{bg} = & \max \left[0, \sum_{u=1}^{m} \left(\frac{1}{\hat{\rho}[\hat{\mathbf{p}}(\hat{\mathbf{y}}_0, h_0), \hat{\mathbf{q}}]} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0, h_0)}} - \right. \right. \\ & \left. \left. - \frac{1}{\hat{\rho}[\hat{\mathbf{p}}(\hat{\mathbf{y}}_0, h_0), \hat{\mathbf{bg}}]} \sqrt{\frac{\hat{bg}_u}{\hat{p}_u(\hat{\mathbf{y}}_0, h_0)}} \right) \delta[b(\mathbf{x}_i) - u] \right]. \end{aligned} \tag{4.30}$$

The $\max$ operator sets the weights w_i^{bg} to be non-negative. In the case of non-negative weights, the mean-shift algorithm preserves its convergence properties. The formulas for the mean-shift update remains the same except for the replaced weights by w_i^{bg}.

4.2.3 Tracking Algorithm

Introducing scale estimation into the mean-shift procedure reveals two issues: Firstly, there is a difference in the MS behavior when the position and scale estimation is imprecise. While errors in position are usually corrected later on during the mean-shift iteration, the error in scale estimation has no ”self-correcting” ability in the presence of a non-trivial background. Secondly, the scale ambiguity of self-similar objects usually leads to an underestimation of the scale and a tracking failure (see Figure 4.17).

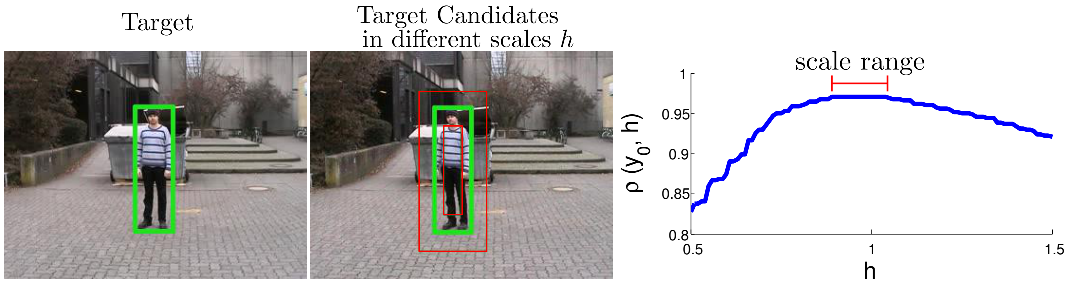


Figure 4.17: Illustration of the scale ambiguity problem. (a) target and target candidates at different scales with fixed center location (green rectangle corresponds to $h = 1$), (b) target candidate similarity with target as a function of the scale parameter measured by Bhattacharyya coefficient.

To cope with this problem and make the tracking more robust, we propose a mean-shift algorithm with regularized scale estimation. The algorithm, denoted MS$_s$, is summarized in Alg. 2.

Algorithm 2: MS$_s$ – mean-shift with regularized scale estimation.

```
Input: Target model $\hat{\mathbf{q}}$, starting position $\mathbf{y}_0$ and starting object size $\mathrm{s}_0$
Output: Position $\mathbf{y}_t$ and scale $h_t$
$iter = 1$;
repeat
    Compute $\hat{p}_u(\mathbf{y}_{t-1}, h_{t-1})$ using Eq. (4.17);
    Compute weights $w_i^{bg}$ according to Eq. (4.30);
    Update position $\mathbf{y}_t$ according to Eq. (4.27), neglecting the constants $a, b$
    assuming that $a \approx b$;
    Update scale $h_t$ according to Eq. (4.28);
    Apply corrections $h_t$ = $h_t$ + Eq. (4.31) + Eq. (4.32);
    $iter = iter + 1$;
until $\| \mathbf{y}_t - \mathbf{y}_{t-1} \|^2 < \varepsilon$ $OR$ $iter > maxIter$;
```

The structure of the algorithm is similar to the standard mean-shift algorithm, except for the scale update step. Two regularization terms are introduced in the scale update step. The first term rs reflects our prior assumption that the target scale does not change dramatically; therefore, the change of scale is penalized according to Eq. (4.31):

$$rs(h) = \begin{cases} -\log(h) & |\log(h)| \leq b_2 \\ b_2 & \log(h) < -b_2 \\ -b_2 & \log(h) > b_2 \end{cases} \tag{4.31}$$

where the h is a scaling factor and the function in absolute value is bounded by the constant b_2. The second term rb addresses the problem of scale ambiguity by forcing the search window to include a portion of background pixels. In other words, from the possible range of scales (generated by the object self-similarity), a slight bias towards the largest is introduced. The rb function is defined by Eq. (4.32):

$$rb(\mathbf{y}, h) = \begin{cases} \varrho - B(\mathbf{y}, h) & |\varrho - B(\mathbf{y}, h)| \leq b_1 \\ -b_1 & \varrho - B(\mathbf{y}, h) < -b_1 \\ b_1 & \varrho - B(\mathbf{y}, h) > b_1 \end{cases} \tag{4.32}$$

where $(\mathbf{y}, h)$ are the position and scaling factor and ϱ define the percentage of weighted background pixels that should be contained in the search window. The function response lies in the interval $(-b_1, b_1)$. The percentage of weighted background pixels is computed as follows:

$$B(\mathbf{y}, h) = \sum_{i=1}^{n} \delta[\hat{q}_{b(\mathbf{x}_i)}] \sum_{u=1}^{m} \hat{p}_u \delta[b(\mathbf{x}_i) - u] \Bigg/ \sum_{i=1}^{n} \sum_{u=1}^{m} \hat{q}_u \delta[b(\mathbf{x}_i) - u] \tag{4.33}$$

where the numerator is the sum of bin weights of the target candidate for pixels in which the target model has $\hat{\mathbf{q}}_u = 0$, and the denominator is the sum of bin weights of the target model over all pixels.

The MS$_s$ algorithm works well for sequences with scale change, but for sequences without scale change or with a significant background clutter, the algorithm tends to estimate non-zero scale, which may lead to accumulation of incorrect scale estimates and a tracking failure. Therefore, we adopted a technique to validate the estimated scale change: *the Backward scale consistency check.* The Backward check uses reverse tracking from position $\mathbf{y}_t$ obtained by forward tracking and validates the estimated scale from step $t - 1$ to t and t to $t - 1$. This validation ensures that in the presence of background clutter the scale estimation does not "grow without bounds" and enables the tracker to recover from erroneous estimates. The algorithm using this technique is summarized in Alg. 3, and we call it Adaptive Scale mean-shift (ASMS).

In the case of a detected scale inconsistency, the object size is computed as a weighted combination of three parts: (i) the previous size; (ii) the new estimated size; (iii) "default" size, which in our case is the initial size of the object. The parameters for this combination were selected experimentally on the subset of testing sequences as a trade-off between scale adaptability of the MS$_s$ and stability of the standard mean-shift algorithm.

We also noticed that mean-shift is more stable if the bandwidth size is biased toward a larger size so that the whole target is included; therefore, the computation of the weight α (Alg. 3) is not symmetric but it prefers enlarging the object size. The default

Algorithm 3: ASMS – mean-shift with scale and backward consistency check.

```
Input: Target model $\hat{\mathrm{q}}$, starting position $\mathbf{y}_0$ and starting object size $\mathrm{s}_0$
Output: Position and scale in each frame $(\mathbf{y}_t, \mathrm{s}_t)$, where $t \in \{1, \ldots, n\}$
foreach Frame $t \in \{1, \ldots, n\}$ do
    $[\mathbf{y}_t, h] = \mathrm{MS}_s(q, \mathrm{image}_t, \mathbf{y}_{t-1}, \mathrm{s}_{t-1})$;
    if $|log(h)| > \Theta_s$ then
        // Scale change - proceed with consistency check
        $[\sim, h_{back}] = \mathrm{MS}_s(q, \mathrm{image}_{t-1}, \mathbf{y}_t, h\mathrm{s}_{t-1})$;
        if $|log(h * h_{back})| > \Theta_c$ then
            // Inconsistent scales
            $\mathrm{s}_t = (1 - \alpha - \beta)\mathrm{s}_{t-1} + \alpha\mathrm{s}_{\text{default}} + \beta h\mathrm{s}_{t-1}$   where   $\alpha = c_1(\frac{\mathrm{s}_{\text{default}}}{\mathrm{s}_{t-1}})$;
    else
        $\mathrm{s}_t = (1 - \gamma)\mathrm{s}_{t-1} + \gamma h\mathrm{s}_{t-1}$;
```

size is kept constant during tracking, and preliminary experiments with size adaptation show no significant benefit and only introduce error caused by incorrect updates. This can be explained by the character of the data, where the target scale usually oscillates around initial value.

4.2.4 Experimental Protocol

Experiments were conducted on 77 sequences[2] collected from the literature. The sequences vary in length from dozens of frames to thousands, contain diverse object types (rigid, articulated) and have different scene settings (indoor/outdoor, static/moving camera, lighting conditions). Object occlusions and objects that disappear from the field of view are also present in the data.

The proposed mean-shift algorithm ASMS is compared with the standard published mean-shift algorithm (MS) and its scale adaptation ($\mathrm{MS}_{\pm}$) proposed by [CRM00]. All algorithms are evaluated with and without the proposed background weighting.

The proposed method is also compared with the state-of-the-art tracking algorithms that are available as source code, namely SOAMST by [NZZW12b] based on the mean-shift algorithm, LGT by [ČKL11], TLD by [KMM10a], CT by [ZZY12] and STRUCK by [HST11]. Parameters for these algorithms were left in default values as set by the authors. Note that our results for those algorithms may differ from results reported in other publications since we did not optimize their parameters for the best performance for each sequence as was done, e.g., by [ZZY12], but were fixed for all experiments. Moreover, the target was initialized in the first frame using the ground truth position for all algorithms. Stochastic methods were run multiple times on each sequence and the average result was reported.

Performance of the algorithms was measured by the recall: the number of correctly tracked frames divided by the number of frames where the target is visible. Recall was chosen because some of the algorithms exhibit detector-like behavior, and therefore, other frequently used criteria, such as first failure frame or failure frame from which the

[2] http://cmp.felk.cvut.cz/~vojirtom/dataset

algorithm does not recover, will not capture the real performance of the algorithm, i.e. in how many frames the algorithm locates the target correctly.

A frame is considered tracked correctly if the overlap with the ground truth is higher than 0.5. The overlap is defined as $o = \frac{area(T \cap G)}{area(T \cup G)}$, where T is an object bounding box reported by the tracker and G is the ground truth bounding box.

The average running time per frame of each algorithm was measured to compare their processing speed. Note that the algorithms are not implemented in the same programming language (SOAMST, LGT, TLD, CT using matlab with MEX files, STRUCT and mean-shift using C++), which may bias the speed measurement towards the more efficient programming languages.

The proposed mean-shift algorithms are written in C++ without heavy optimization or multithreading. All parameters of the algorithm were fixed for all experiments. Some of the parameters are fairly standard (mean-shift termination criterion) and the rest were chosen empirically as follows: bounds for regularization terms $b_1 = 0.05$, $b_2 = 0.1$ and $\varrho = 0.5$; termination of the mean-shift algorithm $\varepsilon = 0.1$, and $maxIter = 15$; scale consistency check $\Theta_s = 0.05 \approx 5\%$ of the scale change, $\Theta_c = 0.1$; exponential averaging $c_1 = 0.1$, $\beta = 0.1$ and $\gamma = 0.3$. The pdf is represented as a histogram computed over the RGB space and quantized into the $16 \times 16 \times 16$ bins.

4.2.5 Results

Background Weighting Evaluation

The experiment evaluates the benefits of different histogram bin weighting based on the background. The proposed BRW method is implemented into a different MS algorithms (i.e. standard MS, the standard scale MS by [CRM00] and the proposed ASMS) and compared to direct histogram weighting (CBWH) proposed by [NZZW12a].

Figure 4.18 shows the recall for 77 sequences. In general, using background weighting improves MS performance. The BRW performs slightly better or equal to CBWH for the standard mean-shift algorithms and dominates for the proposed AMSM. The average recall for the evaluated methods is shown by horizontal dashed lines in the plots. From the experiment, we conclude that ASMS-BRW is superior to other combinations, and therefore, it is used in all subsequent experiments. When not specified otherwise, the abbreviation ASMS refers to ASMS-BRW.

Next, ASMS was compared with the scale adaptation proposed by [CRM00], denoted MS$_\pm$, which runs the MS algorithm three times for different window sizes $(1, 1 \pm 0.1\%)$ and the result with the minimum distance to the target histogram is used. The comparison is included in Figure 4.19 which also shows the results of the state of the art methods. ASMS outperforms MS$_\pm$ for average recall. It performs better on 48 sequences.

Comparison with the State-of-the-Art Methods

Result of the comparison of the ASMS and state-of-the-art algorithms is presented in Figure 4.19, which shows that the performance of the ASMS tracker is comparable to the state-of-the-art methods, and on a large fraction of the sequences (30%) it is the

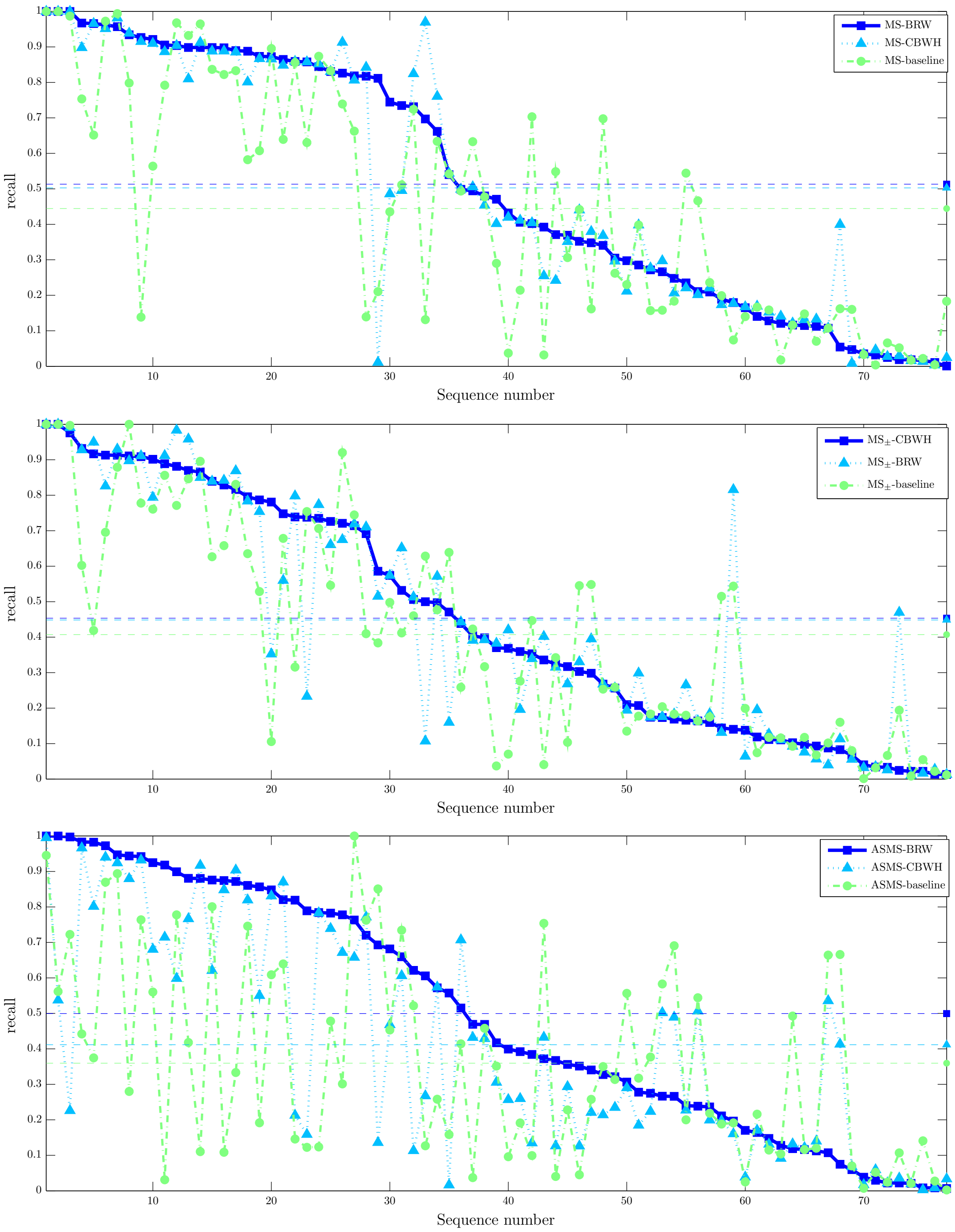


Figure 4.18: Background weighting methods - a comparison of the standard MS, standard scale MS and adaptive scale MS. CBWH denotes the background weighting of [NZZW12a]; the proposed background ratio weighting is denoted BRW. In all plots, sequences (x-axis) are sorted by the recall of the ASMS-BRW. The legend lists the methods in the order of average performance. The dashed lines show average performances.

top performer. However, Figure 4.19 also shows that ASMS performs poorly on some sequences.

The results are summarized in two tables. Results for sequences with at least 30%

object scale difference w.r.t the reference size in at least 20% frames of sequences are presented in Table 4.6. Performance on the remaining ”small scale change” sequences is shown in Table 4.7. The last two rows show the mean performance and the number of sequences where the tracker performed best and second best.

There are few sequences in the set of the 32 sequences with object scale changes where tracking without a re-detection mechanism fails. These “Long-term” sequences with thousands of frames (e.g. *CarChase, Motocross, Panda, Volkswagen*) include object disappearance from the field of view, scene cuts, significant object occlusion and strong background clutter. Some shorter sequences with full object occlusion (e.g. *Vid_F*), cannot be successfully tracked without re-detection too. Since ASMS does not provide any re-detection ability, it can not handle these cases. In these sequences, the TLD tracker achieved the best results.

Method Sequence	MS	MS$_\pm$	ASMS	SOAMST	LGT	TLD	CT	STRUCK
girl	0.70	0.55	0.24	0.14	0.34	0.69	0.13	**0.72**
surfer	0.17	0.14	**0.32**	0.23	0.12	0.16	0.10	0.22
Vid_A_ball	0.86	**1.00**	**1.00**	0.84	0.16	0.44	0.63	0.39
Vid_C_juice	0.49	**0.78**	**0.78**	0.44	0.62	0.44	0.47	0.48
Vid_F_person_fully_occluded	**0.40**	0.26	0.27	0.06	0.27	0.27	0.32	0.35
Vid_I_person_crossing	0.82	0.76	**0.87**	0.06	0.13	0.85	0.19	0.29
Vid_J_person_floor	**0.93**	0.85	0.79	0.08	0.13	0.50	0.44	0.33
Vid_L_coffee	0.24	**0.68**	0.27	0.51	0.22	0.23	0.19	0.23
gymnastics	0.16	0.46	0.24	**0.50**	0.21	0.14	0.14	0.16
hand	0.64	0.53	**0.86**	0.10	0.76	0.26	0.19	0.13
track_running	0.02	0.11	0.33	na	na	**0.41**	0.24	0.25
cliff-dive2	0.15	0.16	0.15	na	0.12	0.08	0.16	**0.18**
motocross1	0.16	0.10	0.13	0.01	**0.17**	0.16	0.08	0.14
mountain-bike	0.80	0.54	0.69	0.00	0.49	0.32	0.34	**0.90**
skiing	0.04	0.04	**0.47**	0.00	0.11	0.07	0.08	0.11
volleyball	0.54	0.50	**0.57**	na	na	**0.57**	0.51	0.48
CarChase	0.07	0.08	0.06	na	0.02	**0.18**	0.00	0.06
Motocross	0.00	0.01	0.04	na	0.00	**0.41**	0.00	0.03
Panda	0.07	0.07	0.17	0.00	0.23	**0.33**	0.20	0.22
Volkswagen	0.00	0.00	0.01	na	0.00	**0.57**	0.01	0.06
pedestrian3	0.11	**0.63**	0.11	0.00	0.02	0.31	0.26	0.47
jump	0.31	0.34	**0.35**	0.01	0.12	0.09	0.09	0.14
animal	0.63	0.18	0.61	0.01	0.17	**0.72**	0.06	0.70
singer1	0.12	0.12	0.12	0.24	0.15	**0.93**	0.29	0.27
singer1(lowfps)	0.26	0.25	**0.36**	na	0.12	0.11	0.14	0.14
skating2	0.83	0.26	**0.94**	0.54	0.30	0.03	0.10	0.35
soccer	0.20	0.20	0.20	0.13	0.16	0.09	0.18	**0.29**
drunk2	0.03	0.03	0.02	0.01	0.17	**0.60**	0.24	0.29
lemming	0.83	0.83	**0.97**	0.85	0.77	0.63	0.27	0.66
dog1	0.18	0.20	0.07	na	na	**0.68**	0.55	0.65
trellis	0.16	0.18	0.37	0.00	**0.60**	0.28	0.16	0.46
coke	0.05	0.07	0.03	0.00	0.32	**0.92**	0.30	0.88
Mean	0.34	0.34	0.39	0.20	0.24	0.39	0.22	0.34
Best+Second (out of 32)	$2+8$	$4+6$	$11+3$	$1+3$	$2+4$	$11+2$	$0+2$	$4+9$

Table 4.6: Recall on sequences with scale change (target was 30% smaller or larger on at least 20% of frames of the sequence). Bold text - the best result for the sequence, underscore - the second best. na indicates that the algorithm fails to process the whole sequence.

ASMS achieved the best score on the *Vid_X* sequences of [KSFC10]. The sequences

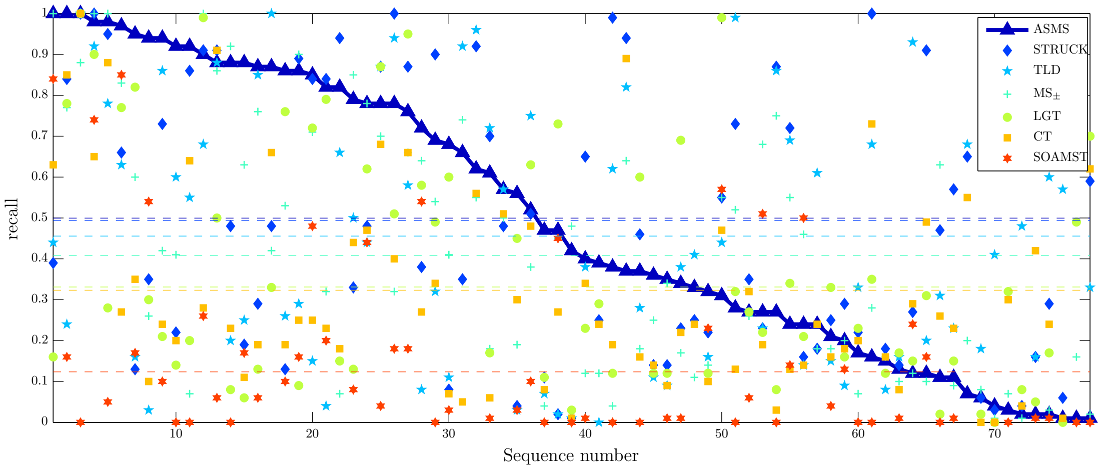


Figure 4.19: ASMS and the state-of-the-art algorithms - comparison of the recall on 77 sequences. Sequences (x-axis) are sorted by the recall measure of the ASMS algorithm. The legend lists the methods in the order of average performance. The dashed lines show average performances.

contain small amounts of background clutter and out-of-plane or in-plane rotation, which is difficult for many state-of-the-art algorithms whose representation of the object is usually spatial dependent and out or in-plane rotation is not explicitly modeled.

The performance of the mean-shift algorithms, in general, drops in the presence of significant background clutter. This issue is more prominent when the tracker estimates more parameters (such as translation and scale) and the estimation errors induce a larger drift (in scale dimension) than in the case of estimating pure translation. This was mainly the case for the *drunk2 and dinosaur* sequences where the color distribution of the target was similar to the background.

Due to RGB color histogram representation, MS algorithms also perform poorly for grayscale sequences (e.g. *track_running, coke, dog1, OccludedFace2, david or shaking*).

Overall, ASMS achieved the best average performance along with the TLD tracker on the sequences with scale and second best performance on the sequences without scale where the STRUCK tracker performs best. ASMS achieved the best score for 30% (which is the highest among other methods) of the sequences and the second best for 13%.

VOT Challenge Results

The proposed ASMS algorithm was also evaluated in the VOT2015 Challenge. The evaluation protocol and results are described in Section 3.5. The ASMS tracker achieved state-of-the-art results (performing above the state-of-the-art bound as estimated by the VOT) and achieves overall rank 20 among the 62 submitted tracking methods. Moreover, the ASMS is the best method in trade off between performance and processing speed, running at more than 100fps on average and the only real-time method from the top 40 methods.

Method Sequence	MS	MS$_\pm$	ASMS	SOAMST	LGT	TLD	CT	STRUCK
OccludedFace2	0.44	0.28	0.16	0.00	0.35	0.68	0.73	**1.00**
Vid_B_cup	**1.00**	**1.00**	0.98	0.74	0.90	0.92	0.65	**1.00**
Vid_D_person	0.97	**1.00**	0.98	0.05	0.28	0.78	0.88	0.95
Vid_E_person_part_occluded	0.90	0.86	0.88	0.06	0.50	0.88	**0.91**	**0.91**
Vid_G_rubikscube	0.63	0.77	**1.00**	0.16	0.78	0.24	0.85	0.84
Vid_H_panda	**1.00**	0.88	**1.00**	0.00	**1.00**	**1.00**	**1.00**	**1.00**
Vid_K_cup	0.87	**1.00**	0.90	0.26	0.99	0.68	0.28	0.91
dinosaur	0.23	0.17	0.34	0.01	**0.69**	0.38	0.22	0.23
hand2	0.63	0.41	**0.68**	0.03	0.60	0.11	0.07	0.08
torus	0.65	0.60	**0.95**	0.17	0.82	0.16	0.35	0.13
head_motion	0.70	0.64	0.37	*na*	*na*	0.82	0.89	**0.94**
shaking_camera	0.44	0.74	0.66	*na*	*na*	**0.92**	0.05	0.35
cliff-dive1	**0.99**	0.66	0.76	0.18	0.95	0.58	0.66	0.87
motocross2	0.74	0.70	0.78	0.04	**0.87**	**0.87**	0.68	**0.87**
car	0.54	0.52	0.28	0.00	0.32	**0.99**	0.13	0.73
david	0.02	0.01	0.02	*na*	0.08	**0.48**	0.04	0.04
jumping	0.51	0.75	0.27	0.00	0.08	0.86	0.03	**0.87**
pedestrian4	0.56	0.41	**0.92**	0.00	0.14	0.60	0.20	0.22
pedestrian5	0.97	0.42	0.87	*na*	0.33	**1.00**	0.66	0.48
diving	0.18	0.18	0.21	0.04	**0.33**	0.15	0.16	0.25
gym	**0.96**	0.90	0.86	0.16	0.09	0.29	0.25	0.89
trans	0.55	0.55	0.31	0.57	**0.99**	0.44	0.47	0.55
basketball	0.48	0.45	0.47	0.45	**0.73**	0.02	0.27	0.02
football	0.16	0.16	0.01	0.00	0.49	0.76	0.73	**0.86**
shaking	0.07	0.05	0.02	0.01	0.05	0.16	**0.42**	0.16
singer2	0.21	0.19	**0.56**	0.03	0.45	0.03	0.30	0.04
skating1	0.16	0.12	0.40	0.01	0.23	0.38	0.34	**0.65**
skating1(lowfps)	0.14	0.09	0.11	0.01	0.15	0.23	0.23	**0.57**
Asada	0.66	0.64	**0.72**	0.54	0.58	0.08	0.27	0.38
dudek-face	0.47	0.18	0.24	*na*	*na*	**0.61**	0.24	0.18
faceocc1	0.79	0.32	0.78	0.18	0.51	0.94	0.40	**1.00**
figure_skating	0.61	0.32	0.82	0.20	0.79	0.04	0.23	**0.84**
woman	0.14	0.07	0.82	*na*	0.15	0.66	0.18	**0.94**
board	0.84	0.71	**0.85**	0.48	0.72	0.15	0.25	0.84
box	0.16	0.10	0.12	0.16	0.31	0.20	0.49	**0.91**
liquor	0.58	0.42	**0.94**	0.10	0.21	0.86	0.24	0.73
Sylvestr	0.72	0.55	0.62	*na*	*na*	**0.96**	0.56	0.92
car11	0.29	0.04	0.38	0.00	0.12	0.62	0.19	**0.99**
person	**0.99**	0.92	0.88	0.00	0.08	0.20	0.23	0.48
tiger1	0.14	0.07	**0.92**	0.00	0.20	0.55	0.64	0.86
tiger2	0.02	0.02	0.01	0.00	**0.70**	0.33	0.62	0.59
bird_1	0.03	0.12	**0.39**	*na*	0.29	0.00	0.24	0.25
bird_2	0.13	0.38	0.52	0.10	0.63	**0.75**	0.48	0.51
bolt	0.21	**0.48**	0.42	0.00	0.03	0.01	0.01	0.01
girl_mov	0.75	0.63	**0.88**	0.17	0.06	0.25	0.11	0.19
Mean	0.52	0.46	0.58	0.13	0.45	0.50	0.40	0.60
Best+Second (out of 45)	$5+8$	$4+4$	$12+7$	$0+1$	$7+7$	$9+9$	$3+6$	$15+6$

Table 4.7: Recall on sequences without a scale change. Bold text - the best result for the sequence, underscore - the second best. *na* indicates that the algorithm fails to process the whole sequence.

Speed

To characterize the speed, the average running time per frame of each algorithm was measured across the whole testing dataset. The forward-backward (FB) validation step has been shown to benefit the ASMS, but it comes at the price of slowing the tracking two times. The experiment in Table 4.8 shows that the slow down factor w.r.t. to standard MS is 2 on average. However, ASMS is still faster then MS$_\pm$ and significantly faster than the state-of-the-art algorithms.

Method	MS	MS$_\pm$	ASMS	SOAMST	LGT	TLD	CT	STRUCK
max	14.4	61	48	6107	864	152	36	112
min	0.4	0.8	0.6	207	107	6	11	43
mean	2.9	7.3	6.1	816	250	51	21	82

Table 4.8: Processing speed in milliseconds. Max (min) are computed as a maximum (minimum) of the average time per sequence; mean is the average time over all sequences.

4.2.6 Conclusions

In this work, a theoretically justified scale estimation for the mean-shift algorithm using Hellinger distance has been proposed. The new scale estimation procedure is regularized, which makes it more robust. Furthermore, we proposed a new formulation of the histogram bin weighting function (BRW) that takes into account background appearance. The formulation is general and can be used in any MS-based algorithm. The increase in performance when using BRW is shown in Figure 4.18.

We introduced the Forward-Backward scheme for automatic decision to accept the newly estimated scale or to use a more robust weighted combination, which is shown to reduce erroneous scale updates. This technique reduces tracking speed twice, however, ASMS is still faster then MS$_\pm$ and outperforms the speed of the state-of-the-art methods by a large margin (see Table 4.8).

The newly proposed ASMS has been compared with the state-of-the-art algorithms on a very large dataset of tracking sequences. It outperforms the reference algorithms in the average recall, processing speed and it achieves the best score for 30% of the sequences (the highest percentage among the reference algorithms) and it is the second best performer for 13% of the sequences.

4.2.7 Appendix - Approximation of C_h

Let us assume we do not use an approximation for C_h. Thus to minimize the Hellinger distance

$$\begin{aligned}&\rho[\hat{\mathbf{p}}(\mathbf{y},h),\hat{\mathbf{q}}] = \\ &= \textstyle\sum_{u=1}^{m}\sqrt{C_h \sum_{i=1}^{N} k\left(\frac{(y^1-x_i^1)^2}{a^2h^2}+\frac{(y^2-x_i^2)^2}{b^2h^2}\right)\delta[b(\mathbf{x}_i)-u]\hat{q}_u}\end{aligned} \tag{4.34}$$

is maximized using a gradient method. The only difference from the derivation using the approximation (Eq. 4.20) is in the partial derivative w.r.t. h:

$$\begin{aligned}&\tfrac{\partial \rho(\mathbf{y},h)}{\partial h}(\hat{\mathbf{y}}_0,h_0) = \\ &= \tfrac{C_{h_0}}{(h_0)^2}\left[\tfrac{1}{h_0}\sum_{i=1}^{N} w_i \cdot \left(\tfrac{(\hat{y}_0^1-x_i^1)^2}{a^2}+\tfrac{(\hat{y}_0^2-x_i^2)^2}{b^2}\right)\cdot g\left(\tfrac{(\hat{y}_0^1-x_i^1)^2}{a^2h_0^2}+\tfrac{(\hat{y}_0^2-x_i^2)^2}{b^2h_0^2}\right)\right. \\ &\left. -h_0\sum_{i=1}^{N} w_i \cdot k\left(\tfrac{(\hat{y}_0^1-x_i^1)^2}{a^2h_0^2}+\tfrac{(\hat{y}_0^2-x_i^2)^2}{b^2h_0^2}\right)\cdot A\right],\end{aligned} \tag{4.35}$$

where

$$A = \frac{\sum_{i=1}^{N}\left(\frac{(\hat{y}_0^1-x_i^1)^2}{a^2}+\frac{(\hat{y}_0^2-x_i^2)^2}{b^2}\right)\cdot g\left(\frac{(\hat{y}_0^1-x_i^1)^2}{a^2h_0^2}+\frac{(\hat{y}_0^2-x_i^2)^2}{b^2h_0^2}\right)}{\sum_{i=1}^{N} k\left(\frac{(\hat{y}_0^1-x_i^1)^2}{a^2h_0^2}+\frac{(\hat{y}_0^2-x_i^2)^2}{b^2h_0^2}\right)}, \tag{4.36}$$

and A tends to 1 for large numbers of pixels in a target candidate. The proposed approximation replaces A by 1 and therefore eliminates the noise caused by A term for small scales of the objects. It is illustrated in Figure 4.20 for a target represented by an ellipsoidal region with $a = 10$ and $b = 10$ (i.e. object size equal to $20x20$px).

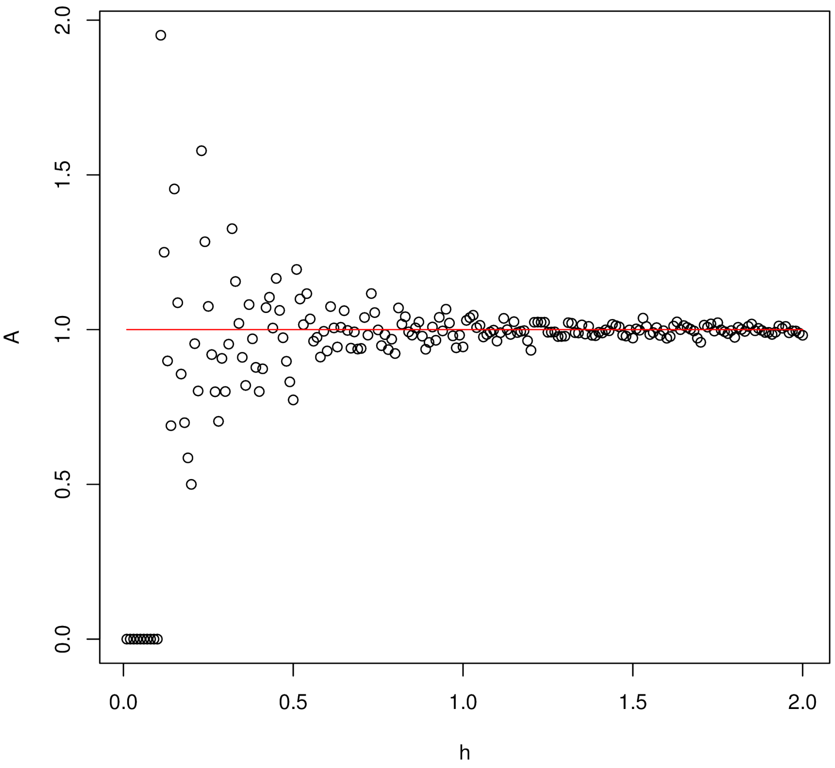


Figure 4.20: Behaviour of A term for a target represented by an ellipsoidal region with Epanechnikov kernel and $a = 10$ and $b = 10$ for a variable scale parameter h.

Chapter 5

Towards Long-Term Tracking by Tracker Fusion

In this chapter, we propose a novel method for visual object tracking called HMMTxD. The method fuses observations from complementary out-of-the-box trackers and a detector by utilizing a hidden Markov model whose latent states correspond to a binary vector expressing the failure of individual trackers. The Markov model is trained in an unsupervised way, relying on an online learned detector to provide a source of tracker-independent information for a modified Baum-Welch algorithm that updates the model w.r.t. the partially annotated data.

We show the effectiveness of the proposed method on a combination of two and three tracking algorithms. The performance of HMMTxD is evaluated on two standard benchmarks (CVPR2013 and VOT) and a rich collection of 77 publicly available sequences. The HMMTxD outperforms the state-of-the-art, often significantly, on all datasets in almost all criteria.

5.1 Introduction

In the last thirty years, a large number of diverse visual tracking methods has been proposed [YJS06, SCC$^+$13a]. The methods differ in the formulation of the problem, assumptions made about the observed motion, in optimization techniques, the features used, in the processing speed, and in the application domain. Some methods focus on specific challenges like tracking of articulated or deformable objects [KL09, GRB11, ČKL13], occlusion handling [GMVGCne], abrupt motion [ZL10] or long-term tracking [PB13, KMM12].

Three observations motivate the presented research. First, most trackers perform poorly if run outside the scenario for which they were designed. Second, some trackers make different and complementary assumptions and their failures are not highly correlated (called complementary trackers in the paper). And finally, even fairly complex well-performing trackers run at the frame rate or faster on standard hardware, opening the possibility for multiple trackers to run concurrently and yet in or near real-time.

We propose a novel methodology that exploits a hidden Markov model (HMM) for fusion of non-uniform observables and pose prediction of multiple complementary trackers using an on-line learned high-precision detector. The non-uniform observables, in this sense, means that each tracker can produce its ”confidence estimate” which may not be directly comparable to each other.

The HMM, trained in an unsupervised manner, estimates the state of the trackers – failed, operates correctly – and outputs the pose of the tracked object taking into account the past performance and observations of the trackers and the detector. The HMM treats the detector output as correct if it is not in contradiction with its current most likely state in which the majority of trackers are correct. This limits the cases where the HMM would be wrongly updated by a false detection. For the potentially many frames where reliable detector output is not available, it combines the trackers. The detector is trained on the first image and interacts with the learning of the HMM by partially annotating the sequence of HMM states in the time of verified detections. The recall of the detector is not critical, but it affects the learning rate of the HMM and the long-term properties of the HMMTxD method, i.e. its ability to reinitialize trackers after occlusions or object disappearance.

Related work. The most closely related approaches include Santner et al. [SLS$^+$10], where three tracking methods with different rates of appearance adaptation are combined to prevent drift due to incorrect model updates. The approach uses simple, hard-coded rules for tracker selection. Kalal et al. [KMM12] combine a tracking-by-detection method with a short-term tracker that generates so-called P-N events to learn new object appearance. The output is defined either by the detector or the tracker based on visual similarity to the learned object model. Both these methods employ pre-defined rules to make decisions about object pose and use one type of measurement, a certain form of similarity between the object and the estimated location. In contrary, HMMTxD learns continuously and causally the performance statistics of individual parts of the systems and fuses multiple ”confidence” measurements in the form of probability densities of observables in the HMM. Zhang et al. [ZMS14] use a pool of multiple classifiers learned from different time spans and chose the one that maximizes an entropy-based cost function. This method addresses the problem of model drifting due to wrong model updates, but the failure modes inherent to the classifier itself remains the same. This is unlike the proposed method which allows combining diverse tracking methods with different inherent failure modes and with various learning strategies to balance their weaknesses.

Similarly to the proposed method, Wang et al. [WY14] and Bailer et at. [BPS14] fuse different out-of-the-box tracking methods. Bailer et al. combine offline the *outputs* of multiple tracking algorithms. There is no interaction between trackers, which for instance implies that the method avoids failure only if one method correctly tracks the whole sequence. Wang et al. use a factorial hidden Markov model and a Bayesian approach. The state space of their factorial HMM is the set of potential object positions, and therefore it is very large. The model contains a probability description of the object motion based on a particle filter. Trackers interact by reinitializing those with low reliability to the pose of the most confident one. The Yuan et al. [YYFL15] using HMM in the same setup, but rather than merging multiple tracking methods, they focus on modeling the temporal change of the target appearance in the HMM framework by introducing an observational dependencies. In contrast, the HMMTxD method is online with tracker interaction via a high precision object detector that supervises tracker reinitializations which happen on the fly. The appearance modeling is performed inside of each tracker and the HMMTxD capture the relation of the confidence provided by tracker and its performance, validated by the object detector, by the observable distributions. Moreover, the HMMTxD confidence estimation is motion-model free and this prevents biases towards support of trackers with a particular motion model.

Yoon et al. [YKY12] combines multiple trackers in a particle filter framework. This approach models observables and transition behavior of individual trackers, but the trackers are self-adapting which makes it prone to wrong model updates. The adaptation of HMMTxD model is supervised by a detector method set to a specific mode of operation – near 100% precision – alleviating the incorrect update problem.

The contributions of this approach are: a novel method for fusion of multiple trackers based on HMMs using non-uniform observables, a simple, and so far unused, unsupervised method for HMMs training in the context of tracking, a tunable feature-based detector with very low false positive rate, and the creation of a tracking system that shows state-of-the-art performance.

5.2 Fusing Multiple Trackers

HMMTxD uses a hidden Markov model (HMM) to integrate pose and observational confidence of different trackers and a detector and updates its confidence estimates that in turn define the pose that it outputs. In the HMM, each tracker is modeled as working correctly (1) or incorrectly (0). The HMM poses no constraints on the definition of tracker correctness, and we adopted target overlap above a threshold. Having at our disposal $\mathbf{n}$ trackers, the set of all possible states is $\{s_1, s_2, \ldots, s_N\} = \{0,1\}^{\mathbf{n}}, N = 2^{\mathbf{n}}$ and the initial state $s_1 = (1, 1, \ldots, 1)$. Note that the trackers are not assumed to be independent, because an independence of tracker correctness is not a realistic assumption. For example, if the tracking problem is relatively easy, all trackers tend to be correct and in the case of occlusion all tend to be incorrect (see the analysis in [KML$^+$16]). The number of states $2^{\mathbf{n}}$ grows exponentially with the number of trackers. However, we do not consider this a significant issue – due to ”real-time” requirements of tracking, the need to combine more than a small number of trackers, say $\mathbf{n} = 4$, is unlikely.

The HMMTxD method overview is illustrated in Figure 5.1. Each tracker provides an estimate of the object pose ($\mathbf{B}_i$) and a vector of observables ($\mathbf{x}_i$), which may contain a similarity measure to some model (such as normalized cross-correlation to the initial image patch or a distance of template and current histograms at given position) or any other estimates of the tracker performance. The $\mathbf{x}_i, i = \{1, 2, \ldots, \mathbf{n}\}$ serve as observables to relate the tracker current confidence to the HMM. Each individual observable depends only on one particular tracker and its correctness, hence, they are assumed to be conditionally independent conditioned on the state of the HMM (which encodes the tracker correctness).

In general, there are no constraints on observable values, however, in the proposed HMM the observable values are required to be normalized to the $(0, 1)$ interval. The observables are modeled as beta-distributed random variables (Eq. 5.1), and its parameters are estimated online. The beta distribution was chosen for its versatility, where practically any kind of unimodal random variable on $(0, 1)$ can be modeled by the beta distribution, i.e. for any choice of any lower and upper quantiles, a beta distribution

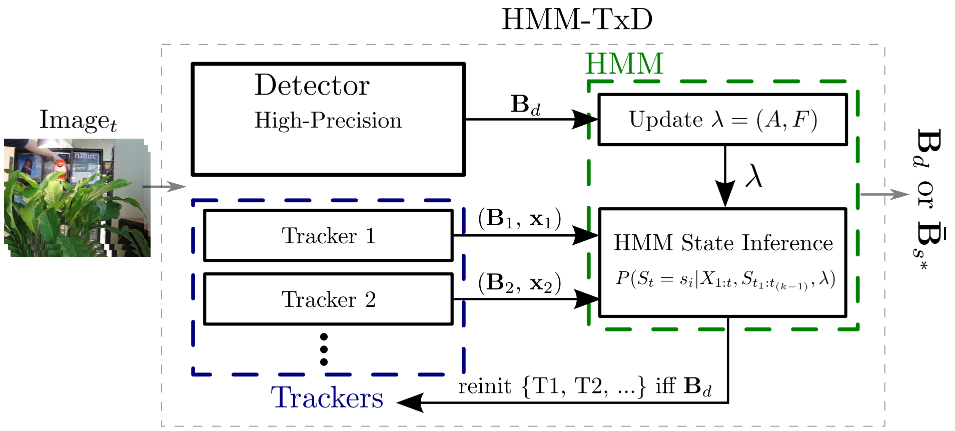

HMM-TxD
Image$_t$
Detector
High-Precision
$\mathbf{B}_d$
HMM
Update $\lambda = (A, F)$
λ
Tracker 1
Tracker 2
$(\mathbf{B}_1, \mathbf{x}_1)$
$(\mathbf{B}_2, \mathbf{x}_2)$
HMM State Inference
$P(S_t = s_i | X_{1:t}, S_{t_1:t_{(k-1)}}, \lambda)$
Trackers
reinit {T1, T2, ...} iff $\mathbf{B}_d$
$\mathbf{B}_d$ or $\bar{\mathbf{B}}_{s^*}$

Figure 5.1: The structure of the HMMTxD. For each frame, the detector and trackers are run. Each tracker outputs a new object pose and observables $(\mathbf{B}_i, \mathbf{x}_i)$ and the detector outputs either the verified object pose $\mathbf{B}_d$ or nothing. If detector fires, HMM is updated and trackers are reinitialized and the final output is the $\mathbf{B}_d$, otherwise, HMM estimate the most probable state s^* and outputs an average bounding box $\bar{\mathbf{B}}_{s^*}$ of trackers that are correct in the estimated state s^*.

exists satisfying the given quantile constraint [GN04].

Learning the parameters of the beta distributions online is crucial for the adaptability to particular tracking scenes, where the observable values from different trackers may be biased due to scene properties, or to adapt to different types of observables of trackers and their correlations to the ”real” tracker performance. For example, taking correlation with the initial target patch as an observable for one tracker and color histogram distance to an initial target for a second tracker, the correlation between their values and the performance of the tracker may differ depending on object rigidity and color distribution of object and background.

The HMM is parameterized by the pair $\lambda = (A, F)$, where A are the probabilities of state transition and F are the beta distributions of observables with shape parameters $p, q > 0$ and density defined for $x \in (0, 1)$

$$f(x|p,q) = \frac{x^{p-1}(1-x)^{q-1}}{\int_0^1 u^{p-1}(1-u)^{q-1}du}. \tag{5.1}$$

Since the goal is real-time tracking without any specific pre-processing, learning of HMM parameters has to be done online. Towards this goal, the object detector, which is set to operating mode with low false positive rate, is utilized to annotate the sequence of hidden states partially. In contrast to classical HMM, where only a sequence of observations $\mathbb{X} = \{X_t\}_{t=1}^{T}, X_t = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x_n})$ is available, we are in a semi-supervised setting and have a time sequence $0 = t_0 < t_1 < t_2 \ldots < t_K \leq T$ of observed states of a Markov chain $\mathbb{S} = \{S_{t_k} = s_{i_k}, \{t_k\}_{k=1}^{K}\}$, and Markov chain starting again in state s_1, all trackers correct, at any time $\{t_k + 1, 0 \leq k \leq K\}$, since there are reinitialized to common object pose. This information is provided by the detector, where $\{t_k\}_{k=1}^{K}$ is a sequence of detection times. The HMM parameters are learned by a modified Baum-Welch algorithm runs on the observations $\mathbb{X}$ and the annotated sequence of states $\mathbb{S}$. The partial annotation and HMM parameter estimation update is done strictly online.

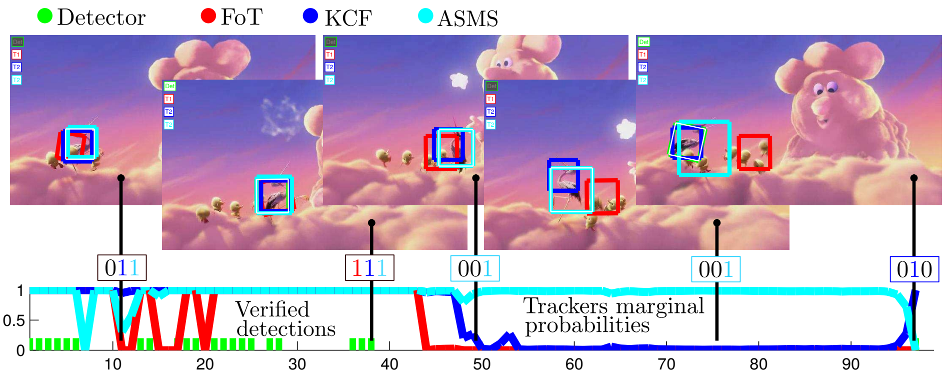


Figure 5.2: Illustration of HMM state and trackers probability estimation during tracking. The bottom graph shows the marginal probabilities for each tracker being correct and the detection times (green spikes). Above the graph the inferred states s_t^* with color encoded correct trackers (1) are displayed. The final output is defined by the state s_t^* and the bounding box is highlighted by white color. Best viewed zoomed in color.

The output of the HMMTxD is an average bounding box of correct trackers of the current most probable state s_t^*. For $t_{(k-1)} < t < t_k, 1 \leq k \leq K$ the forward-backward procedure [Rab89] for HMM is used to calculate probability of each state at time t (see Eq. 5.15-5.21) and the state $s_t^* \in \{0,1\}^{\mathbf{n}} \setminus (0,0,\dots,0)$ is the state for which

$$P(S_t = s_i | X_1, \dots, X_t, S_{t_1}, \dots, S_{t_{(k-1)}}, \lambda) \tag{5.2}$$

is maximal. This equation is computed using Eq. 5.19 and maximized w.r.t $i, 1 \leq i \leq N$. For $t_K < t \leq T$ the Eq. 5.2 holds with $t_{(k-1)} = t_K$. This ensures that the algorithm outputs a pose for each frame which is required by most benchmark protocols. Illustration of the tracking process and HMM insight is shown in Figure 5.2. Theoretically, the parameters of HMM could be updated after each frame. However, in our implementation, learning takes place only at frames where the detector positively detects the object, i.e. the sequence of states starting and ending with observed state inferred by the detector[1]. The detector is only used if the detection pose is not in contradiction with the pose of the current most probable state in which the majority of trackers are correct. This ensures that even when the detector makes a mistake, the HMM is not wrongly updated. When we are in the state that one or none of the trackers are correct, the detector gets precedence.

5.3 Learning the Hidden Markov Model

For learning of the parameters λ of the HMM an MLE inference is employed, however maximizing the likelihood function $P(\mathbb{X}, \mathbb{S}|\lambda)$ is a complicated task that cannot be solved analytically. In the proposed method, the Baum-Welch algorithm [BPSW70] is adapted. The Baum-Welch algorithm is a widespread iterative procedure for estimating parameters of HMM where each iteration increases the likelihood function but,

[1] If pure online fusion is not required, future observations can also be used to determine the probability of each state.

in general, the convergence to the global maximum is not guaranteed. The Baum-Welch algorithm is, in fact, an application of the EM (Expectation-Maximization) algorithm [DLR77].

5.3.1 Classical Baum-Welch Algorithm

Let us assume the HMM with N possible states $\{s_1, s_2, \dots, s_N\}$, the matrix of state transition probabilities $A = \{a_{ij}\}_{i,j=1}^N$, the vector of initial state probabilities $\pi = (1, 0, 0, \dots, 0)$, the initial state $s_1 = (1, 1, \dots, 1)$, a sequence of observations $\mathbb{X} = \{X_t\}_{t=1}^T, X_t \in R^m$ and $F = \{f_i(x)\}_{i=1}^N$ the system of conditional probability densities of observations conditioned on $S_t = s_i$

$$f_i(x) = f(x|S_t = s_i) \text{ for } 1 \le i \le N, 1 \le t \le T, x \in R^m \tag{5.3}$$

where S_t are random variables representing the state at time t, and $\lambda = (A, F)$ is denoting the parameter set of the model.

Let us denote

$$Q(\lambda, \lambda') = \sum_{\mathfrak{s} \in \mathfrak{S}} P(\mathfrak{s}|\mathbb{X}, \lambda) \log[P(\mathfrak{s}, \mathbb{X}|\lambda')], \tag{5.4}$$

where $\mathfrak{S} = \{s_1, s_2, \dots, s_N\}^T$ is a set of all possible T-tuples of states and
$\mathfrak{s} \in \mathfrak{S}, \mathfrak{s} = (\mathfrak{s}_1, \dots, \mathfrak{s}_t, \dots, \mathfrak{s}_T)$ is one sequence of states. According to Theorem 2.1. in [BPSW70]

$$Q(\lambda, \lambda') \ge Q(\lambda, \lambda) \Rightarrow P(\mathbb{X}|\lambda') \ge P(\mathbb{X}|\lambda) \tag{5.5}$$

and the equality holds if and only if $P(\mathfrak{s}|\mathbb{X}, \lambda) = P(\mathfrak{s}|\mathbb{X}, \lambda')$ for $\forall \mathfrak{s} \in \mathfrak{S}$. The classical Baum-Welch algorithm repeats the following steps until convergence:

1. Compute $\lambda^* = \arg\max_\lambda Q(\lambda_n, \lambda)$

2. Set $\lambda_{n+1} = \lambda^*$.

5.3.2 Modified Baum-Welch Algorithm

We propose the modified Baum-Welch algorithm that exploits the partially annotated sequence of states, where the known states are inferred from the detector output. Let $0 = t_0 < t_1 < t_2 \dots < t_K \le T$ be a sequence of detection times, $\mathbb{S} = \{S_{t_k} = s_{i_k}, \{t_k\}_{k=1}^K\}$ be observed states of Markov chain, marked by the detector, and $S_{t_k+1} = s_1$ for $0 \le k \le K$. So the sequence of observations of the HMM is divided into $K+1$ independent subsequences, each with a fixed initial state s_1, the first K subsequences with a known terminal state defined by the detector and the last subsequence with an unknown terminal state.

The following equations are obtained by employing the modification to the Baum-Welch algorithm,

$$\log[P(\mathfrak{s}, \mathbb{X}, \mathbb{S}|\lambda)] = \sum_{t=1}^{T-1} \log a_{\mathfrak{s}_t \mathfrak{s}_{t+1}} + \sum_{t=1}^{T} \log f_{\mathfrak{s}_t}(X_t), \tag{5.6}$$

$$
\begin{aligned}
Q(\lambda_n, \lambda) = \sum_{\mathfrak{s} \in \mathfrak{S}} P(\mathfrak{s}|\mathbb{X}, \mathbb{S}, \lambda_n) \sum_{t=1}^{T-1} \log a_{\mathfrak{s}_t \mathfrak{s}_{t+1}} + \\
\sum_{\mathfrak{s} \in \mathfrak{S}} P(\mathfrak{s}|\mathbb{X}, \mathbb{S}, \lambda_n) \sum_{t=1}^{T} \log f_{\mathfrak{s}_t}(X_t).
\end{aligned}
\tag{5.7}
$$

The maximization of the $Q(\lambda_n, \lambda)$ can be separated to maximization w.r.t. transition probability matrix $A = \{a_{ij}\}_{i,j=1}^{N}$ by maximizing the first term and w.r.t. observable densities $F = \{f_i(x)\}_{i=1}^{N}$ by maximizing the second term.

The maximization of Eq. 5.7 w.r.t. A constrained by $\sum_{j=1}^{N} a_{ij} = 1$ for $1 \leq i \leq N$ is obtained by re-estimating the parameters $\hat{A} = \{\hat{a}_{ij}\}_{i,j=1}^{N}$ as follows:

$$
\begin{aligned}
\hat{a}_{ij} &= \frac{\text{expected number of transitions from state } s_i \text{ to state } s_j}{\text{expected number of transitions from state } s_i} \\
&= \frac{\sum_{(t=1 \text{ and } t \neq t_k, 1 \leq k \leq K)}^{T-1} P(S_t = s_i, S_{t+1} = s_j|\mathbb{X}, \mathbb{S}, \lambda)}{\sum_{(t=1 \text{ and } t \neq t_k, 1 \leq k \leq K)}^{T-1} P(S_t = s_i|\mathbb{X}, \mathbb{S}, \lambda)}.
\end{aligned}
\tag{5.8}
$$

This equation is computed using modified forward and backward variables of the Baum-Welch algorithm to reflect the partially annotated states. For the exact derivation of formulas for computation of $\hat{a}_{ij}$ see Appendix 5.8.

Learning Observable Distributions

The maximization of Eq. 5.7 w.r.t. $F = \{f_i(x)\}_{i=1}^{N}$ depends on assumptions on the system of probability densities F. It is usually assumed (e.g. in [Rab89, BPSW70]) that F is a system of probability distributions of the same type and differ only in their parameters.

In the HMMTxD the m-dimensional observed random variables $X_t = (X_t^1, X_t^2, \ldots, X_t^m) \in R^m$ are assumed conditionally independent and to have the beta-distribution, so $f_i(x), 1 \leq i \leq N$ are products of m one-dimensional beta distributions with parameters of shape $\{({p_i}^j, {q_i}^j)\}_{j=1}^{m}, 1 \leq i \leq N$. In this case maximization of the second term of Eq. 5.7 is an iterative procedure using inverse digamma function which is very computationally expensive [GN04].

We propose to estimate the shape parameters of the beta distributions with a generalized method of moments. The classical method of moments is based on the fact that sample moments of independent observations converge to its theoretical ones due to the law of large numbers for independent random variables. In the HMMTxD observations $\mathbb{X} = \{X_t\}_{t=1}^{T}$ are not independent. The generalized method of moments is based on the fact that $\{X_t - E(X_t|X_1, X_2, \ldots, X_{t-1})\}_{t=1}^{T}$ is a sequence of martingale differences for which the law of large numbers also holds. Using the generalized method of moments gives estimates of the parameters of shape

$$
\hat{p}_i^j = \hat{\mu}_i^j \left(\frac{\hat{\mu}_i^j (1 - \hat{\mu}_i^j)}{(\hat{\sigma}_i^j)^2} - 1 \right)
\tag{5.9}
$$

and

$$
\hat{q}_i^j = (1 - \hat{\mu}_i^j) \left(\frac{\hat{\mu}_i^j (1 - \hat{\mu}_i^j)}{(\hat{\sigma}_i^j)^2} - 1 \right)
\tag{5.10}
$$

where

$$\hat{\mu}_i^j = \frac{\sum_{t=1}^{T} X_t^j P(S_t = s_i|\mathbb{X}, \mathbb{S}, \lambda)}{\sum_{t=1}^{T} P(S_t = s_i|\mathbb{X}, \mathbb{S}, \lambda)} \tag{5.11}$$

and

$$(\hat{\sigma}_i^j)^2 = \frac{\sum_{t=1}^{T} (X_t^j - \hat{\mu}_i^j)^2 P(S_t = s_i|\mathbb{X}, \mathbb{S}, \lambda)}{\sum_{t=1}^{T} P(S_t = s_i|\mathbb{X}, \mathbb{S}, \lambda)}. \tag{5.12}$$

Let us denote the system of probability densities with re-estimated parameters as $\hat{F} = \{\hat{f}_i(x)\}_{i=1}^{N}$. The generalized method of moments is described in detail in the 5.9.

Algorithm Overview

The complete modified Baum-Welch algorithm is summarized in Alg. 4, where after each iteration $P(\mathbb{X}, \mathbb{S}|\lambda_{n+1}) \geq P(\mathbb{X}, \mathbb{S}|\lambda_n)$ and we repeat these steps until convergence. Note that $\hat{A}_n$ is a maximum likelihood estimate of A therefore always increases $P(\mathbb{X}, \mathbb{S}|\lambda_n)$ (shown in [Rab89]) but $\hat{F}_n$ is estimated by the method of moments so the test on likelihood increase is required (”if statement” in the Alg. 4). In fact, this algorithm structure matches to the generalized EM algorithm (GEM) introduced in [DLR77].

Algorithm 4: Algorithm for HMM parameters learning

```
Input: $\mathbb{X}, \mathbb{S}, \lambda_n = (A_n, F_n)$
Output: $\lambda_{n+1} = (A_{n+1}, F_{n+1})$
repeat
    Compute likelihood $P(\mathbb{X}, \mathbb{S}|\lambda_n)$
    Estimate $\hat{A}_n$ by Eq. 5.8 and $\hat{F}_n$ by Eq. 5.9, 5.10
    if $P(\mathbb{X}, \mathbb{S}|\hat{A}_n, \hat{F}_n) < P(\mathbb{X}, \mathbb{S}|A_n, F_n)$ then
        $\lambda_{n+1} = (\hat{A}_n, F_n)$
    else
        $\lambda_{n+1} = (\hat{A}_n, \hat{F}_n)$
    $\lambda_n = \lambda_{n+1} = (A_{n+1}, F_{n+1})$
until convergence $\vee$ max number of iteration
```

5.4 Feature-Based Detector

The requirements for the detector are adjustable operation mode (e.g. set for high precision but possibly low recall), (near) real-time performance and the ability to model pose transformations up to at least similarity (translation, rotation, isotropic scaling). Any detector-like approach can be used and it may vary based on application. We choose to adopt a feature-based detector which has been shown to perform well in image retrieval, object detection and object tracking [PB13] tasks.

There are many possible combinations of features and their descriptors with different advantages and drawbacks. We exploit multiple feature types: specifically, Hessian key points with the SIFT [Low04] descriptor, ORB [RRKB11] with BRISK and ORB with FREAK [Ort12]. Each feature type is handled separately, up to the point where point correspondences are established. A weight value is assigned to each feature type w^g and is set to be inversely proportional to the number of features on the reference template, to balance the disparity in individual feature numbers.

The detector works as follows. In the initialization step, features are extracted from the inside and the outside of the region specifying the tracked object. Descriptors of the features outside of the region are stored as the background model.

Usually, the input region is not 100% occupied by the target; therefore, fast color segmentation [KPSK14] attempts to delineate the object more precisely than the axis-aligned bounding box to remove the features that are most likely not on the target. The step is not critical for the function of the detector, since the bounding box is a fall-back option. We assume that at least 50% of the bounding box is filled with pixels that belong to the target, if the segmentation fails (returns a region containing less than 50% of the area of the bounding box), all features in the initial bounding box are used.

Additionally, for each target feature, we use a normal distribution $\mathcal{N}(\mu^f, \sigma^{2f})$ to model the similarity of the feature to other features. The parameters μ^f and σ^{2f} are estimated in the first frame by randomly sampling 100 features, other than f, and computing distances to the feature f, from which the mean and variation are computed. This allows defining the quality of correspondence matches in a probabilistic manner for each feature, thus getting rid of a global static threshold for the acceptable correspondence distance.

In the detection phase, features are detected and described in the whole image. For each feature g_i from the image the nearest neighbour (in Euclidean space or in Hamming distance metric space, depending on the feature type) feature b^* from the background model and the nearest neighbour feature f^* from the foreground model are computed. A tentative correspondence is formed if the feature match passes the second nearest neighbour test and a probability that the correspondence distance belongs to the outlier distribution is lower than a predefined significance set to 0.1%. So

$$\frac{d(g_i, f^*)}{d(g_i, b^*)} < 0.8 \;\wedge\; \mathcal{F}(d(g_i, f^*)|\mu^{f^*}, \sigma^{2f^*}) < 0.1\% \tag{5.13}$$

where $\mathcal{F}(d|\mu^{f^*}, \sigma^{2f^*})$ is a c.d.f. of the normal distribution with parameters μ^{f^*} and σ^{2f^*} of a distance distribution of features not corresponding to f^*. The 0.1% significance corresponds to the $\mu - 3\sigma$ threshold. Finally, RANSAC estimates the target current pose using a sum of weighted inliers as a cost function for model support

$$\text{cost} = \sum_i w^{g_i} * [g_i == \text{inlier}], \tag{5.14}$$

which takes into account the different numbers of features per feature type on the target.

The decision whether the detected pose is considered correct depends on the number of weighted inliers that supports the RANSAC-selected transformation and it controls the trade-off between precision and recall of the method. This threshold is automatically computed in the first frame of the sequence as $\max(5, \min(0.03 * \#\text{fc}, 10))$, where #fc is the number of features in the initial bounding box. The threshold interval (5,10) and the feature multiplier (0.03) were set experimentally to have the false positive rate close to zero for the most of the testing sequences. Furthermore, majority voting is used to

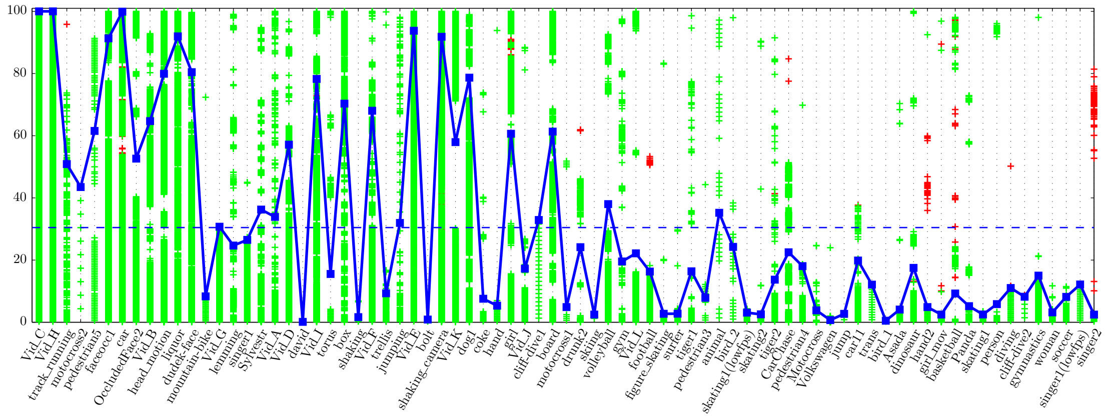


Figure 5.3: Frames with the detections for 77 sequences dataset. The green marks show true positive detections and red marks are false positive. The blue line shows the recall of the detector and blue dashed line shows the average recall over all sequences. The length of each sequence is normalized to range $(0, 100)$.

verify that the detection is not in contradiction to the estimated HMM state, i.e. if we are in the state where two or more (majority) trackers are correct and the detector is not consistent with them, the detection is not used. This mitigates the false positive detections, therefore HMM updates, when the trackers work correctly (based on the HMM model).

The true and false positives for 77 sequences are shown in Figure 5.3, where the detector works on almost all sequences with zero false positive rate (0.46% average false positive rate on the dataset) and 30% recall rate. The failure cases of this feature-based detector are mostly caused by the imprecise initial bounding box, which contains a significant portion of structured background (i.e. background where the detector finds features) and due to the presence of a similar object in the scene, e.g. sequences *hand2, basketball, singer2*.

5.5 HMMTxD Implementation

To demonstrate the performance of the proposed framework, a pair and a triplet of published short-term trackers were plugged into the framework to show the performance gain by a combination of a different number of trackers. As Bailer et al. [BPS14] pointed out, not all trackers, when combined, can improve the overall performance (i.e. adding tracking method with similar failure mode will not benefit).

Therefore, we choose methods that have different designs and work with different assumptions (e.g. rigid global motion vs. color mean-shift estimation vs. maximum correlation response). These trackers are the Flock of Trackers (FoT) [VM14], scale adaptive mean-shift tracker (ASMS) [VNM13] and kernelized correlation filters (KCF) [HCMB15]. This choice shows that superior performance can be achieved by using simple, fast trackers (above 100fps) that may not represent the state-of-the-art. The trackers can be arbitrarily replaced depending on the user application or requirements.

Trackers

The Flock of Trackers (FoT) [VM14] evenly covers the object with patches and establishes a frame-to-frame correspondence by the Lucas-Kanade method [LK81]. The global motion of the target is estimated by RANSAC.

The second tracker is a scale adaptive mean-shift tracker (ASMS) [VNM13] where the object pose is estimated by minimizing the distance between RGB histograms of the reference and the candidate bounding box. The KCF [HCMB15] tracker learns a correlation filter by a ridge regression to have high response to the target object and low response on a background. The correlation is done in the Fourier domain which is very efficient.

These three trackers have been selected since they are complementary by design. FoT enforces a global motion constraint and works best for a rigid object with texture. On the other hand, ASMS does not enforce object rigidity and is well suited for articulated or deformable objects assuming their color distribution is discriminative w.r.t. the background. KCF can be viewed as a tracking-by-detection approach using sliding window like scanning.

For each tracker position, two global observable measurements are computed, namely the Hellinger distance between the target template histogram and the histogram of the current position and the normalized cross-correlation score of the current patch and the target model patch. These target models are initialized in the first frame and then updated exponentially with factor of 0.5 during each positive detection of the detector part. Additionally, each tracker produces its estimate of performance. For FoT it is the number of predicted correspondences (for details please see [VM14]) that support the global model. For ASMS it is the Hellinger distance between its histogram model and current neighbourhood background (i.e. color similarity of the object and background) and for KCF it is a correlation response of the tracking procedure.

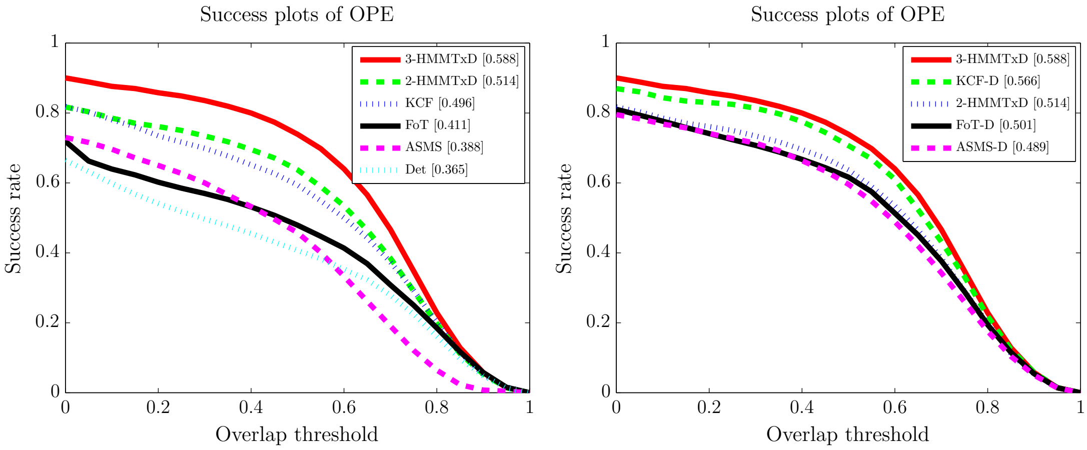


Figure 5.4: CVPR2013 OPE benchmark comparison of individual trackers and their combination in the proposed HMMTxD. The 2-HMMTxD denotes the combination of FoT and ASMS trackers and 3-HMMTxD is a combination of FoT, ASMS and KCF trackers. Det stands for the proposed detector. The right plot show simple combination of individual trackers with the proposed detector. Suffix ”-D” refers to the combination with detector.

5.6 Experiments

The HMMTxD was compared with state-of-the-art methods on two standard benchmarks and on a dataset TV77[2] containing 77 public video sequences collected from tracking-related publications. The dataset exhibits a wider diversity of content and variability of conditions than the benchmarks.

Parameters of the method were fixed for all the experiments. In the HMM, the initial beta distribution shape parameters (p, q) were set to $(2, 1)$ for the correct state (1) and $(1, 2)$ for fail state (0) for all observations. The transition matrix was set to prefer staying in the current state and has 0.98 on diagonal, 0 in fist column, 0.001 in the last column, $1e - 10$ in the last row and 0.05 otherwise. The matrix is normalized so that rows sum to one. States in the matrix are binary encoded starting from the left column which corresponds to the state $s_1 = (1, ..., 1)$. The number of iterations for the Baum-Welch alg. was set to 3.

The processing speed on the VOT2015 dataset is (in frames per second) minimum 1.03, maximum 33.72 and average 10.83 measured on a standard notebook with Intel Core-i7 processor. This speed is mostly affected by the number of features detected in the images which correlate to the resolution of the image (in the dataset the range is from 320x180 to 1280x720).

First, we compare the performance of individual parts of the HMMTxD framework (i.e. KCF, ASMS, FoT trackers) and their combination via HMM as proposed in this paper. Two variants of HMMTxD are evaluated – 2-HMMTxD refers to a combination of FoT and ASMS trackers and the 3-HMMTxD to a combination of all mentioned trackers. We also show the benefit of the proposed detector when simply combined with the individual trackers in such way that if detector fires the tracker is reinitialized. The Figure 5.4 shows the benefit gained from the detector and further consistent improvement achieved by the combination of the trackers. More detailed per sequence analysis on the TV77 dataset (Figure 5.5 and Figure 5.6) shows more clearly the efficiency of learning tracker performance online. In almost all sequences the HMMTxD is able to identify and learn which trackers work correctly and achieve the performance of at least the best tracker or higher (e.g. *motocross1, skating1(low), Volkswagen, singer1, pedestrian3, surfer*). Most notable failure cases are caused by the detector failure, e.g. in sequences *singer2, woman, skating1, basketball, girl_mov*.

In all other experiments, the abbreviation HMMTxD refers to the combination of all 3 trackers.

Evaluation on the CVPR2013 Benchmark [WLY13] that contains 50 video sequences. Results on the benchmark have been published for about 30 trackers. The benchmark defines three types of experiments: (i) one-pass evaluation (OPE) – a tracker initialized in the first frame is run to the end of the sequence, (ii) temporal robustness evaluation (TRE) – the tracker is initialized and starts at a random frame, and (iii) spatial robustness evaluation (SRE) – the initialization is perturbed spatially. Performance is measured by precision (spatial accuracy, i.e. a center distance of ground truth and reported bounding box) and success rate (the number of frames where the overlap with the ground truth was higher than a threshold). The results are visualized in Figure 5.7

[2] http://cmp.felk.cvut.cz/~vojirtom/dataset/index.html

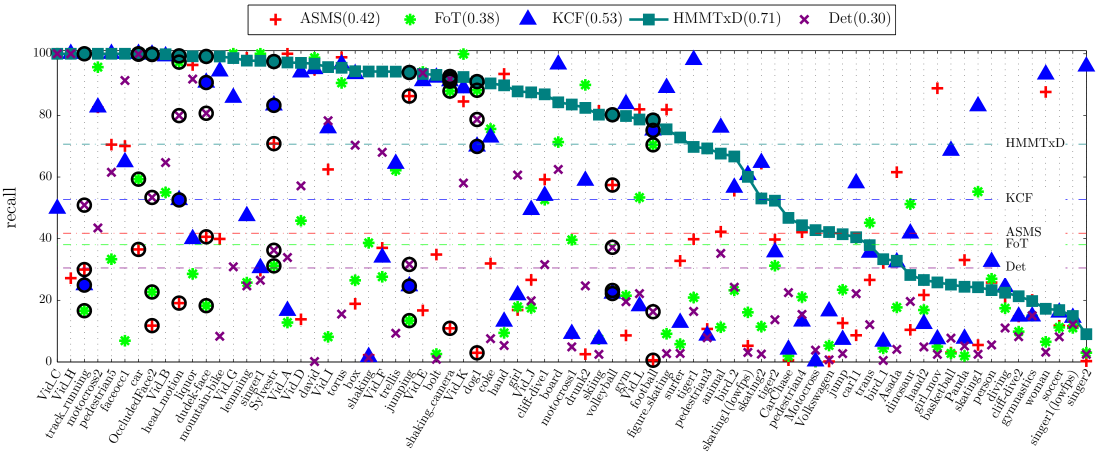


Figure 5.5: Per sequence analysis of the single trackers (i.e. KCF, ASMS, FoT) and the proposed HMMTxD. The average recall is shown by the dashed lines (precise number is in the legend). Black circles mark grayscale sequences. The sequences are ordered by HMMTxD performance.

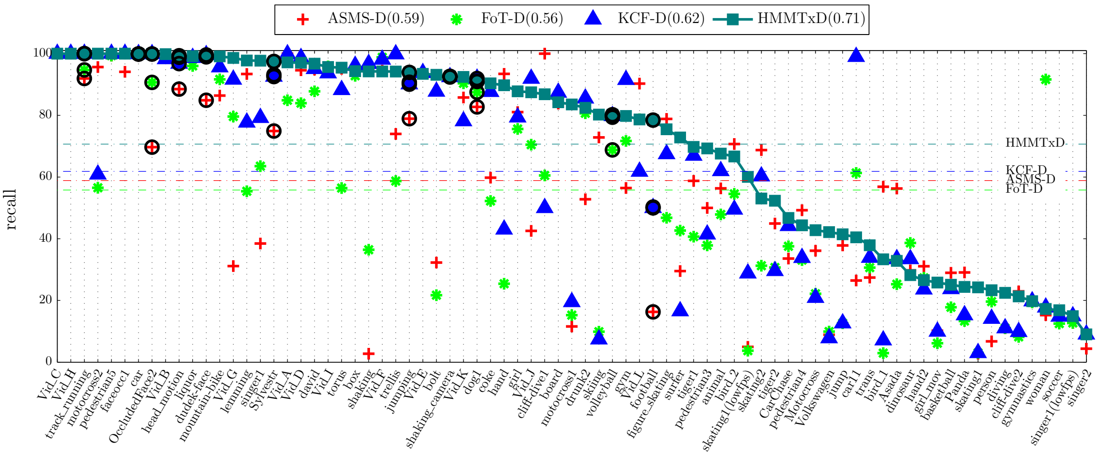


Figure 5.6: Per sequence analysis of the single trackers combined with the detector (i.e. KCF-D, ASMS-D, FoT-D) and the proposed HMMTxD. The average recall is shown by the dashed lines (precise number is in the legend). Black circles mark grayscale sequences. The sequences are ordered by HMMTxD performance.

where only results of the 10 top performing trackers are plotted. Together with the tracker from this benchmark, we also added the MEEM [ZMS14] tracker, which is a recent state-of-the-art tracker. The proposed HMMTxD outperforms all trackers in the success rate in all three experiments. Its precision is comparable to MEEM [ZMS14] the top performing tracker in terms of precision. HMMTxD outperforms significantly the OPE results reported in Wang et al. [WY14], where 5 top performing trackers from this particular benchmark were used for combination (other experiments were not reported in the paper).

VOT2013 benchmark [KPL$^+$13] evaluates trackers on a collection containing 16 sequences carefully selected from a large pool by a semi- automatic clustering method. For comparison, results of 27 tracking methods are available and the added MEEM

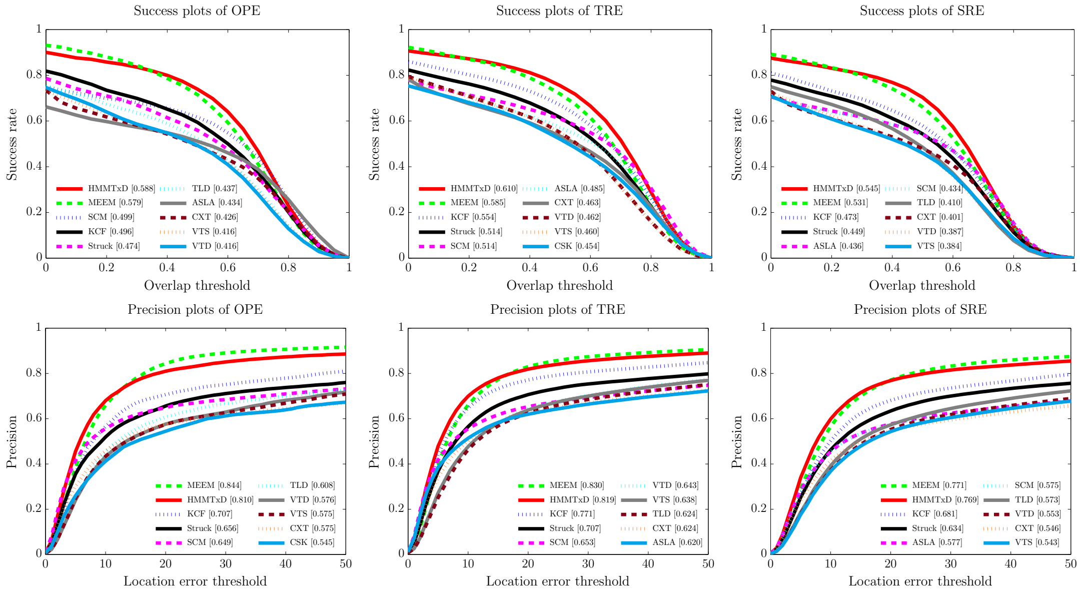
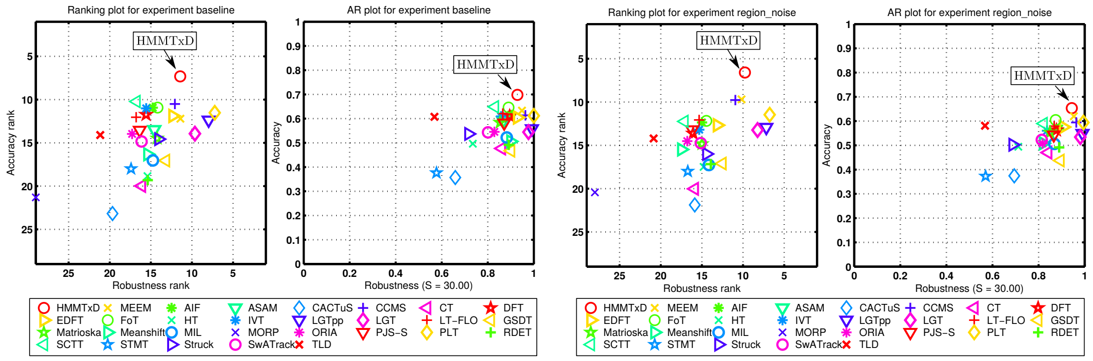


Figure 5.7: Evaluation of HMMTxD on the CVPR2013 Benchmark [WLY13]. The top row shows the success rate as a function of the overlap threshold. The bottom row shows the precision as a function of the localization error threshold. The number in the legend is AUC, the area under ROC-curve, which summarizes the overall performance of the tracker for each experiment.

tracker was evaluated by us using default setting from the publicly available source code. The performance is measured by accuracy – average overlap with the ground truth – and robustness – the number of re-initialization of the tracker so that it is able to track the whole sequence. Average rank of trackers is used as an overall performance indicator.



(a) baseline experiment

(b) region-noise experiment

Figure 5.8: Evaluation of HMMTxD on the VOT 2013 Benchmark [KPL$^+$13]. HMMTxD result is shown as the red circle. The left plot shows the ranking in accuracy (vertical axis) and robustness (horizontal axis) and the right plot shows the raw average values of accuracy and robustness (normalized to the $(0, 1)$ interval). For both plots the top right corner is the best performance.

In this benchmark, the proposed HMMTxD achieves clearly the best accuracy (Fig-

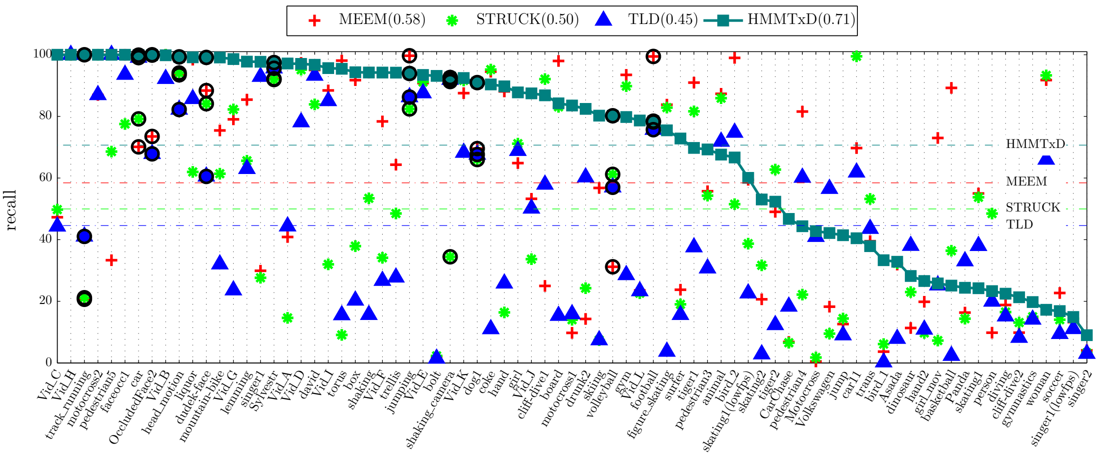


Figure 5.9: Evaluation of state-of-the-art trackers on the TV77 dataset in terms of recall, i.e. number of correctly tracked frames. The average recall is shown by the dashed lines (precise number is in the legend). Black circles mark grayscale sequences. The sequences are ordered by HMMTxD performance.

ure 5.8). With less than one re-initialization per sequence, it performs slightly worse in terms of robustness due to two reasons.

Firstly, the HMM recognizes a tracker problem with a delay and switching to other tracker (here even one frame where the overlap with the ground truth is zero leads to penalization) and secondly the VOT evaluation protocol, which requires re-initialization after failure and to forget all previously learned models (the VOT2013 refer to this as causal tracking), therefore the learned performance of the trackers is forgotten and has to be learned from scratch.

The results of the baseline and region-noise experiments are shown in Figure 5.8. Note that the ranking of the methods differs from the original publication since two new methods (HMMTxD and MEEM) were added and the relative ranking of the methods changed. The top three performing trackers and their average ranks are HMMTxD (8.77), PLT (9.24), LGTpp [XSL13] (10.11). MEEM tracker ends up at the fifth place with average rank 10.87. The ranking was obtained by the provided VOT toolkit in default settings for baseline and region noise experiments.

The second best performing method on the VOT2013 is the unpublished PLT for which just a short description is available in [KPL$^+$13]. PLT is a variation of structural SVM that uses multiple features (color, gradients). STRUCK [HST11] and MEEM [ZMS14] are similar methods to the PLT based on SVM classification. We compared these methods with HMMTxD on the diverse 77 videos along with the TLD [KMM12] which has a similar design as HMMTxD. HMMTxD outperforms all these methods by a large margin on average recall – measured as the number of frames where the tracker overlap with the ground truth is higher than 0.5 averaged over all sequences. Results are shown in Figure 5.9. Qualitative comparison of these state-of-the-art methods is shown in Figure 5.10. Even for sequences with lower recall (e.g. *bird_1, skating2*), the HMMTxD is able to follow the object of interest.

Other VOT benchmarks (2014, 2015) [KPL$^+$14, KML$^+$15] were also used to evaluate the performance of the HMMTxD tracker. In the VOT2014 Challenge, described

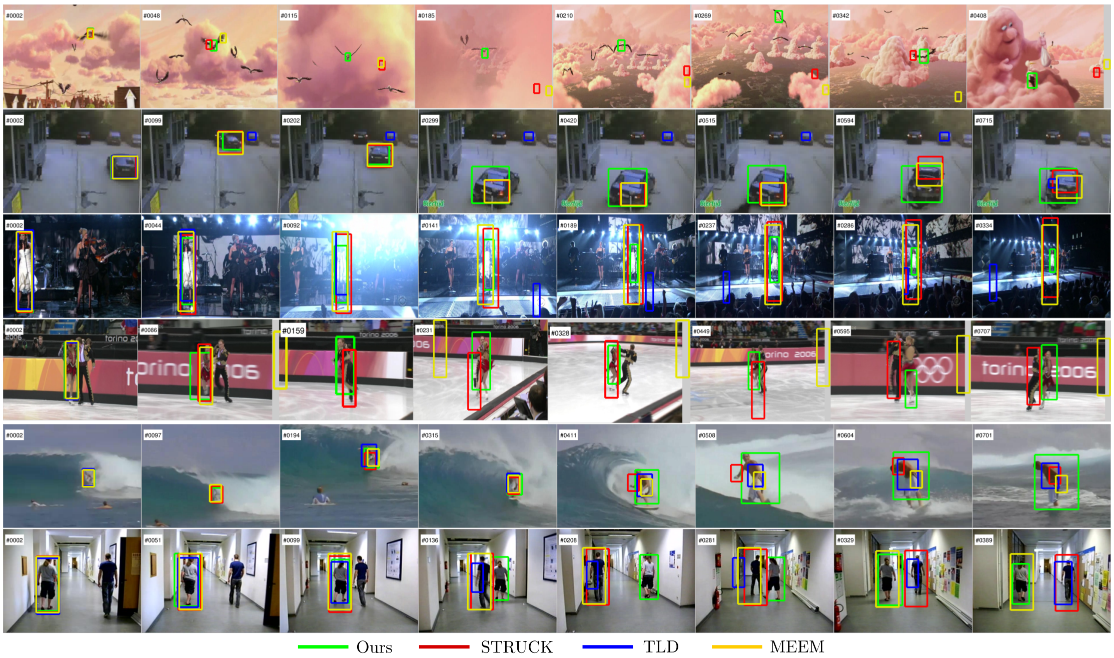


Figure 5.10: Qualitative comparison of the state-of-the-art trackers on challenging sequences from the TV77 dataset (from top *bird_1, drunk2, singer1, skating2, surfer, Vid_J*).

in the [KPL$^+$14], the HMMTxD fused only the FoT and ASMS tracker and ranked 8 of 33 trackers. In the VOT 2015 Challenge, the HMMTxD, which was extended by the baseline KCF tracker, achieved the state-of-the-art results i.e. outperformed the state-of-the-art bound as estimated by the VOT, even though the individual trackers did not score well (except ASMS). Moreover, the HMMTxD outperformed all the individual trackers that are used for fusion, hence, demonstrating the effectiveness of the proposed method. The VOT2015 results are described in Section 3.5. In overall rank, the HMMTxD ranks as 19 of 62, whereas the ASMS ranks 20, FoT ranks 50 and improved versions of the baseline KCF tracker rank 40 and 41. These ranks show that the only tracker, from the combination, that performed well was ASMS tracker, which was rightly identified by the HMMTxD and the performance was slightly improved by exploiting the other trackers in small portions of the dataset.

5.7 Conclusions

A novel method called HMMTxD for fusion of multiple trackers had been proposed. The method utilizes an on-line trained HMM to estimate the states of the individual trackers and to fuse different types of observables provided by the trackers. The HMMTxD outperforms its constituent parts (FoT, ASMS, KCF, Detector and its combinations) by a large margin and shows the efficiency of the HMM with the combination of three trackers.

HMMTxD outperforms all methods included in the CVPR2013 benchmark and performs favorably against most recent state-of-the-art tracker. The HMMTxD also outperforms all method of the VOT2013 benchmark in accuracy, while maintaining good robustness, and ranking in the first place in the overall ranking. Experiments conducted on a diverse dataset TV77 show that the HMMTxD outperforms state-of-the-art MEEM, STRUCK and TLD methods, which are similar in design, by a large margin. The processing speed of the HMMTxD is $5 - 15$ frames per second on average, which is comparable with other complex tracking methods.

5.8 Appendix - Forward-Backward Procedure for Modified Baum-Welch Algorithm

Let us assume the HMM with N possible states $\{s_1, s_2, \ldots, s_N\}$, the matrix of state transition probabilities $A = \{a_{ij}\}_{i,j=1}^{N}$, the vector of initial state probabilities $\pi = (1, 0, 0, \ldots, 0)$, the initial state $s_1 = (1, 1, \ldots, 1)$, a sequence of observations $\mathbb{X} = \{X_t\}_{t=1}^{T}, X_t \in R^m$ and $F = \{f_i(x)\}_{i=1}^{N}$ the system of conditional probability densities of observations conditioned on $S_t = s_i$.

Let $0 = t_0 < t_1 < t_2 \ldots < t_K \leq T$ be a sequence of detection times, $\mathbb{S} = \{S_{t_k} = s_{i_k}, \{t_k\}_{k=1}^{K}\}$ be observed states of Markov chain, marked by the detector, and $S_{t_k+1} = s_1$ for $0 \leq k \leq K$.

The forward variable for the Baum-Welch algorithm is defined as follows. Let $1 \leq i \leq N, 1 \leq k \leq K, t_{(k-1)} < t \leq t_k$ and

$$\alpha_t(i) = P(X_{t_{(k-1)}+1}, \ldots, X_t, S_t = s_i|\lambda) \text{ then} \tag{5.15}$$

$$\alpha_{t_{(k-1)}+1}(1) = f_1(X_{t_{(k-1)}+1}), \tag{5.16}$$

$$\alpha_{t_{(k-1)}+1}(i) = 0 \text{ for } i \neq 1 \tag{5.17}$$

and for $t_{(k-1)} < t < t_k$

$$\alpha_{(t+1)}(i) = \sum_{j=1}^{N} \alpha_t(j) a_{ji} f_i(X_{t+1}), \tag{5.18}$$

$$P(S_t = s_i|X_1, \ldots X_t, S_{t_1}, S_{t_2}, \ldots, S_{t_{(k-1)}}, \lambda) = \frac{\alpha_t(i)}{\sum_{j=1}^{N} \alpha_t(j)}. \tag{5.19}$$

For $t_K < t < T$ the forward variable is in principle the same as above with $t_{(k-1)} = t_K$. So

$$P(X_{t_K+1}, \ldots, X_T|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{5.20}$$

$$P(\mathbb{X}, \mathbb{S}|\lambda) = \prod_{k=1}^{K} \alpha_{t_k}(i_k) * \sum_{i=1}^{N} \alpha_T(i) \quad \text{where } S_{t_k} = s_{i_k}. \tag{5.21}$$

The backward variable for $t_{(k-1)} < t < t_k$ is

$$\beta_t(i) = P(X_{t+1}, \ldots, X_{t_k}, S_{t_k}|S_t = s_i, \lambda), \tag{5.22}$$

where $\beta_{t_k}(i_k) = 1$ and $\beta_{t_k}(i) = 0$ for $i \neq i_k$ and

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} f_j(X_{t+1}) \beta_{t+1}(j). \tag{5.23}$$

For $t_K < t < T$ the backward variable is in principle the same as above where $\beta_T(i) = 1$ for $1 \leq i \leq N$.

Given the forward and backward variables, we get the following probabilities, that are used to update parameters of HMM. For $0 < t < T$ and $t \neq t_k, 1 \leq k \leq K$

$$P(S_t = s_i, S_{t+1} = s_j | \mathbb{X}, \mathbb{S}, \lambda) = \tag{5.24}$$

$$\frac{\alpha_t(i) a_{ij} f_j(X_{t+1}) \beta_{(t+1)}(j)}{\sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_t(k) a_{kl} f_l(X_{t+1}) \beta_{t+1}(l)} \tag{5.25}$$

and for $0 < t \leq T$

$$P(S_t = s_i | \mathbb{X}, \mathbb{S}, \lambda) = \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j) \beta_t(j)}. \tag{5.26}$$

The final equation for the update of transition probabilities A of HMM is as follows.

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } s_i \text{ to state } s_j}{\text{expected number of transitions from state } s_i} \tag{5.27}$$

$$= \frac{\sum_{(t=1 \text{ and } t \neq t_k, 1 \leq k \leq K)}^{T-1} P(S_t = s_i, S_{t+1} = s_j | \mathbb{X}, \mathbb{S}, \lambda)}{\sum_{(t=1 \text{ and } t \neq t_k, 1 \leq k \leq K)}^{T-1} P(S_t = s_i | \mathbb{X}, \mathbb{S}, \lambda)}. \tag{5.28}$$

5.9 Appendix - Generalized Method of Moments

For a simplification let us assume HMM with one-dimensional observed random variables $\{X_t\}_{t=1}^{+\infty}, X_t \in R$. The sequence $\{X_t - E(X_t | X_1, X_2, \ldots, X_{t-1})\}_{t=1}^{+\infty}$ is a martingale difference series where

$$E(X_t | X_1, X_2, \ldots, X_{t-1}) = \sum_{i=1}^{N} E(X_t | X_1, X_2, \ldots, X_{t-1}, S_t = i) P(S_t = i) \tag{5.29}$$

$$= \sum_{i=1}^{N} E(X_t | S_t = i) P(S_t = i). \tag{5.30}$$

Under the assumption that $\{X_t\}_{t=1}^{+\infty}$ are uniformly bounded random variables i.e. $|X_t| < c, c \in (0, +\infty)$ for all $t \geq 1$, the strong law of large numbers for a sum of martingale differences can be used(see Theorem 2.19 in [HH80]). So

$$\lim_{T \to +\infty} \frac{1}{T} \sum_{t=1}^{T} [X_t - \sum_{i=1}^{N} E(X_t | S_t = i) P(S_t = i)] = 0 \text{ almost surely.} \tag{5.31}$$

Let us denote $\mu_i = E(X_t|S_t = i)$ for $1 \leq t \leq T$ and $\hat{\mu}_i$ the estimate of μ_i based on the modified method of moments. The estimate $\hat{\mu}_i$ is a solution of a following equation w.r.t. μ_i

$$\frac{1}{T}\sum_{t=1}^{T} X_t = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N} \mu_i P(S_t = i). \tag{5.32}$$

Having one equation for N unknown variables $\mu_i, 1 \leq i \leq N$ it is necessary to add some constrains to get a unique solution. We propose to minimize

$$\sum_{t=1}^{T}\sum_{i=1}^{N}(X_t - \mu_i)^2 P(S_t = i), \tag{5.33}$$

w.r.t. $\mu_i, 1 \leq i \leq N$ giving

$$\hat{\mu}_i = \frac{\sum_{t=1}^{T} X_t P(S_t = s_i)}{\sum_{t=1}^{T} P(S_t = s_i)} \tag{5.34}$$

which satisfy the moment equation (5.32). The same way of reasoning can be used for higher moments of $\{X_t\}_{t=1}^{T}$. For example using $\{(X_t)^2\}_{t=1}^{T}$ we get estimates $\hat{\sigma}_i^2$ for $\sigma_i^2 = var(X_t|S_t = i)$ for $1 \leq t \leq T$,

$$\hat{\sigma}_i^2 = \frac{\sum_{t=1}^{T}(X_t - \hat{\mu}_i)^2 P(S_t = s_i)}{\sum_{t=1}^{T} P(S_t = s_i)}. \tag{5.35}$$

In the HMMTxD m-dimensional observed random variables $X_t = (X_t^1, X_t^2, \ldots, X_t^m)$ are assumed, each of them having beta- distribution and being conditionally independent. There are well-known relations for a mean value EX and a variance $varX$ of a random variable X having beta distribution and its shape parameters (p, q)

$$p = EX\left(\frac{EX(1-EX)}{varX} - 1\right) \tag{5.36}$$

and

$$q = (1-EX)\left(\frac{EX(1-EX)}{varX} - 1\right). \tag{5.37}$$

Using the modified method of moments gives

$$\hat{\mu}_i^j = \frac{\sum_{t=1}^{T} X_t^j P(S_t = s_i|\mathbb{X}, \mathbb{S}, \lambda)}{\sum_{t=1}^{T} P(S_t = s_i|\mathbb{X}, \mathbb{S}, \lambda)} \tag{5.38}$$

and

$$(\hat{\sigma}_i^j)^2 = \frac{\sum_{t=1}^{T}(X_t^j - \hat{\mu}_i^j)^2 P(S_t = s_i|\mathbb{X}, \mathbb{S}, \lambda)}{\sum_{t=1}^{T} P(S_t = s_i|\mathbb{X}, \mathbb{S}, \lambda)}. \tag{5.39}$$

Then

$$\hat{p}_i^j = \hat{\mu}_i^j\left(\frac{\hat{\mu}_i^j(1-\hat{\mu}_i^j)}{(\hat{\sigma}_i^j)^2} - 1\right) \tag{5.40}$$

and

$$\hat{q}_i^j = (1-\hat{\mu}_i^j)\left(\frac{\hat{\mu}_i^j(1-\hat{\mu}_i^j)}{(\hat{\sigma}_i^j)^2} - 1\right). \tag{5.41}$$

If we assume in our model $\lambda = (A, F)$ that for some $\{(i_r, j_r) \in \{1, 2, \ldots, N\} \times \{1, 2, \ldots, m\} : p_{i_r}^{j_r} = p, q_{i_r}^{j_r} = q\}_{r=1}^{R}$ then

$$\hat{p} = \hat{\mu}\left(\frac{\hat{\mu}(1-\hat{\mu})}{\hat{\sigma}^2} - 1\right) \tag{5.42}$$

and

$$\hat{q} = (1-\hat{\mu})\left(\frac{\hat{\mu}(1-\hat{\mu})}{\hat{\sigma}^2} - 1\right) \tag{5.43}$$

where

$$\hat{\mu} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} X_t^{j_r} P(S_t = s_{i_r}|\mathbb{X}, \mathbb{S}, \lambda)}{\sum_{r=1}^{R}\sum_{t=1}^{T} P(S_t = s_{i_r}|\mathbb{X}, \mathbb{S}, \lambda)} \tag{5.44}$$

and

$$\hat{\sigma}^2 = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} (X_t^{j_r} - \hat{\mu})^2 P(S_t = s_{i_r}|\mathbb{X}, \mathbb{S}, \lambda)}{\sum_{r=1}^{R}\sum_{t=1}^{T} P(S_t = s_{i_r}|\mathbb{X}, \mathbb{S}, \lambda)}. \tag{5.45}$$

Chapter 6

Conclusions

In the thesis, contributions to the short-term single target visual object tracking are presented. Three tracking approaches were proposed each addressing a different aspect of the tracking task. The first two, Flock of Trackers (FoT) and scale-adaptive meansshift (ASMS), are robust short-term tracking methods with extremely fast processing speed. The FoT method computes sparse optical flow correspondences from which a target pose is robustly estimated via RANSAC. The RANSAC as a target pose estimator allows for range of complex rigid transformation (e.g. affine transformation) of the target between consecutive frames of a video sequence. The FoT achieved state-of-the-art performance in the VOT2013 benchmark at processing speed between 150-300 fames per second and was used as a building block for complex applications such as multi-object tracking frameworks (e.g. car tracking on highways [CVT+12]) or as an temporal link between text detection in the *TextSpotter* software[1] for text localization in the videos.

The ASMS tracker introduced two novelties to the mean-shift tracking. A theoretically justified method to optimize the target window position and size jointly and a technique to incorporate a background information into the optimization. The ASMS achieved state-of-the-art results in the VOT2015 benchmark at a speed higher than 100 frames per second and was declared as the best performing method in terms of a trade-off between performance and running time.

To take advantage of strengths of the proposed methods (FoT and ASMS), we proposed a novel tracking framework HMMTxD that fuses multiple tracking methods together with an online detector. The framework utilizes a hidden Markov model (HMM) to learn online how well each tracking method performs using sparse detector outputs and the estimates of confidence provided by the trackers. The HMM estimates the probability that a tracker is correct in the current frame given the previously learned HMM model and the tracker confidence. The HMMTxD framework enables to fuse several trackers with different designs to automatically adapt in general tracking scenarios. The HMMTxD demonstrated state-of-the-art performance in multiple benchmarks (VOT2014, VOT2015 and OTB) while running at around 15 frames per second on average.

The thesis provided an overview to the Visual Object Tracking (VOT) evaluation methodology and proposed an automatic algorithm for creating a dataset in a systematic

[1] http://textspotter.org/

and user unbiased manner and unified evaluation criterion (EAO) for tracker ranking. The VOT methodology enables the evaluation of different tracking approaches in a standard way and comparison of results across publications and time. Furthermore, workshops and tracking challenges on major conferences (ICCV 2013, ECCV 2014, ICCV 2015) were organized, where the most recent tracking methods, submitted by their authors, as well as baseline methods are compared and the result are discussed. These workshops allow the computer vision community to contribute to a discussion on further improvements to the methodology and to monitor the advance in the state-of-the-art in the visual object tracking.

6.1 Future Work Discussion

The possible feature work that would continue the direction of the HMMTxD method is the full extension for long-term tracking. The current HMMTxD (mainly the detector) is able to re-detect the object and therefore run in long-term tracking scenarios. However, its power to adapt to object appearance change is limited by the generalization power (or invariance) of features used in the detection part of the framework. To truly enable the long-term tracking capabilities, the method needs to incorporate a technique for long-term sustainable (i.e. drift-free) appearance model learning. In the case of HMMTxD, it is required to update the feature based detector in a way that it remains very precise but adapts to the varying object appearance. This is known to be a very difficult task that receiving high interest from computer vision community, and yet, it is far from being ”reasonably solved”.

The current trends in visual object tracking are diverging from the presented approaches. The benchmark (and also real world) objects of interest vary widely and are complex (e.g. deformable or articulated), which shifts the priorities in tracking such that the discriminative ability to distinguish the object and a background are more crucial than ever. Recent state-of-the-art methods are mostly a discriminative correlation or deep neural network based approaches. Their success lies in their high discriminative power, which is shown to be more valuable in tracking benchmarks (e.g. VOT or OTB) than precise motion estimation or processing speed.

The Visual Object Tracking (VOT) benchmark and methodology is still improving thanks to colleagues and continuing effort of the VOT community. There are several aspects of VOT which are planned to be improved in the future, i) the dataset (ground truth quality, more diverse sequences), ii) different types of experiments and more robust and expressive evaluation criteria and iii) the evaluation toolkit (support for other programming languages, faster processing, more plotting and visualization capabilities). In our effort to improving these parts of VOT, we are mainly limited by the time and human power since this project is mostly self-sponsored.

Appendix A Resumé in Czech language

V této práci navrhujeme dvě nové metody pro krátkodobé sledování objektů: Flock of Trackers (FoT) a Scale-Adaptive Mean-Shift (ASMS), a systém pro spojení několika metod sledování objektu s detektorem. Práce také přispívá k řešení problému vyhodnocování metod pro sledování objektů v rámci Visual Object Tracking (VOT) iniciativy.

Flock of Trackers (FoT) je metoda, která rozdělí objekt zájmu na rovnoměrně velké části. Pro každou tuto část FoT spočítá korespondenci přes optický tok a odhadne její spolehlivost. Spolehlivé korespondence jsou použity pro robustní odhad geometrické transformace objektu pomocí RANSAC algoritmu, který umožňuje odhadovat komplexní rigidní transformace, např. Afinní transformaci. ASMS metoda využívá gradientní optimalizaci k iterativnímu posunu vyhledávacího okna do pozice, která minimalizuje vzdálenost modelu extrahovaného z vyhledávacího okna k modelu objektu. ASMS navrhuje teoretickou modifikaci Mean-Shift algoritmu, která je zaměřená na jeden z hlavních nedostatků tohoto algoritmu, t.j. pevná velikost vyhledávacího okna neboli velikost objektu. ASMS dále představuje způsob využití informace z okolí objektu k redukci chyb ve sledování objektu způsobených rušivými elementy v jeho okolí.

Abychom využili rozdílných předností předchozích metod, navrhujeme nový systém HMMTxD, který umožňuje spojit více metod pro sledování objektů společně s detektorem učeným za běhu algoritmu. Tento systém využívá skrytý Markovův model (HMM) k učení toho, jak jednotlivé sledovací metody fungují za pomoci řídce "anotovaných" dat z detektoru a konfidence odhadnuté danými metodami. Během sledování umožňuje HMM vypočítat pro každý sledovací algoritmus pravděpodobnost, že funguje správně vzhledem k doposud naučenému HMM a současné konfidenci odhadnuté metodou. Tento způsob kombinace zmírňuje nevýhody jednotlivých sledovacích metod, jelikož HMMTxD se sám učí, které sledovací metody fungují dobře, a vypíná zbylé metody v průběhu sledování.

Všechny nově představené metody byly rozsáhle vyhodnoceny na několika veřejně dostupných datových sadách a dosáhly excelentních výsledků v různých kritériích. FoT dosáhl nejlepších výsledků ve VOT2013, kde skončil na druhém místě. ASMS byla jedna z nejlepších metod v VOT2015 a byla vybrána jako nejlepší v poměru kvality sledování objektů a rychlosti běhu algoritmu. Systém HMMTxD demonstroval výborné výsledky ve více vyhodnoceních (VOT2014, VOT2015 a OTB).

Práce také poskytuje přehled o vyhodnocovací metodice Visual Object Tracking (VOT) a přispívá k jejímu rozvoji. VOT poskytuje nástroje pro objektivní porovnání metod sledování objektů napříč publikacemi, což je stěžejní pro rozvoj algoritmů v dlouhodobém časovém horizontu. VOT kolektiv každoročně organizuje semináře na hlavních konferencích počítačového vidění, kde se diskutují pokroky v oblasti sledování objektů a kam mohou autoři sledovacích algoritmů posílat své nové metody a soutěžit tak mezi sebou.

Appendix B Author’s Publications

Journal Papers

[VNM16] T. Vojíř, J. Nosková, and J. Matas. Online Adaptive Hidden Markov Model for Multi-Tracker Fusion. *Computer Vision and Image Understanding*, Special issue on Visual Tracking volume 153, pages 109–119, 2016. 10

[KML^{+}16] M. Kristan, J. Matas, A. Leonardis, T. Vojíř, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin. A Novel Performance Evaluation Methodology for Single-Target Trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 38, pages 2137–2155, 2016. 10

[VNM14] T. Vojíř, J. Nosková, and J. Matas. Robust scale-adaptive mean-shift for tracking. *Pattern Recognition Letters*, volume 49, pages 250–258, 2014. 9

Conference Papers

[LVC^{+}17] A. Lukezic, T. Vojíř, L. Čehovin, J. Matas, M. Kristan. Discriminative Correlation Filter with Channel and Spatial Reliability. *Computer Vision and Pattern Recognition (CVPR)*, 2017 (Accepted to). 15

[VNM14] T. Vojíř, J. Nosková, and J. Matas. Robust scale-adaptive mean-shift for tracking. *Scandinavian Conferences on Image Analysis (SCIA)*, Springer Berlin Heidelberg Image Analysis volume 7944, pages 652–663, 2013. **Best paper award.** 9, 15

[VM14] T. Vojíř, and J. Matas. The Enhanced Flock of Trackers. *Registration and Recognition in Images and Videos - Studies in Computational Intelligence*, volume 532, pages 113–136, Springer 2014. 9

[CVT^{+}12] C. Caraffi, T. Vojíř, J. Trefny, J. Sochman, and J. Matas. A System for Real-time Detection and Tracking of Vehicles from a Single Car-mounted Camera. *The 15th IEEE Intelligent Transportation Systems Conference*, pages 975–982, 2012. 7, 9, 105

[VM11] T. Vojíř, and J. Matas. Robustifying the Flock of Trackers. *CVWW ’11: Proceedings of the 16th Computer Vision Winter Workshop*, pages 91–97, 2011. 9

Visual Object Tracking Workshop Papers

[KML$^+$16] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, L. Čehovin, T. Vojíř, G. Fernandez, G. Hager, A. Lukezic, and R. Pflugfelder. The Visual Object Tracking VOT2016 Challenge Results. *Computer Vision – ECCV 2016 Workshops. ECCV 2016. Lecture Notes in Computer Science*, vol 9914, 2016.

[FBH$^+$16] M. Felsberg, M. Kristan, J. Matas, A. Leonardis, G. Hager, A. Berg, A. Eldesokey, J. Ahlberg L. Čehovin, T. Vojíř, A. Lukezic, G. Fernandez, and R. Pflugfelder The Thermal Infrared Visual Object Tracking VOT-TIR2016 Challenge Results. *Computer Vision – ECCV 2016 Workshops. ECCV 2016. Lecture Notes in Computer Science*, vol 9914, 2016.

[KML$^+$15] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Čehovin, G. Fernandez, T. Vojíř, G. Hager, G. Nebehay, and R. Pflugfelder. The Visual Object Tracking VOT2015 Challenge Results. *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2015. 10

[FBH$^+$15] M. Felsberg, A. Berg, G. Hager, J. Ahlberg, M. Kristan, J. Matas, A. Leonardis, L. Čehovin, G. Fernandez, T. Vojíř, G. Nebehay, and R. Pflugfelder The Thermal Infrared Visual Object Tracking VOT-TIR2015 Challenge Results. *The IEEE International Conference on Computer Vision (ICCV) Workshops*, pp. 1-23, 2015. 10

[KPL$^+$14] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojíř, and G. Fernandez. The Visual Object Tracking VOT2014 challenge results. *European Conference on Computer Vision (ECCV) Workshops*, pages 191–217, 2014. 10

[KPL$^+$13] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Čehovin, G. Nebehay, F. Gustavo, and T. Vojíř. The Visual Object Tracking VOT2013 challenge results. *The IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 98–111, 2013. 10

Bibliography

[ARS06] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2006. 19

[Avi04] S. Avidan. Support vector tracking. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, Aug 2004. 19

[Avi07] S. Avidan. Ensemble tracking. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, Feb 2007. 19

[BBDL10] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2544–2550. IEEE, 2010. 23, 48

[BDB09] D. S. Bolme, B. A. Draper, and J. R. Beveridge. Average of synthetic exact filters. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2105–2112, June 2009. 23

[BDS12] F. Bousetouane, L. Dib, and H. Snoussi. Improved mean shift integrating texture and color features for robust real time object tracking. *The Visual Computer*, 29(3):155–170, 2012. 19

[BPS14] C. Bailer, A. Pagani, and D. Stricker. A superior tracking approach: Building a strong tracker through fusion. In *European Conference on Computer Vision*, Lecture Notes in Computer Science, 2014. 86, 94

[BPSW70] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 02 1970. 89, 90, 91

[Bra98] G. R. Bradski. Computer Vision Face Tracking For Use in a Perceptual User Interface. *Intel Technology Journal*, 1998. 70

[Bre01] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 21, 22

[BS08] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *Journal on Image and Video Processing*, 2008:1:1–1:10, jan 2008. 12

[BYB09] B. Babenko, M. Yang, and S. Belongie. Visual Tracking with Online Multiple Instance Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 20

[BYB11] B. Babenko, M. H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011. 14

[CCCR12] P. Carvalho, J. S. Cardoso, and L. Corte-Real. Filling the gap in quality assessment of video object tracking. *Image and Vision Computing*, 30(9):630–640, 2012. 14

[CFH09] K. Chaudhuri, Y. Freund, and D. J. Hsu. A parameter-free hedging algorithm. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 297–305. Curran Associates, Inc., 2009. 26

[Che95] Y. Cheng. Mean shift, mode seeking, and clustering. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790 –799, aug 1995. 18

[CHT15] Z. Chen, Z. Hong, and D. Tao. An experimental survey on correlation filter-based tracking. *CoRR*, abs/1509.05520, 2015. 23

[ČKL11] L. Čehovin, M. Kristan, and A. Leonardis. An adaptive coupled-layer visual model for robust visual tracking. In *The IEEE International Conference on Computer Vision*, 2011. 77

[ČKL13] L. Čehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4):941–953, apr 2013. 19, 85

[ČKL14] L. Čehovin, M. Kristan, and A. Leonardis. Is my new tracker really better than yours? In *IEEE WACV2014*, 2014. 10, 13, 15, 27, 30

[CKS97] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997. 23

[ČLK15] L. Čehovin, A. s. Leonardis, and M. Kristan. Visual object tracking performance measures revisited. *arXiv:1502.05803 [cs.CV]*, 2015. 13, 15, 27

[CMK03] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. *Pattern Recognition*, 2003. 52

[Col03] R. T. Collins. Mean-shift blob tracking through scale space. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 234–240. IEEE Computer Society, 2003. 70

[Cre06] D. Cremers. Dynamical statistical shape priors for level set-based tracking. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1262–1273, Aug 2006. 23

[CRM00] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2000. 7, 9, 18, 48, 70, 71, 72, 73, 77, 78

[CST00] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000. 19

[CTWS02] D. Cremers, F. Tischhäuser, J. Weickert, and C. Schnörr. Diffusion snakes: Introducing statistical shape knowledge into the mumford-shah functional. *International Journal of Computer Vision*, 50(3):295–313, 2002. 23

[DDS$^+$09] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. 25

[Dem06] J. Demšar. Statistical comparisons of classifiers over multiple datasets. *Journal of Machine Learning Research*, 7:1–30, 2006. 28, 29

[DG13] S. Duffner and C. Garcia. Pixeltrack: A fast adaptive algorithm for tracking non-rigid objects. In *The IEEE International Conference on Computer Vision*, pages 2480–2487, Dec 2013. 22

[DHSKF14] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014. 24

[DHSKF15] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4310–4318, 2015. 24, 25

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, series B*, 39(1):1–38, 1977. 90, 92

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *The IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893 vol. 1, June 2005. 24

[DVM11] T. B. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1177–1184, June 2011. 19, 21

[EEVG$^+$14] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge - a retrospective. *The International Journal of Computer Vision*, 2014. 13, 28

[FB81] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 1981. 21, 52

[FD07] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007. 32

[FH75] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory*, 21(1):32 – 40, jan 1975. 70

[FS97] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997. 19

[FSW12] J. Fan, X. Shen, and Y. Wu. Scribble tracker: a matting-based approach for robust tracking. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(8):1633–1644, August 2012. 19, 22

[GDDM14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 26

[GJP$^+$12] N. Goyette, P. M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection.net: A new change detection benchmark dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2012. 28

[GLB08] H. Grabner, C. Leistner, and H. Bischof. *Semi-supervised On-Line Boosting for Robust Tracking*, pages 234–247. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. 20

[GMVGCne] H. Grabner, J. Matas, L. Van Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *The IEEE Conference on Computer Vision and Pattern Recognition*, June. 85

[GN04] A. K. Gupta and S. Nadarajah. *Handbook of Beta Distribution and Its Applications*. Statistics: A Series of Textbooks and Monographs. CRC Press, 2004, 2004. 88, 91

[GRB11] M. Godec, P. M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. In *International Conference on Computer Vision*, 2011. 22, 85

[HCMB12] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *proceedings of the European Conference on Computer Vision*, 2012. 24, 48

[HCMB15] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. 24, 32, 94, 95

[HH80] P. Hall and C. C. Heyde. *Martingale limit theory and its application.* Probability and mathematical statistics. Academic Press, 1980. 102

[HST11] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *International Conference on Computer Vision*, pages 263–270, Nov 2011. 15, 16, 19, 20, 43, 77, 99

[HYKH15] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *Proceedings of the 32nd International Conference on Machine Learning, 2015, Lille, France, 6-11 July 2015*, 2015. 26

[KGBN$^+$15] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. Icdar 2015 competition on robust reading. In *Proceedings of the 2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, ICDAR '15, pages 1156–1160, Washington, DC, USA, 2015. IEEE Computer Society. 7

[KK11] B. Karasulu and S. Korukoglu. A software for performance evaluation and comparison of people detection and tracking methods in video processing. *Multimedia Tools and Applications*, 55(3):677–723, 2011. 13

[KKLP10] M. Kristan, S. Kovacic, A. Leonardis, and J. Pers. A two-stage dynamic model for visual tracking. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 40(6):1505–1520, Dec 2010. 12

[KL09] 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 *The IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 12, 19, 85

[KM09] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)*, Orlando, October 2009. 3

[KML$^+$15] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Čehovin, G. Fernandez, T. Vojíř, G. Hager, G. Nebehay, and R. Pflugfelder. The visual object tracking vot2015 challenge results. In *The IEEE International Conference on Computer Vision Workshops*, December 2015. 15, 25, 27, 32, 42, 43, 70, 99

[KML$^+$16] M. Kristan, J. Matas, A. Leonardis, T. Vojíř, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin. A novel performance evaluation methodology for single-target trackers. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. 6, 14, 15, 17, 27, 32, 34, 35, 37, 87

[KMM10a] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. *The IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 77

[KMM10b] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-Backward Error: Automatic Detection of Tracking Failures. *The IEEE International Conference on Pattern Recognition*, 2010. 9, 18, 19, 21, 49, 51, 52, 54, 62, 63

[KMM12] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012. 13, 15, 16, 19, 21, 85, 86, 99

[KPL$^+$13] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Čehovin, G. Nebehay, G. Fernandez, and T. Vojíř. The visual object tracking vot2013 challenge results. In *The IEEE International Conference on Computer Vision Workshops*, June 2013. 27, 30, 49, 97, 98, 99

[KPL$^+$14] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojíř, and G. F. e. al. The visual object tracking vot2014 challenge results. In *The European Conference on Computer Vision Workshops*, 2014. 27, 29, 30, 32, 38, 99, 100

[KPPK05] M. Kristan, J. Pers, M. Perse, and S. Kovacic. Bayes spectral entropy-based measure of camera focus. In *Computer Vision Winter Workshop*, pages 155–164, February 2005. 31

[KPPK09] M. Kristan, J. Perš, M. Perše, and S. Kovačič. Closed-world tracking of multiple interacting targets for indoor-sports applications. *Computer Vision and Image Understanding*, 113(5):598 – 611, 2009. Computer Vision Based Analysis in Sport Environments. 12

[KPSK14] M. Kristan, J. Perš, V. Sulic, and S. Kovacic. A graphical model for rapid obstacle image-map estimation from unmanned surface vehicles. In *Asian Conference on Computer Vision. Accepted, to be published.*, 2014. 93

[KSFC10] D. A. Klein, D. Schulz, S. Frintrop, and A. B. Cremers. Adaptive real-time video-tracking for arbitrary objects. In *Intelligent Robots and Systems*, pages 772–777, Oct 2010. 80

[KT04] M. Kölsch and M. Turk. Fast 2d hand tracking with flocks of features and multi-cue integration. In *The IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2004. 18, 19, 49, 52

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988. 23

[LHJ$^+$07] D. Liang, Q. Huang, S. Jiang, H. Yao, and W. Gao. Mean-shift blob tracking with adaptive feature selection and scale adaptation. In *International Conference Image Processing*, 2007. 18, 71

[LHMB16] K. Lebeda, S. Hadfield, J. Matas, and R. Bowden. Texture-independent long-term tracking using virtual corners. *IEEE Transactions on Image Processing*, 25(1):359–371, Jan 2016. 13, 19, 21

[LK81] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision (ijcai). In *The International Joint Conference on Artificial Intelligence*, 1981. 17, 48, 49, 50, 56, 95

[Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, nov 2004. 21, 22, 92

[LSS11] H. Li, C. Shen, and Q. Shi. Real-time visual tracking using compressive sensing. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1305–1312. IEEE, 2011. 13

[LZ15] Y. Li and J. Zhu. *A Scale Adaptive Kernel Correlation Filter Tracker with Feature Integration*, pages 254–265. Springer International Publishing, Cham, 2015. 24

[MC11] E. Maggio and A. Cavallaro. *Video tracking: theory and practice*. John Wiley & Sons, 2011. 5

[MHYY15] C. Ma, J. B. Huang, X. Yang, and M. H. Yang. Hierarchical convolutional features for visual tracking. In *The IEEE International Conference on Computer Vision*, pages 3074–3082, Dec 2015. 25

[MS89] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, pages 577–685, 1989. 23

[MVKC87] A. Mahalanobis, B. V. K. Vijaya Kumar, and D. Casasent. Minimum average correlation energy filters. *Appl. Opt.*, 26(17):3633–3640, Sep 1987. 23

[Nav11] W. Navidi. *Statistics for Engineers and Scientists*. McGraw-Hill, 2011. 29

[NC13] T. Nawaz and A. Cavallaro. A protocol for evaluating video trackers under real-world conditions. *The IEEE Transactions on Image Processing*, 22(4):1354–1361, 2013. 14, 15

[NH15] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. *CoRR*, abs/1510.07945, 2015. 25

[NZZW09] J. Ning, L. Zhang, D. Zhang, and C. Wu. Robust object tracking using joint color-texture histogram. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(07):1245–1263, 2009. 19

[NZZW12a] J. Ning, L. Zhang, D. Zhang, and C. Wu. Robust mean-shift tracking with corrected background-weighted histogram. *Computer Vision, IET*, 6(1):62 –69, january 2012. 18, 78, 79

[NZZW12b] J. Ning, L. Zhang, D. Zhang, and C. Wu. Scale and orientation adaptive mean shift tracking. *Computer Vision, IET*, 6(1):52 –61, january 2012. 18, 70, 77

[Ort12] R. Ortiz. Freak: Fast retina keypoint. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517, Washington, DC, USA, 2012. 93

[OS88] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12 – 49, 1988. 23

[PB13] F. Pernici and A. D. Bimbo. Object tracking by oversampling local features. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints):1, 2013. 85, 92

[PD00] N. Paragios and R. Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):266–280, Mar 2000. 23

[PDB14] F. Pernici and A. Del Bimbo. Object tracking by oversampling local features. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2538–2551, 2014. 21

[PP06] J. Pu and N. Peng. Adaptive kernel based tracking using mean-shift. In *International conference on Image Analysis and Recognition*, ICIAR'06, pages 394–403, 2006. 18, 70

[PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992. 24

[QZQ$^+$16] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, and J. L. M. Yang. Hedged deep tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 26

[Rab89] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989. 89, 91, 92

[Rey87] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH*, 1987. 18, 49, 57

[RKB04] C. Rother, V. Kolmogorov, and A. Blake. ”grabcut”: Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH ’04, pages 309–314, New York, NY, USA, 2004. ACM. 22

[RLLY08] D. A. Ross, J. Lim, R. Lin, and M. Yang. Incremental learning for robust visual tracking. *The International Journal of Computer Vision*, 2008. 12

[RRKB11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *International Conference on Computer Vision*, ICCV ’11, pages 2564–2571, Washington, DC, USA, 2011. 92

[RVTY07] Y. Rathi, N. Vaswani, A. Tannenbaum, and A. Yezzi. Tracking deforming objects using particle filtering for geometric active contours. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1470–1475, Aug 2007. 23

[SCC$^+$13a] A. Smeulder, D. Chu, R. Cucchiara, S. Calderara, A. Deghan, and M. Shah. Visual tracking: an experimental survey. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, nov 2013. 46, 85

[SCC$^+$13b] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Deghan, and M. Shah. Visual Tracking: an Experimental Survey. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. 13, 15, 31, 34, 35, 36, 37

[SK05] Y. Shi and W. C. Karl. Real-time tracking using level sets. In *The IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 34–41 vol. 2, June 2005. 23

[SLS$^+$10] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST Parallel Robust Online Simple Tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, 2010. 19, 21, 86

[SM05] J. Sochman and J. Matas. Waldboost - learning for time constrained sequential detection. In *The IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 150–156 vol. 2, June 2005. 19

[SNHY08] S. M. Shahed Nejhum, J. Ho, and M. Yang. Visual tracking with histograms and articulating blocks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 19

[SS02] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive computation and machine learning. MIT Press, 2002. 24

[ST94] J. Shi and C. Tomasi. Good features to track. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, Jun 1994. 17, 49, 52

[SZ14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 25

[TK91] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method - part 3 detection and tracking of point features. Technical Report CMU-CS-91-132, Computer Science Department, Pittsburgh, PA, April 1991. 17

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *The IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511–I–518 vol.1, 2001. 19

[VM11] T. Vojíř and J. Matas. Robustifying the flock of trackers. In *The Computer Vision Winter Workshop*, 2011. 69

[VM14] T. Vojíř and J. Matas. The enhanced flock of trackers. In *Registration and Recognition in Images and Videos*, volume 532 of *Studies in Computational Intelligence*, pages 113–136. Springer Berlin Heidelberg, January 2014. 32, 94, 95

[VNM13] T. Vojíř, J. Noskova, and J. Matas. Robust scale-adaptive mean-shift for tracking. In *Image Analysis*, volume 7944 of *Lecture Notes in Computer Science*, pages 652–663. Springer Berlin Heidelberg, 2013. 31, 32, 94, 95

[WLY13] Y. Wu, J. Lim, and M. Yang. Online object tracking: A benchmark. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2411–2418, June 2013. 9, 12, 13, 14, 15, 31, 34, 35, 36, 37, 46, 96, 98

[WLY15] Y. Wu, J. Lim, and M. H. Yang. Object tracking benchmark. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, Sept 2015. 15, 46

[WOWL15] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *The IEEE International Conference on Computer Vision*, pages 3119–3127, Dec 2015. 26

[WY14] N. Wang and D. Yeung. Ensemble-based tracking: Aggregating crowdsourced structured time series data. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning*, pages 1107–1115, 2014. 86, 97

[XSL13] J. Xiao, R. Stolkin, and A. Leonardis. An enhanced adaptive coupled-layer lgtracker++. In *The IEEE International Conference on Computer Vision Workshops*, pages 137–144, Dec 2013. 99

[YDD05] C. Yang, R. Duraiswami, and L. Davis. Efficient mean-shift tracking via a new similarity measure. In *The IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 176 – 183 vol. 1, june 2005. 18, 70

[YJS06] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys, 2006*, dec 2006. 14, 85

[YKY12] J. H. Yoon, D. Y. Kim, and K. Yoon. Visual tracking via adaptive tracker selection with multiple features. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part IV*, ECCV'12, pages 28–41, 2012. 87

[YYFL15] Y. Yuan, H. Yang, Y. Fang, and W. Lin. Visual object tracking by structure complexity coefficients. *Multimedia, IEEE Transactions on*, 17(8):1125–1136, Aug 2015. 86

[ZKR08] C. Zhao, A. Knight, and I. Reid. Target tracking using mean-shift and affine structure. In *The IEEE International Conference on Pattern Recognition*, pages 1–5, 2008. 18, 71

[ZL10] X. Zhou and Y. Lu. Abrupt motion tracking via adaptive stochastic approximation Monte Carlo sampling. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1847–1854, June 2010. 85

[ZLQ08] C. Zhao, Z. Liu, and H. Qiao. Face tracking via block texture feature based mean shift. In *Natural Computation, 2008. ICNC '08. Fourth International Conference on*, volume 3, pages 190–194, Oct 2008. 19

[ZLY95] S. C. Zhu, T. S. Lee, and A. L. Yuille. Region competition: unifying snakes, region growing, energy/bayes/mdl for multi-band image segmentation. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 416–423, Jun 1995. 23

[ZLY12] W. Zhong, H. Lu, and M. Yang. Robust object tracking via sparsity-based collaborative model. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1838–1845, June 2012. 16

[ZMS14] J. Zhang, S. Ma, and S. Sclaroff. MEEM: robust tracking via multiple experts using entropy minimization. In *Proc. of the European Conference on Computer Vision*, 2014. 19, 20, 86, 97, 99

[ZZY12] K. Zhang, L. Zhang, and M. Yang. Real-time compressive tracking. In *European Conference on Computer Vision*, ECCV'12, pages 864–877, 2012. 77