UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques, Sciences et Technologies de l'Information**

Arrêté ministériel : 7 août 2006

Présentée par

Yang HUA

Thèse dirigée par **Cordelia SCHMID**
et codirigée par **Karteek ALAHARI**

préparée au sein **Microsoft Research - Inria Joint Centre, Inria**
et de **l'école doctorale MSTII : Mathématiques, Sciences et Technologies de l'Information, Informatique**

Vers un Suivi robuste d'objets visuels: sélection de propositions et traitement des occlusions

Towards Robust Visual Object Tracking: Proposal Selection and Occlusion Reasoning

Thèse soutenue publiquement le **10 juin 2016**,
devant le jury composé de :

Dr. Patrick PÉREZ
Technicolor Research & Innovation, Rennes, France, Rapporteur

Pr. Deva RAMANAN
Carnegie Mellon University, Pittsburgh, PA, USA, Rapporteur

Pr. Jiří MATAS
Czech Technical University, Prague, Czech Republic, Examinateur

Dr. Florent PERRONNIN
Facebook AI Research (FAIR), Paris, France, Président

Dr. Cordelia SCHMID
Inria Grenoble - Rhône-Alpes, France, Directeur de thèse

Dr. Karteek ALAHARI
Inria Grenoble - Rhône-Alpes, France, Co-Encadrant de thèse

Abstract

In this dissertation we address the problem of visual object tracking, wherein the goal is to localize an object and determine its trajectory over time. In particular, we focus on challenging scenarios where the object undergoes significant transformations, becomes occluded or leaves the field of view. To this end, we propose two robust methods which learn a model for the object of interest and update it, to reflect its changes over time.

Our first method addresses the tracking problem in the context of objects undergoing severe geometric transformations, such as rotation, change in scale. We present a novel proposal-selection algorithm, which extends the traditional discriminative tracking-by-detection approach. This method proceeds in two stages – proposal followed by selection. In the proposal stage, we compute a candidate pool that represents the potential locations of the object by robustly estimating the geometric transformations. The best proposal is then selected from this candidate set to localize the object precisely using multiple appearance and motion cues.

Second, we consider the problem of model update in visual tracking, i.e., determining when to update the model of the target, which may become occluded or leave the field of view. To address this, we use motion cues to identify the state of the object in a principled way, and update the model only when the object is fully visible. In particular, we utilize long-term trajectories in combination with a graph-cut based technique to estimate parts of the objects that are visible.

We have evaluated both our approaches extensively on several tracking benchmarks, notably, recent online tracking benchmark and the visual object tracking challenge datasets. Both our approaches compare favorably to the state of the art and show significant improvement over several other recent trackers. Specifically, our submission to the visual object tracking challenge organized in 2015 was the winner in one of the competitions.

Keywords. Visual object tracking • Tracking-by-detection • Proposal-selection tracking • Long-term tracking • Occlusion reasoning • Video analysis • Computer vision

Résumé

Cette dissertation traite du problème du suivi d'objets visuels, dont le but est de localiser un objet et de déterminer sa trajectoire au cours du temps. En particulier, nous nous concentrons sur les scénarios difficiles, dans lesquels les objets subissent d'importantes déformations et occlusions, ou quittent le champs de vision. A cette fin, nous proposons deux méthodes robustes qui apprennent un modèle pour l'objet d'intérêt et le mettent à jour, afin de refléter ses changements au cours du temps.

Notre première méthode traite du problème du suivi dans le cas où les objets subissent d'importantes transformations géométriques comme une rotation ou un changement d'échelle. Nous présentons un nouvel algorithme de sélection de propositions, qui étend l'approche traditionnelle de "suivi par détection". Cette méthode procède en deux étapes : proposition puis sélection. Dans l'étape de proposition, nous construisons un ensemble de candidats qui représente les localisations potentielles de l'objet en estimant de manière robuste les transformations géométriques. La meilleure proposition est ensuite sélectionnée parmi cet ensemble de candidats pour précisément localiser l'objet en utilisant des indices d'apparence et de mouvement.

Dans un second temps, nous traitons du problème de la mise à jour de modèles dans le suivi visuel, c'est-à-dire de déterminer quand il est besoin de mettre à jour le modèle de la cible, lequel peut subir une occlusion, ou quitter le champs de vision. Pour résoudre cela, nous utilisons des indices de mouvement pour identifier l'état d'un objet de manière automatique et nous mettons à jour le modèle uniquement lorsque l'objet est entièrement visible. En particulier, nous utilisons des trajectoires à long terme ainsi qu'une technique basée sur la coup de graphes pour estimer les parties de l'objet qui sont visibles.

Nous avons évalué nos deux approches de manière étendue sur différents bancs d'essai de suivi, en particulier sur le récent banc d'essai de suivi en ligne et le jeu de donnée du concours de suivi visuel. Nos deux approches se comparent favorablement à l'état de l'art et font montre d'améliorations significatives par rapport à plusieurs autres récents suiveurs. Notre soumission au concours de suivi d'objets visuels de 2015 a par ailleurs remporté l'une de ces compétitions.

Mots-clés. Suivi d'objet visuel • Suivi par détection • Suivi par proposition-sélection • Suivi à long terme • Traitement des occlusions • Analyse vidéo • Vision par ordinateur

Acknowledgements

Foremost, I would like to thank my advisors Dr. Cordelia Schmid and Dr. Karteek Alahari, for their invaluable guidance and sharing priceless expertise with me without any reservation. In particular, I would like to say how lucky I met Cordelia at CVPR 2012 and later was accepted as one of her students at Inria Grenoble Rhône-Alpes, where is my dream place to pursuit a Ph.D. degree. Furthermore, Cordelia's scientific vision and insight as well as Karteek's elaborate instructions and helps have guided me to make progress consistently, encouraged me to push past my boundaries and cultivated my curiosity and passion for research during all these years.

I would like to express my gratitude to my jury members, Dr. Patrick Pérez, Prof. Deva Ramanan, Prof. Jiří Matas and Dr. Florent Perronnin, who agreed to evaluate my work, gave me interesting comments and asked me insightful questions throughout our interactions. I am also grateful to Microsoft Research and Inria Joint Center in Paris, which partially funded my Ph.D. research. Special thanks to Dr. Laurent Massoulie, who kindly invited me to participate in the "Spring Day" to present and demonstrate my latest progress, and to Martine Thirion and Hélène Bessin-Rousseau, who were patient to help me handle with the administrative work remotely.

This thesis was carried out in the LEAR team (now known as Thoth team), which provides an excellent environment to conduct world class research in Computer Vision and Machine Learning. I would like to thank all the team members (or former members) I interacted with, who were not only colleagues but also true friends: Dr. Zaid Harchaoui, Dr. Matthijs Douze, Dr. Jakob Verbeek, Dr. Julien Mairal, Dr. Jerome Revaud, Dr. Heng Wang, Dr. Anoop Cherian, Dr. Albert Gordo, Dr. Xiaojiang Peng, Dr. Guosheng Hu, Dr. Marco Pedersoli, Dr. Gregory Rogez, Dr. Adrien Gaidon, Dr. Thomas Mensink, Dr. Zeynep Akata, Dr. Gokberk Cinbis, Dr. Dan Oneata, Dr. Danila Potapov, Philippe Weinzaepfel, Federico Pierucci, Shreyas Saxena, Vicky Kalogeiton, Pavel Tokmakov, Nicolas Chesneau, Hongzhou Lin, Daan Wynen, Vladyslav Sydorov, Alberto Bietti, Yuansi Chen and Dexiong Chen. Moreover, many thanks to Nathalie Gillot for her enthusiastic help to make my life easier living in France, to Xavier Martin for his unconditional support to solve my tedious technical issues, to Mattis Paulin for translating the abstract of this thesis into French, and to Dr. Henrique Morimitsu for taking over my work and helping me release the source code.

Last but certainly not least, I cannot show how grateful I am for always receiving endless support and trust from my family, my friends and my former colleagues all over the world. Specifically, this thesis is dedicated to my wife, Dr. Zhengui Xue, for her understanding and company in the past three and half years. Because of her, anything becomes possible and meaningful in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

As one of the most important and challenging research topics in computer vision, *visual object tracking* aims at "the problem of estimating the trajectory of an object in the image plane as it moves around a scene" [Yilmaz et al., 2006]. It also plays a fundamental role in many high-level computer vision tasks, e.g., action recognition and video understanding. Visual object tracking has been successfully applied in a wide range of real-world applications, especially in visual surveillance (e.g., W4 system [Haritaoglu et al., 2000] and VSAM project [Collins et al., 2000] for monitoring human activities), monitoring traffic (e.g., traffic flow monitoring [Coifman et al., 1998], traffic accident detection [Tai et al., 2004], and pedestrian counting [Masoud and Papanikolopoulos, 2001]), video compression (e.g., MPEG4 standard [Sikora, 1997]), and human-computer interaction (e.g., hand gesture recognition [Pavlovic et al., 1997] and mobile video conferencing [Paschalakis and Bober, 2004]).

Despite extensive studies during the past several decades, tracking objects under unconstrained scenarios is still a complex and difficult task due to many practical challenges, including illumination change, occlusion, deformable objects, noise corruption, viewpoint variations, and motion blur (cf. Figure 1.1). Furthermore, the proliferation of video data and new data-acquisition devices has stimulated a great deal of interest in building more intelligent tracking algorithms.

Firstly, video has become one of the most popular visual media for

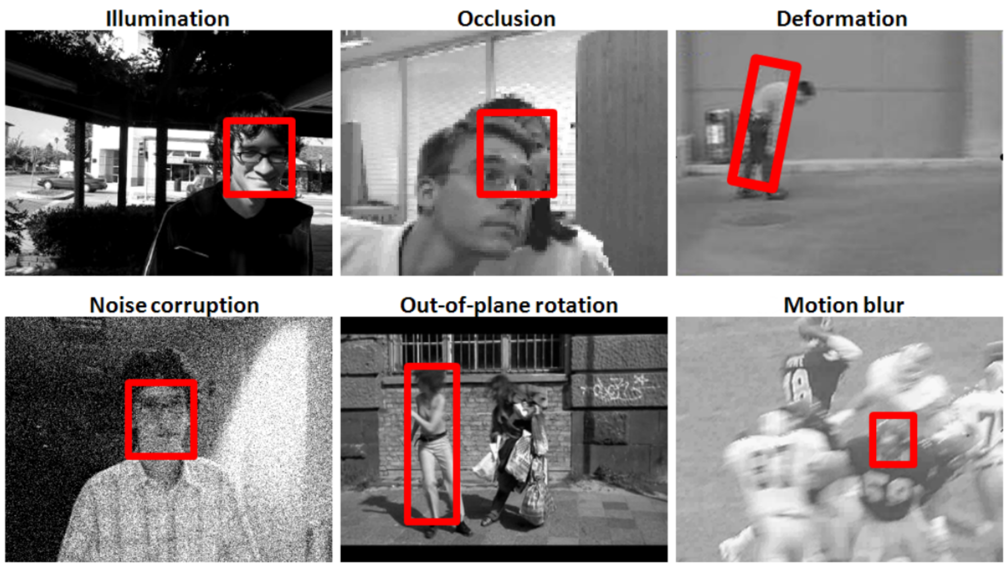


Figure 1.1 – Illustration of challenging appearance changes resulting in challenging scenarios for visual object tracking [Li et al., 2013b]. Image courtesy of [Li et al., 2013b].

communication, surveillance and entertainment in recent years. Owing to the fast growth of the consumer electronics industry, now everyone can shoot videos to record anything that they are interested in, with inexpensive video cameras or even mobile phones. Subsequently, the amount of video data is booming, e.g., 300 hours of new videos are uploaded to YouTube every minute.[1] Another aspect to these videos is that they are recorded by millions of contributors with varied devices and thus vary in quality and content. In addition to the standard RGB format, many videos may also come from various recording apparatuses, such as thermal infrared (TIR) videos from thermal infrared cameras and RGBD videos from Microsoft Kinect. In order to automatically or semi-automatically analyze, index and manage such video data, more *general* visual object tracking algorithms are required. An ideal general tracker can handle multi-domain videos containing arbitrary content with minimum changes in the parameters of its framework.

Secondly, several devices, usually seen in sci-fi movies, made their debut, and have been gradually coming to fruition in the past few years, such as augmented reality glasses, autonomous cars and drones (cf. Figure 1.2).

1. https://www.youtube.com/yt/press/statistics.html

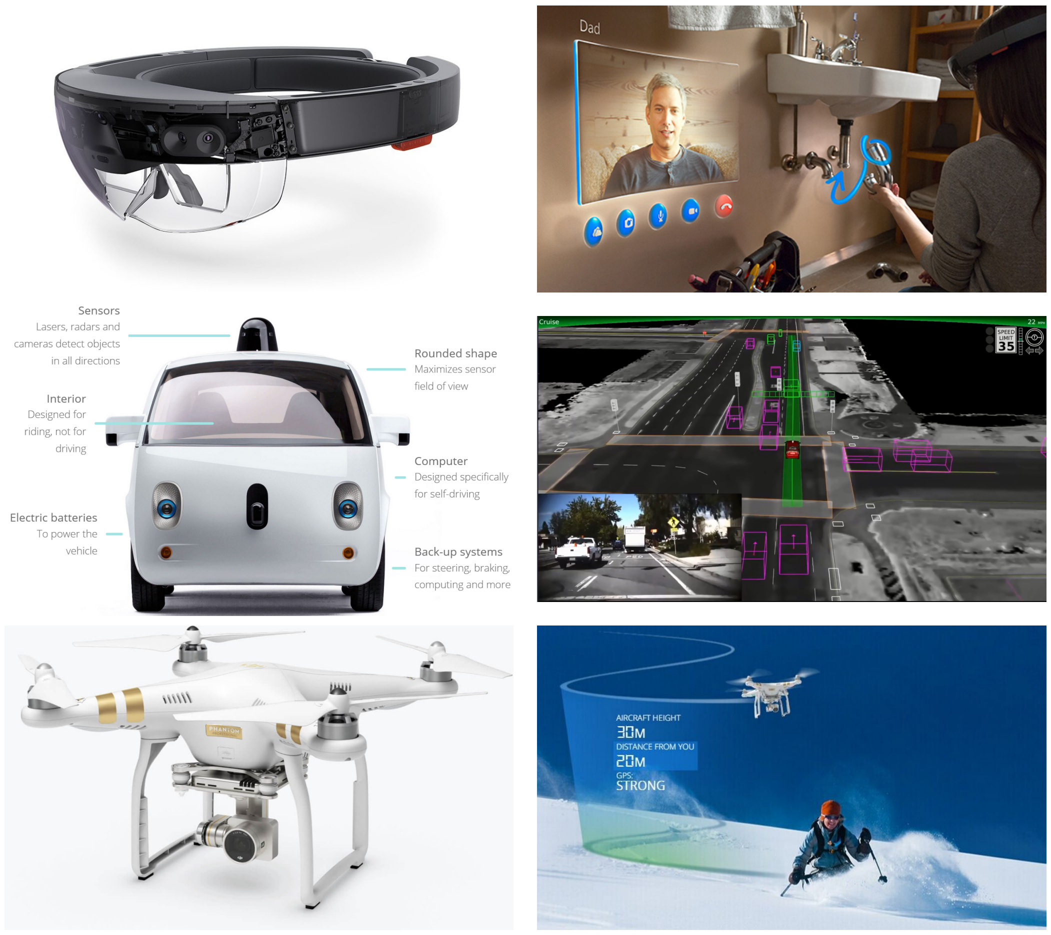

Dad
Sensors
Lasers, radars and cameras detect objects in all directions
Interior
Designed for riding, not for driving
Electric batteries
To power the vehicle
Rounded shape
Maximizes sensor field of view
Computer
Designed specifically for self-driving
Back-up systems
For steering, braking, computing and more
Cruise
SPEED LIMIT 35
AIRCRAFT HEIGHT
30M
DISTANCE FROM YOU
20M
GPS:
STRONG

Figure 1.2 – Illustration of recent devices. First row: Microsoft Hololens (https://www.microsoft.com/microsoft-hololens); Second row: Google autonomous car (https://www.google.com/selfdrivingcar); Third row: DJI drone (http://www.dji.com/product).

Based on these emerging devices, visual object tracking algorithms can certainly be one of the fundamental components to create many new applications in real-world scenarios. To this end, more robust tracking algorithms, which can cope with objects changing in appearance, becoming occluded and leaving the field of view, are highly sought after.

1.1 Goals

The goal of our work is to devise a robust visual object tracker for diverse real-world scenarios. In other words, we aim to track an arbitrary object defined at the beginning of a video in an unconstrained environment, where the object can undergo significant transformations, become occluded or leave the field of view.

Despite several decades of research, tracking arbitrary objects in an uncontrolled setting still poses enormous challenges. One of the main challenges is *generality*. In the early days, many widely used applications only focused on person or car tracking (a.k.a. *model-specific* tracking), which significantly reduced the difficulties of designing tracking systems, but also limited their adaptability to other scenarios. Later, *model-free* tracking has attracted significant attention because of its capability of handling different situations. Nevertheless, many model-free trackers were only evaluated on a limited number of video sequences, which is not sufficient to prove their generality. In this context, we aim to further improve the generality of model-free tracking, building on the most successful discriminative tracking-by-detection framework, and conduct evaluations on diverse model-free tracking datasets from different video domains.

Another challenge is the well-known *template update* problem [Matthews et al., 2004], widely existing in real-world situations. In the case of discriminative tracking-by-detection framework, if a classifier (i.e., template) is only trained with the ground truth annotation in the first frame, it is less prone to drifting caused by inaccurate model updates. However, this robust approach does not take appearance changes into account and cannot adapt to the evolution of the target object. In contrast, a highly adaptive classifier (e.g., classifier updated every frame) can gather essential training samples to capture the diverse appearance changes, but easily result in drifting due to improper updates. Unlike most existing heuristic approaches, we aim to tackle this dilemma in a principled way, based on motion cues, in order to predict the true state of an object, resulting in robust model update.

1.2 Context

As the predecessor of visual object tracking, target tracking first appeared in the 1950s [Wax, 1955]. It mainly focused on the application of radio detection and ranging (RADAR) signal detection and automated tracking. The goal of RADAR tracking was to establish correlations between point-wise "blips". Several algorithms, such as the Kalman filter [Kalman, 1960], were first designed to solve the RADAR tracking problem, and were later successfully applied to visual object tracking, other tasks in computer vision and even other disciplines, e.g., control theory.

Point tracking methods treat the task as a data association problem, i.e., finding correspondences in data across frames, similar to RADAR signal tracking. Point tracking techniques became popular in the late 1980s and remained so until the early 1990s [Sethi and Jain, 1987, Salari and Sethi, 1990]. It is worth noting that point tracking has strong connections with motion-based computer vision problems, e.g., optical flow estimation [Lucas and Kanade, 1981], and thus they share similar approaches. For instance, there are three general methods for computing optical flow [Derpanis, 2006], including matching methods, differential methods and frequency-based methods, which are also popular methodologies for point tracking.

During the 1990s, the interest of visual tracking shifted from simple points to higher-level objects, when appearance-based trackers emerged. Here, tracked objects are represented by pixel-wise templates (e.g., Kanade-Lucas-Tomasi tracker [Lucas and Kanade, 1981, Tomasi and Kanade, 1991]), outer contours of objects (e.g., snakes tracker [Kass et al., 1988]), blobs (e.g., Pfinder tracker [Wren et al., 1997]) or histograms (e.g., mean-shift tracker [Comaniciu et al., 2000]). Many widely used tracking applications only focused on specific objects, i.e., model-specific tracking, such as human faces and bodies [Wren et al., 1997], which are favorable for certain real-world scenarios.

Since 2000, several efforts have been made to develop more general visual object tracking approaches. As a result, model-free tracking, which is designed to find trajectories of arbitrary objects in video sequences, has became popular. The tracking-by-detection approach has surfaced as one of the successful paradigms for model-free tracking in the context of *short-term tracking*, which usually assumes that the target object is always present in the video sequence. This approach has also been used as one of the basic components to build more complex tracking systems. For example, the tracking-learning-detection (TLD) tracker [Kalal et al., 2012] has leveraged it to handle the *long-term tracking* problem, which addresses the model update issue when the target can become occluded or leave the field of view.

There has been rapid progress in visual object tracking more recently, in particular, since 2010. For instance, discriminative correlation filter-based trackers [Bolme et al., 2010, Henriques et al., 2015, Danelljan et al., 2014a, 2015] have received more attention due to their state-of-the-art performance and computational efficiency. Deep learning-based approaches that have dominated many computer vision problems have also shown excellent performance on many tracking datasets [Li et al., 2014, Wang et al., 2015a, Nam and Han, 2016]. In parallel, several visual object tracking datasets and challenges [Wu et al., 2013, Smeulders et al., 2014, Kristan et al., 2013, 2014, 2015, Felsberg et al., 2015, Li et al., 2015a] have been developed, which provide unified evaluation platforms and can accelerate the development of more advanced trackers.

In this context, our work fits into the model-free visual object tracking scenario. In particular, we design and develop general and robust trackers in an unconstrained environment for both short-term and long-term tracking problems.

1.3 Contributions

In this thesis, we propose two novel approaches for model-free visual object tracking under challenge scenarios, where the object can undergo significant transformations, become occluded or leave the field of view. Our contributions are summarized in the following paragraphs.

A general tracking framework that handles diverse situations for short-term tracking. Building upon the successful discriminative tracking-by-detection framework, we present a general *proposal-selection* tracking approach (cf. Figure 1.3). We first compute a candidate pool that consists of the potential locations of the target. Besides proposals from a common tracking-by-detection framework (cf. Figure 1.3-(c)), we calculate additional proposals by robustly estimating the geometry transformation (i.e., similarity transformation) that the target is likely to have undergone (cf. Figure 1.3-(d)). We then utilize a two-phase selection strategy to determine the tracking result. In the first phase, we select the best candidate merely based on detection scores (cf. Figure 1.3-(e)). If detection scores are inconclusive to make a decision, we move to the second phase — select the best candidate by comparing edge scores [Zitnick and Dollár, 2014] computed with edge responses (cf. Figure 1.3-(f)) [Dollár and Zitnick, 2013] and motion boundaries (cf. Figure 1.3-(g)) [Weinzaepfel et al., 2015].

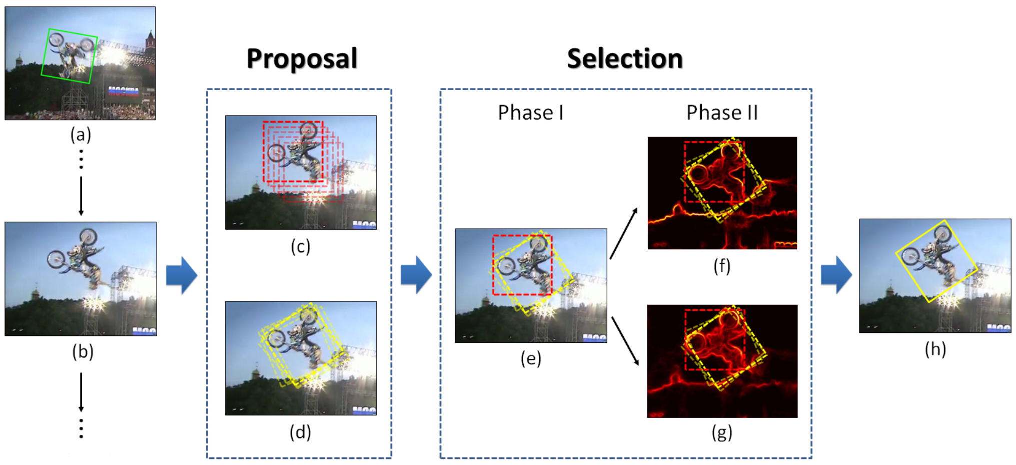


Figure 1.3 – The overall framework of proposal and selection tracker: (a) Initialization; (b) A sample frame in which the object is to be tracked; (c) Tracking-by-detection proposals; (d) Geometry proposals; (e) Selection phase I: Detection score, selection phase II: Edgebox score from (f) edge responses, and (g) motion boundaries; (h) The selected tracking result used for updating the model.

Extensive experimental results show that the proposed tracker achieves the top performance on diverse datasets with fixed parameters. This work was published in ICCV 2015 [Hua et al., 2015] and is presented in Chapter 3.

A robust method for model update in the context of long-term tracking. Model update is one of the most critical components for long-term tracking, where the target object may be occluded or leave the field of view. We propose a principled framework to handle the problem of model update, relying on motion cues. In particular, we utilize long-term trajectories in combination with graph-cut based track labeling to identify the state of the object (cf. Figure 1.4), e.g., no / partial / full occlusion or leaving the field of view. The decision of model update is then made according to the state of the object in each frame. We evaluate our approach on multiple video datasets and show comparable performance with several state-of-the-art trackers. This work was published in ECCV 2014 [Hua et al., 2014] and is discussed in Chapter 4.

Participation in VOT2015 and VOT-TIR2015. We present the result of our participation in the visual object tracking (VOT) 2015 and the thermal

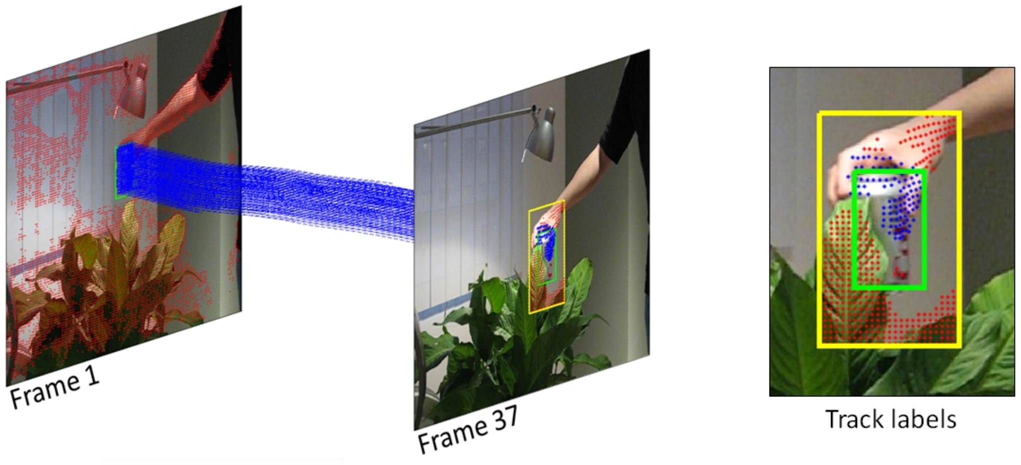


Figure 1.4 – *Left*: Illustration of long-term tracks spanning multiple frames. The yellow box shows the search region used to compute the bounding box most likely to contain the object (green box). *Right*: Close-up of the track labels, where blue denotes an object tracks and red represents background tracks.

infrared visual object tracking (VOT-TIR) 2015 challenges in Appendix A. VOT and VOT-TIR compare short-term model-free single-object trackers, and serve as the de factor state-of-the-art evaluation platform for visual object tracking. VOT focuses on natural RGB video sequences with rotated rectangular ground truth boxes, while VOT-TIR consists of thermal infrared video sequences with axis-aligned ground truth boxes. Our submission is based on the proposal-selection tracker described in Chapter 3. In order to show the generality of our tracker, we set identical parameters for both the VOT2015 and VOT-TIR2015 challenges. Accordingly to the evaluation results [Kristan et al., 2015, Felsberg et al., 2015], our tracker is the winner chosen from 24 trackers in the VOT-TIR2015 challenge, and ranked sixth among 62 trackers in the VOT2015 challenge.

Chapter 2

Related Work

Contents

In this chapter we present an overview of the literature in visual object tracking. Specifically, we focus on single-camera, single-target, model-free trackers, applied to both short-term and long-term tracking. In Section 2.1, we review several approaches for short-term tracking, namely, tracking-by-detection (§2.1.1), and tracking by matching (§2.1.2), correlation filters (§2.1.3), context information (§2.1.4), fusion (§2.1.5), and deep learning (§2.1.6). We then discuss the requirement of long-term tracking and present an overview of relevant methods in Section 2.2. Finally, in Section 2.3, we introduce popular tracking datasets and their corresponding evaluation methods used in this thesis.

We focus our discussion on the most representative and relevant approaches for this thesis. For other single-camera single-target model-free tracking methods, the interested reader is referred to review papers, such as [Yilmaz et al., 2006], [Cannons, 2008], [Yang et al., 2011] and [Li et al., 2013b]. Visual object tracking with multiple cameras [Wang, 2013] and multiple object tracking [Luo et al., 2014], which are other active research problems, are beyond the scope of this thesis.

2.1 Short-term Tracking

2.1.1 Tracking by Detection

During the past several years, tracking-by-detection has become one of the most successful paradigms for visual object tracking, and has achieved state-of-the-art performance [Grabner et al., 2006, Mei and Ling, 2009, Babenko et al., 2011, Hare et al., 2011, Bai et al., 2013, Gao et al., 2014, Possegger et al., 2015]. This is due, in part, to the success seen by components of object detection algorithms. In particular, the popular algorithms for object detection, such as [Viola and Jones, 2001], [Dalal and Triggs, 2005] and [Felzenszwalb et al., 2010], have inspired many representative tracking algorithms, including [Grabner et al., 2006], [Supancic III and Ramanan, 2013] and [Shu et al., 2012]. The two main components in the tracking-by-detection approach are visual representation and statistical modeling. Like other tasks in computer vision, visual representation also plays a fundamental role in visual object tracking. Most features for visual representation can be applied in visual object tracking, such as intensity [Ross et al., 2008], color [Pérez et al., 2002], texture [Avidan, 2007], Haar-like feature [Babenko et al., 2011], LBP [Grabner et al., 2006], HOG [Tang et al., 2007], and deep learning features [Li et al., 2014]. Several popular visual object tracking approaches based on deep learning features will be discussed in §2.1.6.

In this section, we focus on the most representative papers related to statistical modeling, which can be grouped into two categories: generative and discriminative models.

Generative models. Tracking with generative modeling typically focuses on learning a model to represent the target object, and then uses it to find the most similar region in the future frames. For instance, Black and Jepson [1998] learn an offline subspace model to represent relevant objects and Ross et al. [2008] utilize an incremental subspace model to describe the target object to adapt appearance changes. Moreover, matching methods can be applied to estimate the motion of the target object

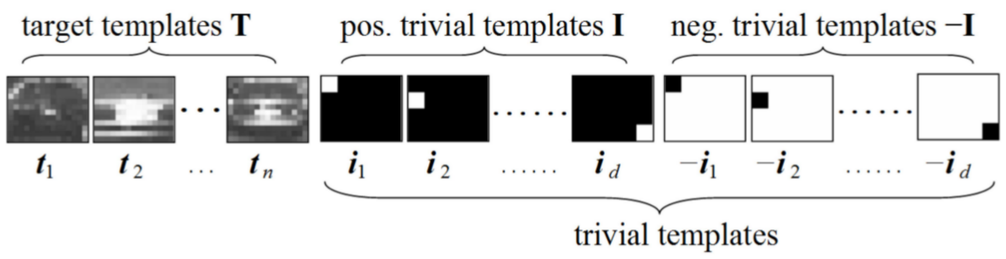


Figure 2.1 – Illustration of templates used for ℓ_1 reconstruction [Mei and Ling, 2009], including target templates and trivial templates. Image courtesy of [Mei and Ling, 2009].

between consecutive frames and then track the object over time. These matching-based tracking approaches will be discussed in §2.1.2.

Sparse coding [Tibshirani, 1996] has also been applied to visual object tracking to determine the target with minimum reconstruction error from the template space. Mei and Ling [2009] proposed a robust tracking method, referred to as L1 tracker, by treating object tracking as a sparse approximation problem. During tracking, target candidates are represented as a sparse linear combination of the template set including both target templates (obtained from previous frames) and trivial templates. Similar to [Wright et al., 2009], target templates correspond to the normal appearance of object, while trivial templates are applied to handle occlusion and other challenging issues, shown as in Figure 2.1. In an ideal case, a target candidate can be efficiently represented by the target templates. Therefore, almost all of the weights (i.e., coefficients) of trivial templates tend to be zeros. In the case of challenging situations (e.g., occlusion), a limited number of trivial coefficients may be activated, but the whole coefficient vector (containing all of coefficients) remains sparse. Then the target candidate with minimum reconstruction error is selected as tracking result. Although L1 tracker performs well on several challenging scenarios, it requires high computational resources due to numerous calculations for ℓ_1 minimization. In order to address this issue, Mei et al. [2011] extended the L1 tracker with minimum error bound re-sampling. Most insignificant samples are filtered out by minimum error bound before solving the computational expensive ℓ_1 minimization function. Thus this new strategy can improve the tracking speed without sacrificing accuracy.

More recently, multi-task sparse learning [Zhang et al., 2012b] and low-rank sparse learning [Zhang et al., 2012a] were applied to extend the L1 tracker, and utilize the underlying relationships between reconstruction samples. A more complete and detailed review on sparse coding based tracking algorithms can be found in [Zhang et al., 2013].

Discriminative models. Despite the success, generative modeling usually faces difficulties to describe the target object without considering background information, especially when the appearance of target object changes dramatically and/or the background is cluttered. On the contrary, instead of trying to build a complete model to represent the target object, discriminative modeling treats object tracking as a classification problem, in order to distinguish the target object from the background. Therefore, it is usually more robust to complex scenarios by explicitly modeling background as negative training samples. Trackers based on discriminative modeling have evolved rapidly and dominated almost all datasets [Wu et al., 2013, Smeulders et al., 2014, Kristan et al., 2016] in recent years.

An early approach of discriminative modeling was proposed by Avidan [2004], where a support vector machine (SVM) classifier and optical-flow-based tracking are combined to track vehicles over long video sequences. However, due to constraints on computational resources, the SVM classifier is learned offline and is not updated in an online manner. The ensemble-based tracking approach [Avidan, 2007] adopts the Adaboost algorithm and breaks the complex training phase into a set of simple and easy-to-learn tasks, i.e., weak classifiers, which can be computed online. First, an ensemble of weak classifiers is trained to separate pixels that belong to object from those that belong to background. Then, given a new frame, this ensemble-based classifier is applied to create a confidence map of the pixels. The mean-shift algorithm [Comaniciu et al., 2003] is applied to this confidence map to find the new position of the object, i.e., the peak in the confidence map. Subsequently, the ensemble-based tracker is updated by adding a weak classifier learned from the tracking result. Later, Collins et al. [2005] and Grabner et al. [2006] also applied similar online boosting framework for real-time object tracking but trained weak classifiers based on rectangle bounding box instead of pixels used in [Avidan, 2007].

Online boosting-based tracking algorithms, however, suffer from drifting, because the weak classifiers updated in each frame are not robust to accumulated tracking errors. To address this, many improvements are proposed from different perspectives. Semi-supervised learning [Chapelle et al., 2006] is one way of dealing with errors in tracking, which affect the

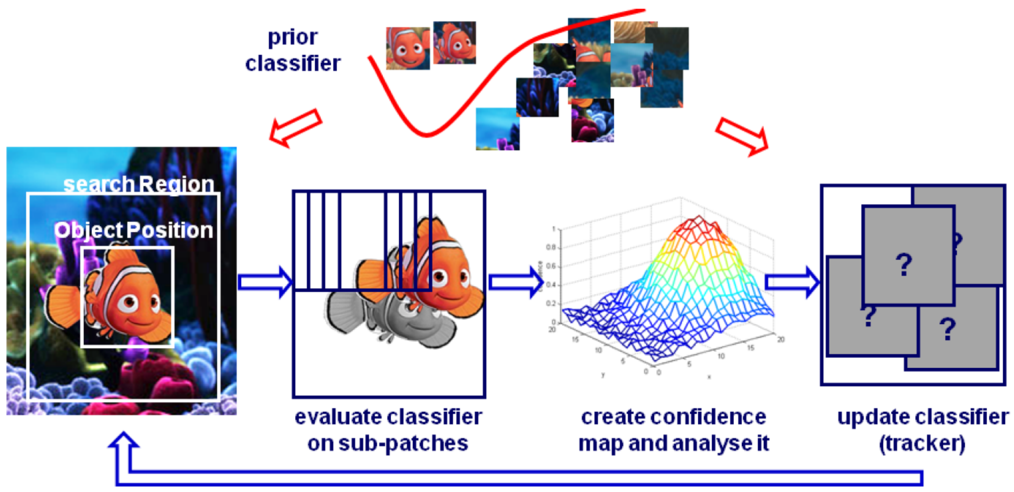

prior classifier
search Region
Object Position
evaluate classifier on sub-patches
create confidence map and analyse it
update classifier (tracker)
?
?
?
?

Figure 2.2 – Overview of SemiBoost tracker [Grabner et al., 2008]. Given a prior classifier learned from the first frame and the position of the object at time t, the classifier is evaluated at several candidate locations in a search region in frame $t+1$. The confidence map from this is analyzed to estimate the most likely location of the object. Finally the classifier is updated in an unsupervised manner using randomly selected patches. Image courtesy of [Grabner et al., 2008].

training set for updating the model. Inspired by co-training [Blum and Mitchell, 1998], Tang et al. [2007] proposed a semi-supervised learning algorithm, where several classifiers learned online were used. In this framework, the tracking results from one classifier are utilized to update other classifier(s). Grabner et al. [2008] proposed a SemiBoost [Mallapragada et al., 2009, Leistner et al., 2008] tracker to explore the continuum between a prior classifier learned from the initial frame and all samples as unlabeled for updating in the following frames, as shown in Figure 2.2. In other words, SemiBoost tracker is updated on-line in an unsupervised manner that can adapt (or drift) to new appearance, but it has the possibility to recover from significant drift, as it retains the prior classifier during the tracking procedure.

Babenko et al. [2011] proposed to apply multiple instance learning (MIL) [Dietterich et al., 1997] in visual object tracking, in order to allow the classifier to select from a number of potential positive samples according to its current state. Compared to other traditional methods (cf. Figure 2.3), e.g., selecting single imperfect positive sample or using several noisy positive samples, MIL tracker treats the training samples as “bags”. A bag is considered as positive if it contains at least one positive instance,

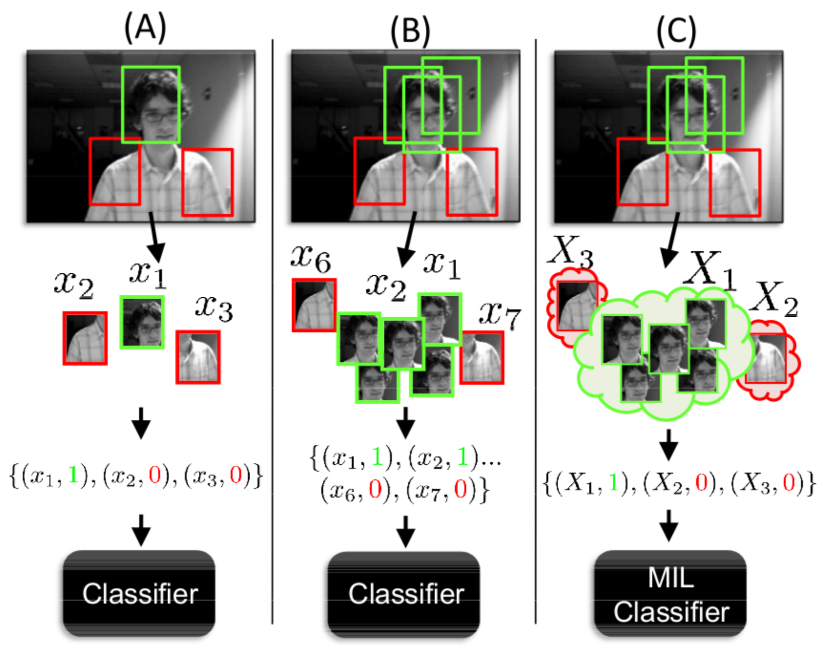

(A)
(B)
(C)
x_2 x_1 x_3
x_6 x_2 x_1 x_7
X_3 X_1 X_2
$\{(x_1,1),(x_2,0),(x_3,0)\}$
$\{(x_1,1),(x_2,1)\ldots(x_6,0),(x_7,0)\}$
$\{(X_1,1),(X_2,0),(X_3,0)\}$
Classifier
Classifier
MIL Classifier

Figure 2.3 – Comparison of three different ways to update a discriminative appearance model [Babenko et al., 2011]. (A) Using a single positive image patch to update a traditional discriminative classifier. Due to accumulated tracking errors, a single positive patch may not capture the object sufficiently. (B) Using several positive patches to update a traditional discriminative classifier. It can introduce noisy training samples and degrade the classifier. (C) Using one positive bag consisting of several image patches to update a MIL classifier. Image courtesy of [Babenko et al., 2011].

otherwise the bag is set to negative. Therefore, MIL tracker keeps sufficient training samples and tolerates labeling noise when updates its model.

Another successful approach is to localize the target object precisely in every frame in order not to accumulate tracking errors. Hare et al. [2011] proposed Struck, which is based on structured output SVM framework [Tsochantaridis et al., 2005]. It addresses the limitation of previous trackers, such as [Grabner et al., 2006], [Grabner et al., 2008] and [Babenko et al., 2011], which separate target localization (i.e., labeling samples within a search region) and model update into two distinct steps, shown as in Figure 2.4. This dichotomy introduces additional errors from the labeling step to the model update step, because the sample chosen by the classifier may not correspond to the best estimate of the object location. In contrast, Struck integrates these two steps into one unified structured output

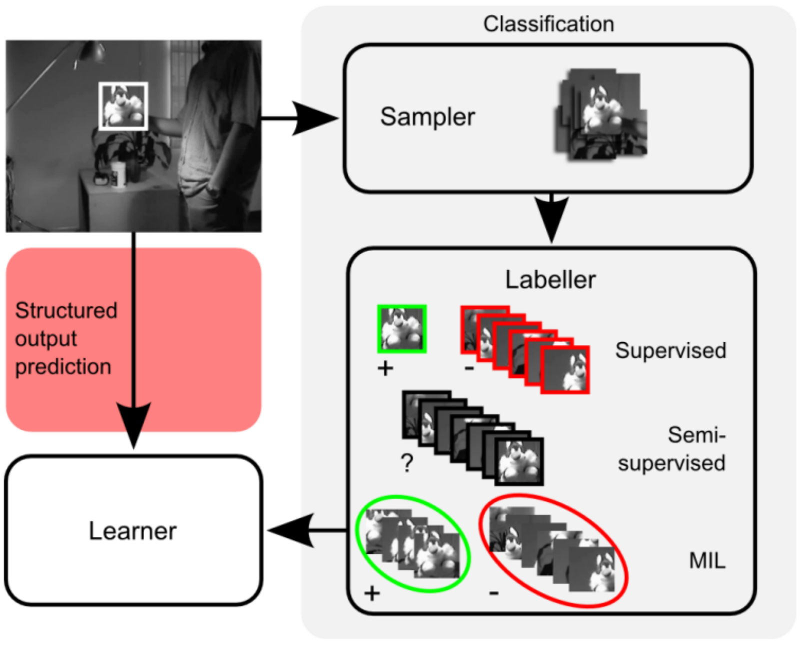


Figure 2.4 – Comparison of adaptive tracking-by-detection paradigms [Hare et al., 2011]. Given the current estimated object location, traditional approaches (shown on the right-hand side) generate a set of samples and, depending on the type of the learner, produce training labels. Struck (left-hand side) avoids this two-step approach, and operates directly on the tracking output. Image courtesy of [Hare et al., 2011].

learning framework (cf. Figure 2.4)

Moreover, discriminative correlation filter-based methods, which conduct pixel-accurate localization in the frequency domain, have evolved and achieved state-of-the-art performance with real-time running speed in recent years. These approaches will be discussed in §2.1.3.

Most discriminative model trackers are limited to a bounding-box representation with a fixed aspect ratio. Thus, they cannot handle highly non-rigid and articulated objects. Godec et al. [2011] proposed a novel framework to overcome this limitation with the use of a Hough forest, shown as in Figure 2.5. First, a Hough-based classification framework is utilized to detect non-rigid objects robustly. Second, the foreground supports (cf. red points shown as in Figure 2.5), detected by the Hough-voting framework, are used to segment the object from the background roughly. The segmentation result delivers a more precise description of the object than a simple bounding box representation, which decreases the noise for online model update, in particular for deformable objects.

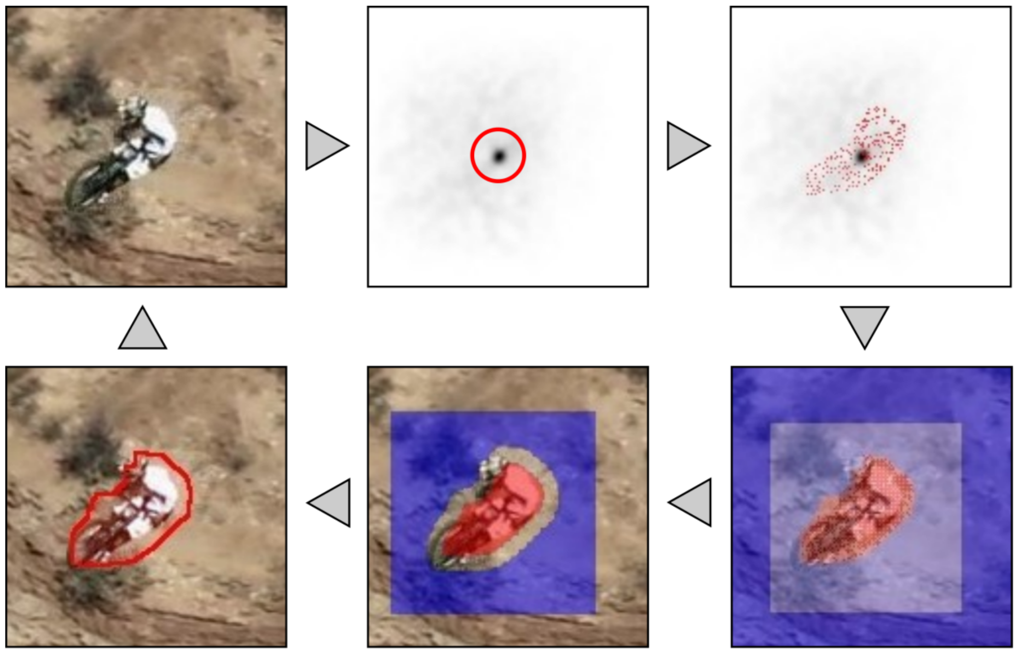


Figure 2.5 – Hough-based tracking [Godec et al., 2011], shown in clockwise order from top-left image: Input frame, Hough-based object detection result, back-projection and supporting image positions, guided segmentation, robust updating, and tracking result. (Red points and regions: foreground support and foreground segments; Blue regions: background segments). Image courtesy of [Godec et al., 2011].

Later, Duffner and Garcia [2013] extended this framework to do pixel-level Hough-based classification, which further improves the tracking performance especially on small regions.

2.1.2 Tracking by Matching

Matching the representation of a target object between two consecutive frames is a natural way to estimate its motion and track it over time. This was a dominant tracking approach in the early days, due to its fairly good performance, simple structure, and low computational requirement.

The simplest approach for object tracking in this paradigm is template matching, where the image content within bounding box in the first frame serves as an initial template. Tracking is then performed by maximizing a similarity function, e.g., normalized cross-correlation (NCC), between this initial patch and patches extracted at the candidate locations. In practice, an exhaustive local search for candidate windows around the previous position of the target is performed, and the candidate window with the highest similarity score is selected as the new target location. This approach, referred to as NCC tracker, also acts as one of the important components in more advanced trackers [Santner et al., 2010, Kalal et al., 2012]. Although NCC tracker can be applied in an effective manner with techniques, such as [Lewis, 1995], it remains computationally expensive, especially when the template size or/and the search region is/are large.

As a more effective template-matching approach, Kanade-Lucas-Tomasi (KLT) tracker [Lucas and Kanade, 1981, Tomasi and Kanade, 1991, Baker and Matthews, 2004] considered that the similarity function of template matching should be locally smooth and built upon gradient-based optimization. First, the KLT tracker finds affine-transformed matches between two consecutive frames by means of spatio-temporal derivatives and warping. Then, the new location of the target is determined by mapping its position in the previous frame to the location in the current frame using the estimated affine transformation. This KLT tracker has been improved in several aspects. Shi and Tomasi [1994] proposed a feature selection criterion for applying template matching in order to improve robustness. Since the smoothness assumption may only be valid in small local regions in practice, Bouguet [2001] demonstrated a coarse-to-fine optimization scheme on an image pyramid to deal with larger motion between two consecutive frames. A forward-backward KLT tracker is devised to automatically detect tracking failures in [Kalal et al., 2010].

Template-based matching methods typically lack robustness to track non-rigid objects. Several approaches have emerged to solve this issue. Based on the mean shift algorithm [Fukunaga and Hostetler, 1975] that was originally proposed for data clustering, Comaniciu et al. [2000] designed mean-shift trackers to perform matching with color histograms, which is invariant to changes in shape of targets. In every new frame, the mean shift algorithm is used to determine the location of the target by maximizing a similarity metric. This tracker, however, can be confused by regions with a similar color distribution, due to the lack of spatial information. Therefore, Yang et al. [2005] reintroduced spatial information into the mean-shift framework by defining a new symmetric similarity between kernel density estimates of the template and the target distributions in a joint feature-spatial space [Elgammal et al., 2003]. Similarly, Adam et al. [2006] added spatial information into the histogram-based matching scheme by dividing the target region into fixed fragments. Given a new frame, each candidate window around the previous location of the tar-

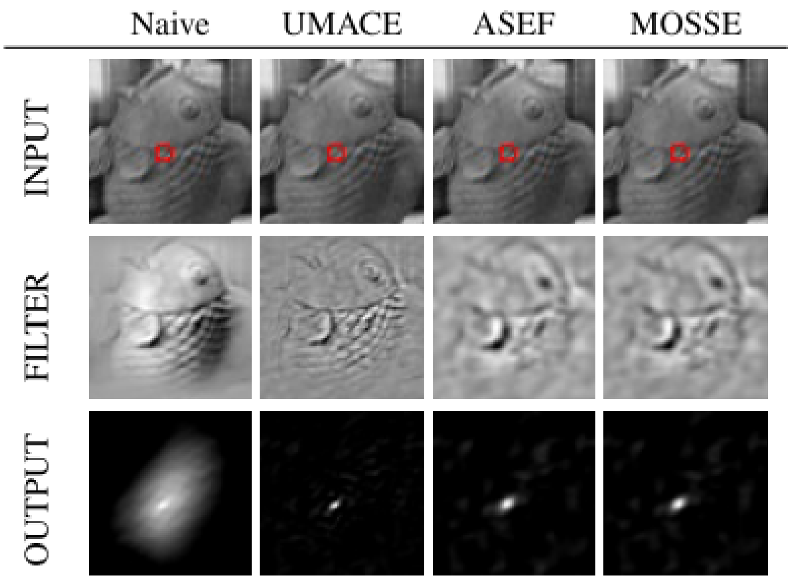


Figure 2.6 – Comparison of outputs using different correlation filters [Bolme et al., 2010]. Top to bottom: input image, filters, and correlation output. The three correlation filters (UMACE, ASEF, MOSSE) produce peaks that are more compact than the one produced by the naive filter. Image courtesy of [Bolme et al., 2010].

gets fragmented. Subsequently, the distance between the corresponding patches regions is calculated by earth mover's distance. The candidate window which contains the top 25% matching patches is then selected as the new target location.

Bayesian filtering-based methods, such as kernel-based Bayesian filtering [Han et al., 2005], Kalman filtering [Cuevas et al., 2005], and particle filtering [Isard and Blake, 1998a, Arulampalam et al., 2002], can be used to extend matching-based methods. These Bayesian filtering-based methods treat tracking as a model estimation problem using measurements of the target in every frame. The estimated model can be the position, scale, geometry transformation, and even 2D/3D shape of the target. The measurement can include color histogram, edges, contours, and other appearance features of the target. Unlike matching-based methods, which rely only on the similarity between template frame and target frame, Bayesian filtering-based methods utilize all the observed frames with emphasis on probabilistic formulation. These methods perform more robust than matching-based methods in real-world applications.

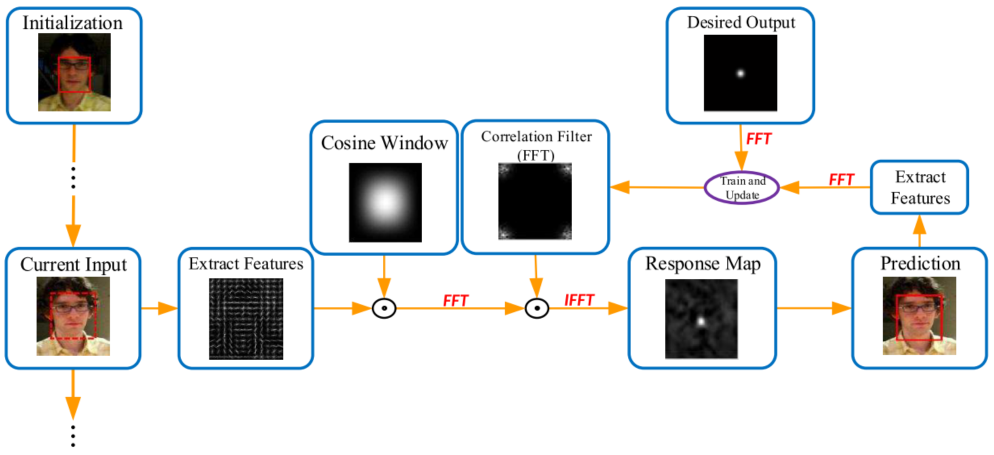
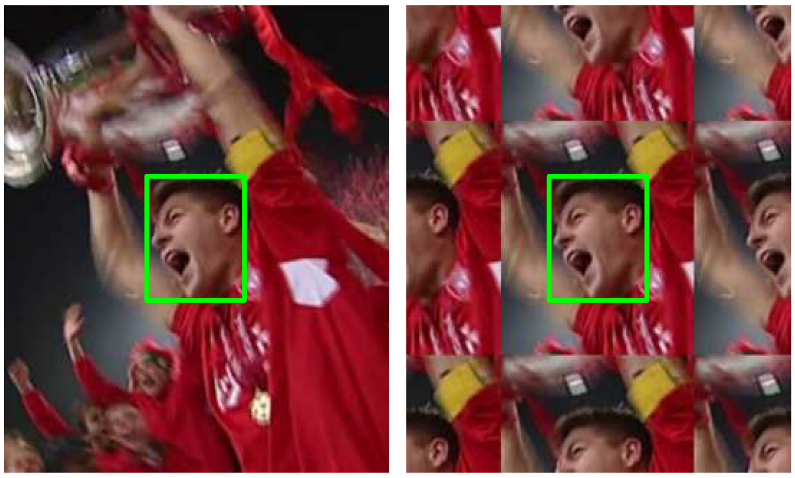


Figure 2.7 – A general flow chart for typical correlation filter-based tracking methods [Chen et al., 2015]. Image courtesy of [Chen et al., 2015].

2.1.3 Correlation Filter Tracking

In recent years, correlation filter-based trackers have received much attention due to their simple structure, state-of-the-art performance and computational efficiency. As a basic operation in digital image processing, a correlation filter is used to find locations in an image that are similar to a pre-defined template. Ideally, a correlation filter produces high responses for a pre-defined template, while generating low responses for background. However, in most real-world applications, the response of a naïve correlation filter for background is relatively high, as shown in the first column of Figure 2.6. Several improvements to correlation filters have been proposed to overcome this issue. They are trained with negative samples to suppress responses for background, while maintaining strong responses for the pre-defined template. This approach has been successfully applied to eye localization [Bolme et al., 2009a], pedestrian detection [Bolme et al., 2009b] and object recognition [Boddeti et al., 2013]. Due to the requirement of a large number of training samples, these methods are infeasible for online visual object tracking.

The minimum output sum of squared error (MOSSE) filter proposed by Bolme et al. [2010], which is a variant of the average of synthetic exact filters (ASEF) [Bolme et al., 2009a], helped to overcome the limitation of previous approaches. Unlike the minimum average correlation energy (MACE) [Mahalanobis et al., 1987] and unconstrained maximum average correlation energy (UMACE) [Mahalanobis et al., 1994] methods, ASEF and MOSSE algorithms are more flexible and suitable for tracking prob-



(a) Original image. (b) Periodicity in correlation filters.

Figure 2.8 – The periodic assumption of correlation filter-based trackers [Danelljan et al., 2015]. Image courtesy of [Danelljan et al., 2015].

lem, as they do not require the target object to present at the center of the image. However, ASEF filters become unstable when trained on a small number of samples because the denominator tends to zero. In contrast, the denominator of MOSSE is the sum of the energy frequencies of training samples, and is more stable numerically. Henriques et al. [2012] proposed an improved approach by kernelizing the correlation filters with a kernel regularized least squares formulation and circulant matrix. This approach, called CSK, shows an excellent performance, while maintaining computational efficiency.

The framework of a typical correlation filter-based tracking method (cf. Figure 2.7) can be summarized as follows. In the first frame, an initial correlation filter is trained based on the ground truth bounding box. For each following frame, various local features are extracted and filtered by a cosine window (cf. Figure 2.7) in order to smooth the boundary effects. Subsequently, a response map is generated efficiently with a fast Fourier transform (FFT). The position with the maximum value in this map is predicted as the new location of the target object. Finally, the new location of the object is used to update the correlation filter. It is worth noting that the correlation filter training and updating procedures are all performed in the frequency domain, which is critical to achieve a high-frame-rate tracker.

Though MOSSE and CSK correlation filter-based trackers have shown excellent performance, they have several limitations, which deserve further investigation. These trackers work directly on the raw pixels of input images, which brings unwanted noise, and limits the tracking performance. Although some preprocessing techniques can be applied to address this, more powerful local features are still required. KCF [Henriques et al., 2015] and DSST [Danelljan et al., 2014a] use HOG feature [Dalal and Triggs, 2005, Felzenszwalb et al., 2010] computed over multiple channels in the Fourier domain. Besides HOG feature, Danelljan et al. [2014b] found color name feature [Van De Weijer et al., 2009] performed well. These two features were used jointly in the SAMF [Li and Zhu, 2014].

MOSSE and CSK trackers also work with a fixed window size, i.e., the window size is determined by the ground truth annotation in the first frame and thus cannot handle scale changes. DSST [Danelljan et al., 2014a] and SAMF [Li and Zhu, 2014] proposed a multi-scale search strategy to estimate the true scale of the target object. In other words, these correlation filter-based trackers are applied on multiple scale, and the window with the highest response is selected as the tracking result.

As shown in Figure 2.8, the periodic extension of training sample, which is generated by a circular sliding window operation, enables efficient training and detection by FFT. However, it also produces unwanted boundary effects, which can cause an inaccurate representation, and more importantly, limit the search region of the tracker and the collection of training samples. Danelljan et al. [2015] proposed spatially regularized discriminative correlation filters (SRDCF) to overcome undesired boundary effects with a spatial regularization component. This allows SRDCF to be applied on, and learned from large image regions.

The methods described above have overcome the intrinsic issues of correlation filter-based tracking. However, it is also worth noting that the computational efficiency of these methods is significantly inferior in comparison to the original correlation filter-based methods. A brief summary of these works is given in Table 2.1. Correlation filter-based trackers have been further extended to address deformable object tracking [Liu et al., 2015, Li et al., 2015b], long-term tracking [Ma et al., 2015b, Hong et al., 2015b], tracking with context information [Zhang et al., 2014b] and tracking with deep learning [Ma et al., 2015a].

2.1.4 Tracking with Context Information

Context information plays an important role among many computer vision tasks, including but not limited to, object classification [Kumar and

Tracker Name Abbreviation	Feature	Speed (fps)	Remarks
MOSSE [Bolme et al., 2010]	Raw pixel	669.0	-
CSK [Henriques et al., 2012]	Raw pixel	320.0	extension of MOSSE
KCF [Henriques et al., 2015]	HOG	172.0	improved CSK
DSST [Danelljan et al., 2014a]	HOG	24.0	based on MOSSE
CN [Danelljan et al., 2014b]	Color name	78.9	based on CSK
SAMF [Li and Zhu, 2014]	HOG & Color name	7.0	based on KCF
SRDCF [Danelljan et al., 2015]	HOG	5.0	based on KCF

Table 2.1 – A summary of correlation filter-based trackers. The computation time, shown as Speed (fps), is taken from the original papers.

Hebert, 2005, Munoz et al., 2009, Song et al., 2011], object detection [Torralba, 2003, Murphy et al., 2003, Hoiem et al., 2008], object segmentation [Shotton et al., 2006, Rabinovich et al., 2007]. Not surprisingly, several works also successfully utilized context information for the visual object tracking task, in particular, for some challenging scenarios.

Yang et al. [2009] proposed a novel tracking framework by integrating context information, called context-aware visual tracking (CAT). In CAT, context information is represented by a set of auxiliary objects, which are easy to track, and have motion consistent with the target. It is worth noting that the auxiliary objects are not required to be semantic objects but informative image regions. There are three main steps in the CAT framework: (i) Data mining techniques are used to choose auxiliary objects automatically by learning their co-occurrence associations, and estimating affine motion models of the target; (ii) The target and auxiliary objects are collaboratively tracked based on a random field model; (iii) Robust fusion is applied to deal with inconsistency among the target and the auxiliary object trackers.

Similar to [Yang et al., 2009], Grabner et al. [2010] introduced supporters which are useful features to predict the target location, even when the target is occluded or leaves the fields of view. These supporters, which need to have motion similar to the target, are maintained by generalized Hough transform to vote and determinate target object location. Extending [Grabner et al., 2010], Dinh et al. [2011] exploited context information with two opposite terms, i.e., supporters and distractors (cf. Figure 2.9), in an online manner. Supporters are local key-points around the target, which co-occur and have motion similar to the target. These supporters can help verify the genuine target. In contrast, distractors are regions

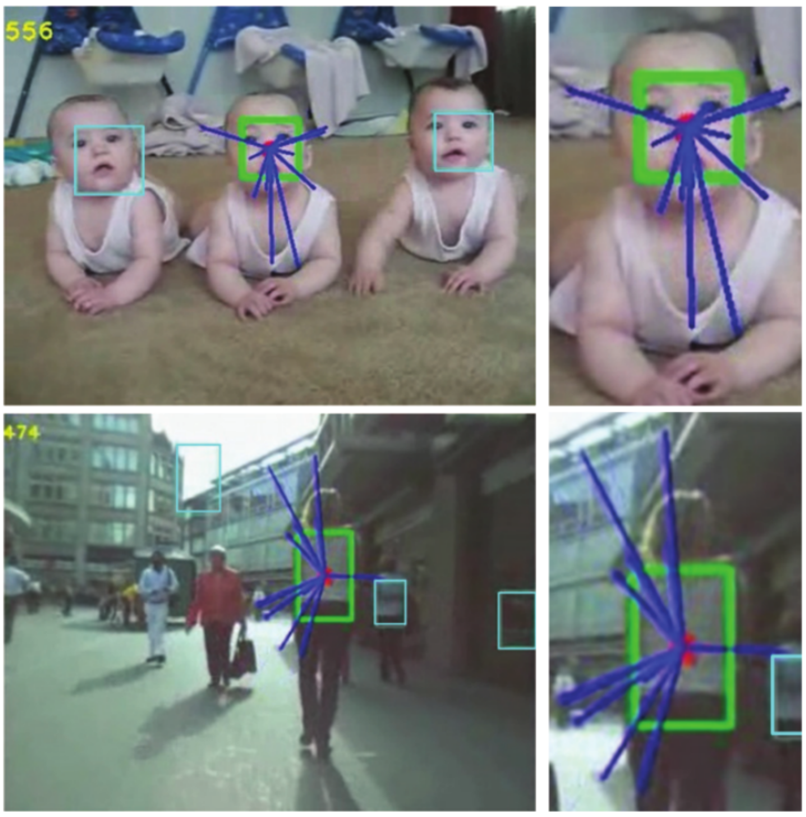


Figure 2.9 – Illustration of context elements, i.e., supporters and distractors [Dinh et al., 2011]. Supporters: the end points of blue lines. Distractors: the bounding boxes in cyan. Image courtesy of [Dinh et al., 2011].

which have similar appearance as the target and consistently co-occur. They are tracked as negative regions in order to avoid confusion with the target.

These initial ideas have evolved in the past few years, and several tracking methods have exploited spatio-temporal context, such as [Wen et al., 2012] and [Zhang et al., 2014b].

2.1.5 Tracking by Fusion

Fusing multi-channel information is another popular approach for several computer vision tasks, including tracking. It is an important strategy to exploit the complementarity of multiple tracking schemes. This can be achieved by fusing either multiple cues or multiple trackers. A tracker with multi-cue fusion combines complementary cues (e.g., features or models from different domains) to tackle different scenarios. Each cue in the fusion framework plays a certain role and cannot be removed or replaced arbitrarily. In contrast, the multi-tracker fusion framework can integrate several existing trackers of different types. In other words, multi-cue fusion happens at the model-level, while multi-tracker fusion is on the output-level.

Multi-cue fusion. Du and Piater [2008] proposed a novel probabilistic approach to integrate multiple cues and exploited the relationship between different cues. Four visual cues from different domains, including color, edge, motion and contours, are used to build separate particle filter-based trackers. A hidden Markov model is then learned to combine the individual cues, to capture not only the temporal evolution of each cue, but also its interaction with others. Inference is performed on this model with sequential auxiliary particle belief propagation to track the target.

A few methods have explicitly handled the trade-off between stability and plasticity. A tracking system requires plasticity for the integration of appearance changes of the target, but also stability in order to prevent it from drafting. To this end, Santner et al. [2010] proposed a parallel robust online simple tracking (PROST) framework, which consists of three trackers selected to cover the entire adaptivity spectrum, as shown in Figure 2.10. For stability, a standard normalized cross-correlation (NCC) template matching tracker, based only on the ground truth in the first frame is applied. At the other extreme, a highly adaptive optical-flow-based mean-shift tracker is utilized to capture frame-to-frame changes. Between the two extremes, an online random forest tracker is learned. These three complementary trackers are organized in a cascaded manner. The adaptive optical-flow-based tracker is set as the main tracker, whose output is verified and can be overruled by the random forest tracker. Furthermore, the NCC tracker is deployed as a superior safe-guard to monitor and prevent the random forest tracker from making incorrect updates.

Lee et al. [2015] employed three Struck-based trackers [Hare et al., 2011] using Haar-like features, color histogram features and illumination invariant features. The fusion framework relies on a multi-hypothesis trajectory analysis algorithm, which extracts geometric similarity, cyclic weight, and appearance similarity from forward and backward trajectories of the three trackers, and their corresponding robustness scores. The forward trajectory of the tracker with the highest score is used to predict the target location.

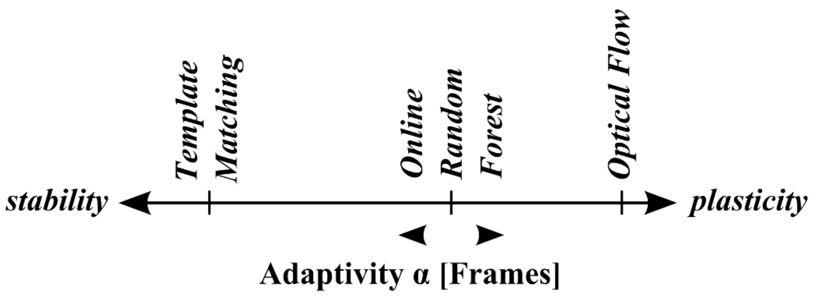


Figure 2.10 – Illustration of the complementary components of PROST [Santner et al., 2010] that have been chosen to cover the entire adaptivity spectrum. The adaptivity of an online tracker is defined as the number of frames it needs to adapt to appearance changes. Image courtesy of [Santner et al., 2010].

Multi-tracker fusion. Leichter et al. [2006] proposed a probabilistic framework for combining many standalone trackers, where each tracker may operate in a different state-space. The principal requirement of this framework is that the outputs of the trackers should be either an explicit probability density function, or a sample-set of it, e.g., from CONDENSATION-based trackers [Isard and Blake, 1998a]. Such a limitation is a big drawback to further improve tracking performance, since most top-performing trackers are not based on a probabilistic framework.

Other fusion frameworks have emerged in order to combine trackers without any constraints. Inspired by joint learning from multiple label sources [Whitehill et al., 2009], Zhong et al. [2014] proposed a novel fusion framework, which can evaluate the performance of multiple individual trackers during the tracking procedure. In other words, in each frame, the best tracker among all the standalone trackers determines the target location. However, this framework merely considers the fusion task as a weakly-supervised binary classification problem, and ignores temporal smoothness. Wang and Yeung [2014] designed a more integrated fusion approach for object tracking based on a factorial hidden Markov model, which generalizes HMM to a distributed state representation. It utilizes a state-space model in a crowdsourcing setting for aggregating structured time series data, i.e., frame-to-frame tracking results, coming from multiple trackers in order to learn jointly the unknown trajectories of the target, and the reliability of each tracker.

Bailer et al. [2014] introduced an energy-based fusion framework that can work with arbitrary trackers, based on attraction fields and trajectory optimization. In order to avoid setting thresholds typically used in a

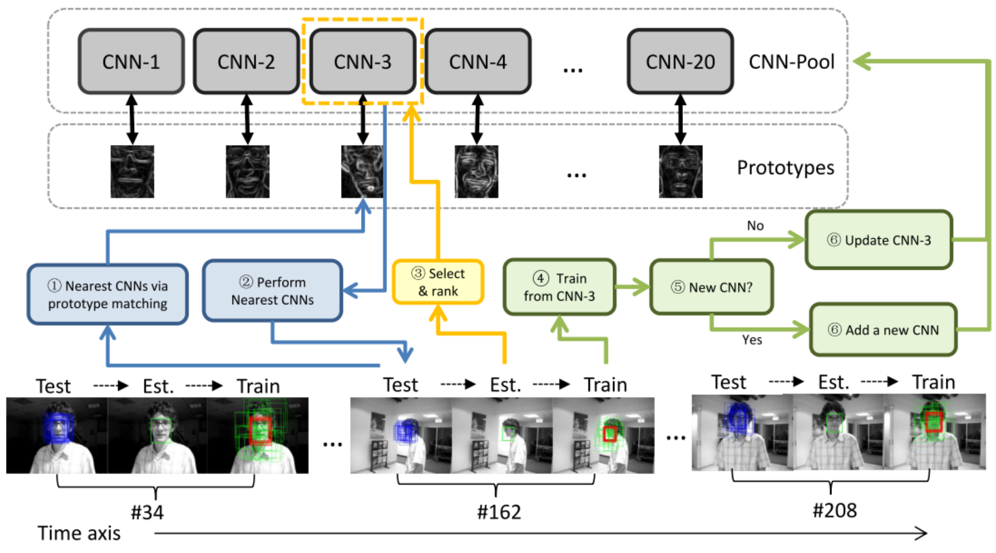


Figure 2.11 – The overall procedure of DeepTrack [Li et al., 2014] for determining the tracking result and updating the corresponding CNN. Image courtesy of [Li et al., 2014].

majority voting approach, attraction fields are employed to find fusion results. Intuitively, the closer a fusion candidate is to the final tracking result the higher the attraction score is, i.e., the stronger it is "attracted" to the target location. Then, the trajectories are optimized with a unified energy function to remove potential discontinuities in the frame-based attractive fields.

Some other fusion-based approaches, either utilize additional training data for specific object tracking (e.g., face and human) [Spengler and Schiele, 2003, Stenger et al., 2009] or integrate information from other domains (e.g., stereo sound or ultra-sound) for certain real-world applications [Perez et al., 2004, Han et al., 2007], and are beyond the scope of this thesis.

2.1.6 Deep Learning for Tracking

In recent years, deep learning based approaches, particularly convolutional neural network (CNN), have achieved great empirical success and dominated many computer vision problems, such as object classification [Krizhevsky et al., 2012], detection [Girshick et al., 2014], and face verification [Taigman et al., 2014]. Applying this paradigm to the visual object tracking problem requires some additional effort, due to the lack of appropriate training data, and the target object evolving over the course

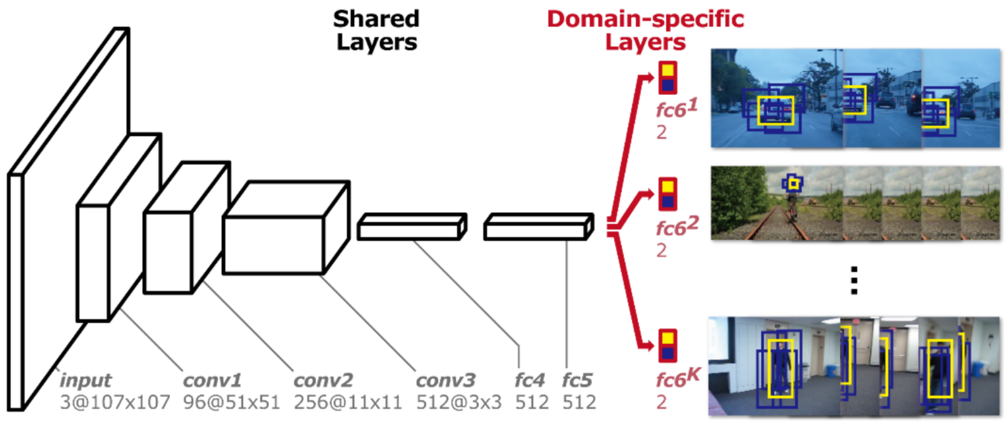


Figure 2.12 – The architecture of multi-domain network [Nam and Han, 2016], which consists of shared layers and K branches of domain-specific layers. Yellow and blue bounding boxes denote the positive and negative samples in each domain, respectively. Image courtesy of [Nam and Han, 2016].

of the video sequence. In general, three main approaches exist so for: (i) Using pre-trained deep learning features as visual representation for object tracking; (ii) Training from scratch or fine-tuning a network in an online manner; (iii) Improving localization accuracy with deep learning networks.

One of the reasons why deep learning has become very popular in computer vision is its representation power compared to traditional hand-crafted features. Motivated by this, deep learning has been exploited as a powerful feature learning tool for visual object tracking in [Wang and Yeung, 2013]. They introduced a deep learning tracker by means of a stacked denoising autoencoder (SDAE). At first, an offline unsupervised image feature extractor is built using the SDAE learned from generic datasets. Then, in the online tracking phase, a sigmoid classification layer is added to the encoder part of SDAE learned offline to form a tracking-by-detection tracker. Following the feature extraction framework in [Wang and Yeung, 2013], Zhou et al. [2014] revisited the ensemble-based online boosting framework with new deep learning features for tracking. A group of binary deep neural network (DNN) classifiers is built by adding a sigmoid layer to the pre-trained SDAE. This group of classifiers is then integrated in an online boosting framework to determine the location of the target.

In order to better adapt to the appearance of the target defined in

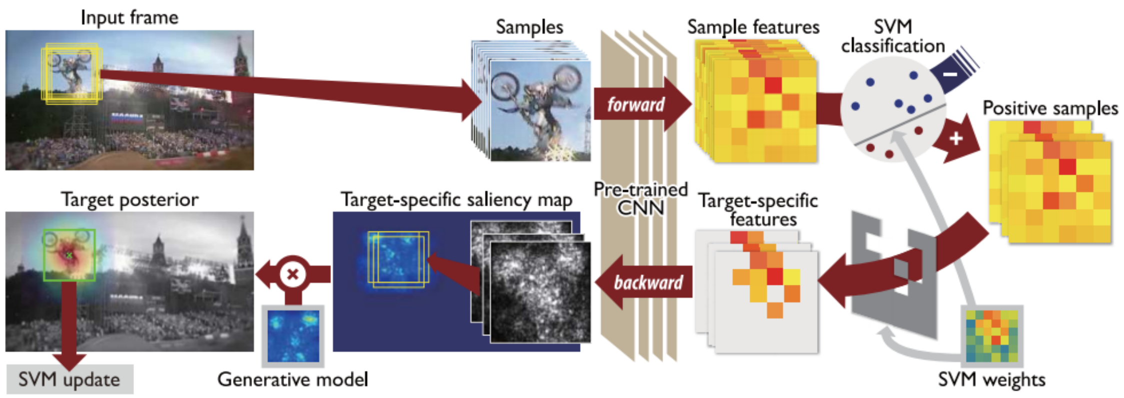


Figure 2.13 – The overall procedure for tracking with a discriminative saliency map [Hong et al., 2015a] computed with a convolutional neural network. Image courtesy of [Hong et al., 2015a].

a video sequence, several works have also trained DNNs from scratch or fine-tuned pre-trained DNNs in an online manner. Li et al. [2014] proposed a novel online CNN for object tracking, referred to as DeepTrack. It maintains a candidate pool of multiple CNNs built with four image cues, namely, three local-contrast normalized images and a gradient image. Given a frame, a subset of CNNs from this pool is applied to patches surrounding the previous target location. Then, the best matching CNN for each patch is determined by k-NN, and the matching score is assigned to the corresponding patch. Finally, the patch with the highest score is selected as the tracking result for the current frame. For the update step, the CNN corresponding to the selected patch is retrained with a warm-start back-propagation scheme, and the fully-connected layers of all the CNNs are retrained jointly. The overall procedure for determining the tracking result and updating the corresponding CNN is shown in Figure 2.11.

Wang et al. [2015b] presented a novel framework, called structured output deep learning tracking (SO-DLT), to pre-train and fine-tune CNN online, with structured output, to distinguish the target and background. First, an offline structured output CNN is trained using the ImageNet dataset [Russakovsky et al., 2015] in order to learn generic object features for classifying object and non-object regions. Then, two CNNs adopted from the offline CNN are fine-turned in an online manner for tracking: one is tuned aggressively to adapt well for appearance changes, while the other one is tuned conservatively to prevent drifting from inaccurate tracking results. The final tracking result is determined by the CNN with

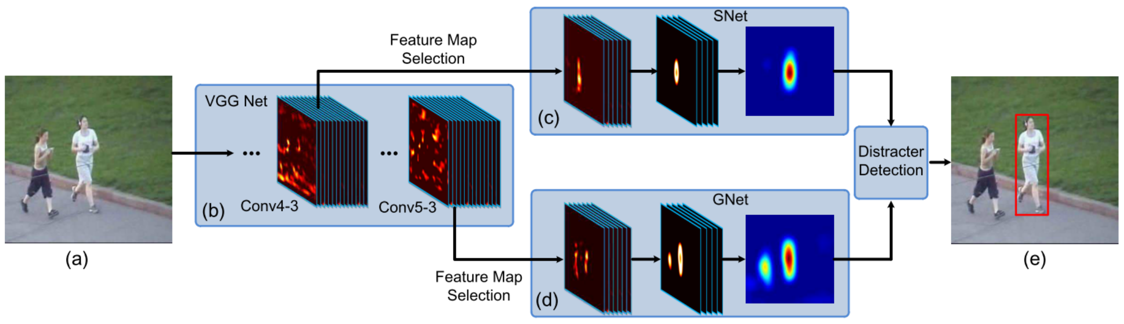


Figure 2.14 – Pipeline of a fully convolutional network based tracker [Wang et al., 2015a]: (a) Input ROI region, (b) VGG network, (c) SNet, (d) GNet, (e) Tracking results. Image courtesy of [Wang et al., 2015a].

the higher confidence.

In order to address the lack of readily-available data to train CNNs for the tracking problems, Nam and Han [2016] used additional video sequences from other tracking datasets, excluding videos belonged to the working tracking dataset. It is truly challenging to train CNNs on these additional video sequences directly, since the same kind of targets can be set as positive samples in one video sequence, and as negative samples in another sequence. Consequently, a novel CNN architecture, *Multi-Domain Network* (MDNet), shown in Figure 2.12, is proposed. During the offline training phase, each additional video sequence is treated as a separate domain, and propagated through the shared layers and the corresponding domain-specific layer of MDNet. By distinguishing domain-independent and domain-specific information, the shared layers of MDNet updated in every iteration, with all the training video sequences, are able to learn a generic feature representation. During the online tracking phase, all the pre-trained domain-specific layers are removed. Then, a new domain-specific layer, to classify the target object, and the pre-trained shared layers are fine-tuned frame-by-frame to adapt to the evolution of the target object. A bounding box regression method [Girshick et al., 2014] is also applied to improve the target localization.

Visual features extracted from the last layer of the network encode high-level semantic information [Zeiler and Fergus, 2014], but they model insufficient spatial details. A few approaches have attempted to address this issue from different perspectives. Inspired by [Simonyan et al., 2014], Hong et al. [2015a] constructed target-specific saliency maps obtained by back-propagating information relevant to the target. This keeps spatial information and provides accurate target segmentation. The overall

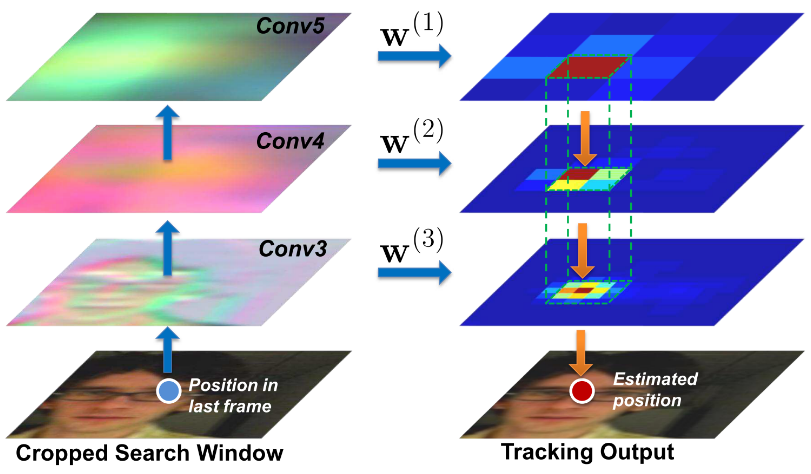


Figure 2.15 – Hierarchical convolutional features for visual tracking [Ma et al., 2015a]. The third, fourth and fifth convolutional layers from CNN are used to represent the target. Each layer indexed by i is then convolved with the learned linear correlation filter $\mathbf{w}^{(i)}$ to generate a multi-level response map. The location of the target is given by the maximum value in this response map following a coarse-to-fine strategy. Image courtesy of [Ma et al., 2015a].

approach is shown in Figure 2.13.

Instead of using the responses of the last CNN layer as a black-box feature, Wang et al. [2015a] utilized the representations learned in the intermediate convolutional layers. In particular, a top layer serves as a generic category detector, while a lower layer encodes a specific-instance detector with more discriminative information. Subsequently, a fully convolutional network based tracker (FCNT), including two convolutional layers of different levels, is designed for robust tracking, as shown in Figure 2.14. First, a selection of feature maps, based on a target heat map regression model, is performed on the lower and the higher layers of the pre-trained VGG network [Simonyan and Zisserman, 2015]. Then, a general network (GNet) that encodes category information is built on the selected feature map of a higher layer (i.e., conv5-3), while a specific network (SNet) that carries instance information is generated on the selected feature map of a lower layer (i.e., conv4-3). For every new frame, a region of interest centered at the target location in the previous frame is cropped and propagated through the fully convolutional network to generate two foreground heat maps with GNet and SNet. Based on these two heat maps, the tracking result is determined by a detection scheme, which distinguishes the target object from its distractors in the background. A

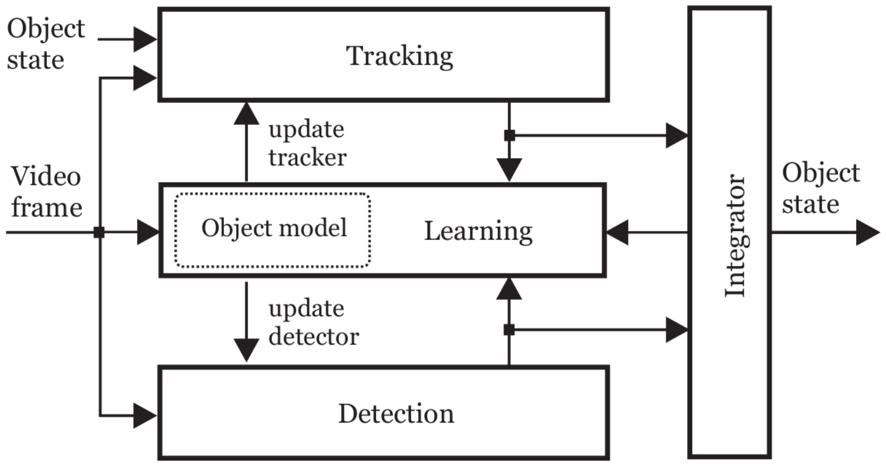


Figure 2.16 – Overview of the Tracking-Learning-Detection framework [Kalal et al., 2012]. Image courtesy of [Kalal et al., 2012].

related approach is proposed in [Ma et al., 2015a]. Three independent correlation filter-based trackers are applied on different convolutional layers, conv3-4, conv4-4, and conv5-4, using the pre-trained VGG network. Illustrated in Figure 2.15, the top layer of CNN encodes more semantic information but is insufficient for capturing fine-grained spatial details. In contrast, the earlier layers have rich information for localizing the object, but do not represent semantics well. The target location is inferred from multi-level correlation filter response maps, in a coarse-to-fine scheme.

2.2 Long-term Tracking

In traditional visual object tracking (i.e., short-term tracking), the stability-plasticity trade-off [Grossberg, 1987] or template update problem [Matthews et al., 2004] has not been addressed well. In order to alleviate this issue, Kalal et al. [2012] pointed out that the visibility of the target object, due to occlusion or the object leaving the field-of-view, should be indicated and processed accordingly in long video sequences, which led to the *long-term tracking* task.

In [Kalal et al., 2012], long-term tracking task is decomposed into three sub-tasks: tracking, learning and detection, and their corresponding components are designed to form a general tracker, called TLD. The tracking component estimates the object motion and follows the object continually in order to produce smooth trajectories, but it also accumulates errors

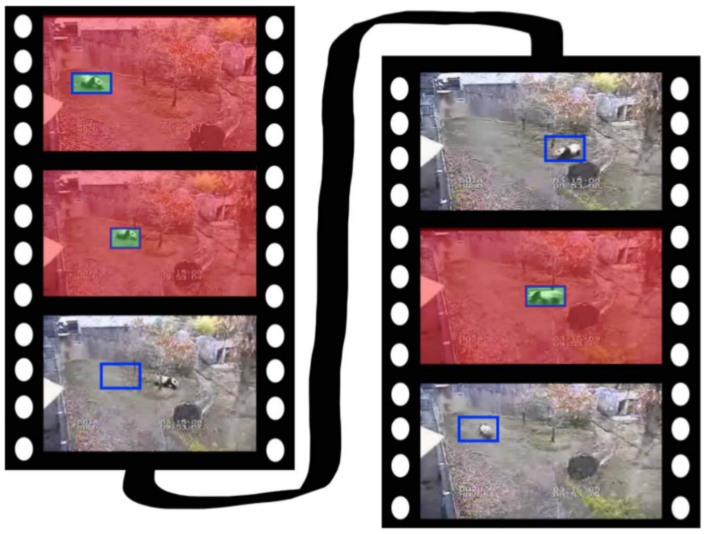


Figure 2.17 – As part of the tracking procedure, SPLTT [Supancic III and Ramanan, 2013] selects reliable frames (shown in red) to extract additional training data. The model is learned (updated) with these additional training data, which can correct the errors retrospectively. Image courtesy of [Supancic III and Ramanan, 2013].

continually and fails to track if the target is invisible. The detection component localizes the object in all its appearances that have been observed so far, and reinitializes tracker when it fails. The learning procedure estimates the quality of the results and updated it with only high reliable results. The overall procedure is shown in Figure 2.16.

Inspired by curriculum [Bengio et al., 2009] and self-paced [Kumar et al., 2010] learning, Supancic III and Ramanan [2013] argued that it is critical to revisit old frames during tracking since some object states are hard to track initially, but may become easy in retrospect. They proposed a novel framework for long-term tracking, referred to as SPLTT. Here, the target appearance is learned conservatively by revisiting previous frames and selecting the reliable ones for model updating, as shown in Figure 2.17.

Many existing trackers use a "censorship mechanism" to perform the update step when certain criteria are met. The main issue of this mechanism is that it can only prevent bad updates from happening, but can not correct the past mistakes. Zhang et al. [2014a] introduced a novel multi-expert entropy minimization (MEEM) restoration scheme, which allows a tracker to evolve backwards in order to undo undesirable

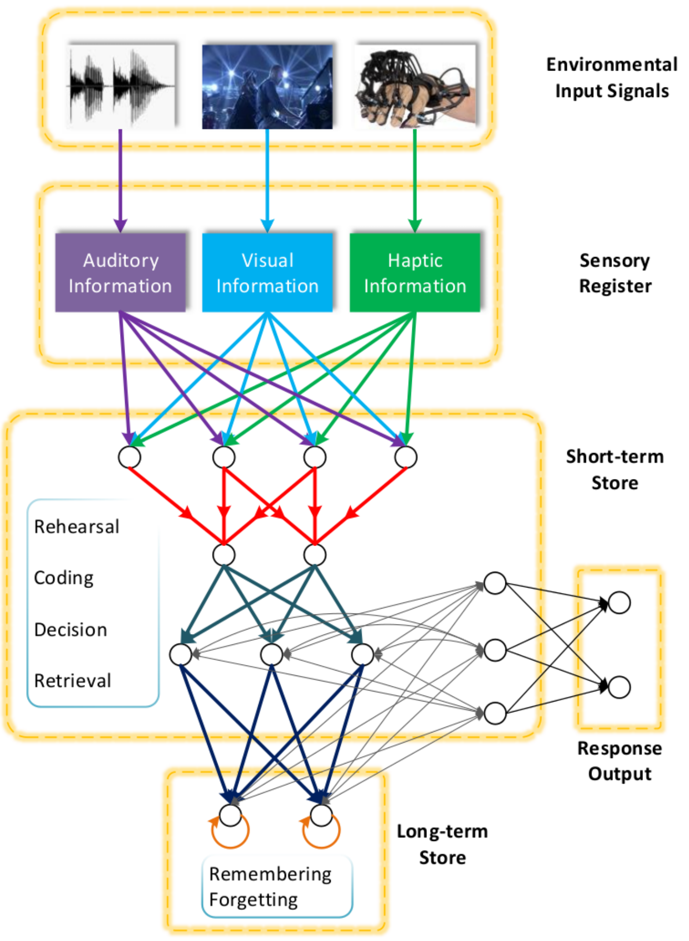


Figure 2.18 – Illustration of Atkinson-Shiffrin memory model with short-term and long-term stores [Hong et al., 2015b]. Image courtesy of [Hong et al., 2015b].

model updates. Within the MEEM tracking framework, a discriminative tracker and its former snapshots constitute an expert ensemble, and the best expert is selected based on a minimum loss criterion, with entropy-regularized optimization, to restore the tracker when a disagreement among the experts occurs.

Ma et al. [2015b] proposed LCT by extending the popular correlation filter-based tracker to the long-term tracking problem. LCT decomposes the long-term tracking task into the estimation of translation and scale parameters of the target objects, in conjunction with an essential re-detection scheme. For estimating translation, a correlation filter-based tracker (CFT_c) is applied to exploit the temporal correlation of the target and the surrounding spatial context, which is updated aggressively. CFT_c is robust to deformation, illumination variation, background clutter, and abrupt motion. In parallel, another correlation filter-based tracker (CFT_t) is constructed to determine the change in scale, and is updated conservatively. Two thresholds, corresponding to CFT_t, are set heuristically for activating target re-detection in case of a tracking failure, and updating the model with only trustworthy frames.

Inspired by the Atkinson-Shiffrin memory model (i.e., multi-store model, illustrated in Figure 2.18), Hong et al. [2015b] designed the multi-store tracker (MUSTer). It consists of one short-term store and one long-term store that collaboratively process the tracking procedure. An integrated correlation filter based on [Henriques et al., 2015] and [Danelljan et al., 2014a] is applied to store short-term memory for short-term tracking via two-stage filtering. Additionally, a complementary keypoint-based tracking and RANSAC estimation is utilized for long-term memory store, which controls the final output and the short-term memory states.

2.3 Tracking Datasets

Datasets play a critical role in almost all computer vision tasks. In the case of the object classification problem, there has been a tremendous evolution from Caltech101 [Fei-Fei et al., 2006] to PASCAL VOC [Everingham et al., 2010] and then to large-scale ImageNet [Russakovsky et al., 2015]. While such an evolution has also occurred in the case of tracking, it has been at smaller scale and a slower pace, and has its fair share of issues. Most video sequences in initial datasets were recorded in an unnatural experimental environment, or in some cases selected to highlight the advantages of the proposed tracker. Furthermore, they lack a common protocol for ground truth annotation, and are typically small in number. These issues are being addressed by recent datasets and benchmarks [Wu et al., 2013, Smeulders et al., 2014, Kristan et al., 2013, 2014, 2015, Felsberg et al., 2015, Li et al., 2015a].

In this section, we first briefly review publicly available model-free tracking datasets, and then introduce the datasets and corresponding evaluation methods used in this thesis.

2.3.1 Overview of tracking datasets

Amsterdam library of ordinary videos (ALOV++) dataset. Generality is a very important feature for good model-free trackers. Smeulders et al. [2014] argued that most trackers, have only been evaluated on a limited number of sequences, and the evaluation results are insufficient to make conclusive remarks on the validity and robustness of the proposed methods in a variety of circumstances. To address this issue, a large and diverse dataset was proposed. The ALOV++ dataset contains 315 video sequences with 89,364 frames in total from several sources: 22 sequences come from standard and recent tracking datasets, 65 sequences are from performance evaluation of tracking and surveillance (PETS) workshop [Chu and Smeulders, 2010], and 250 new sequences are collected from YouTube with 64 different types of targets. The ALOV++ dataset is annotated with a regular bounding box (i.e., axis-aligned box) enclosing the target. Due to the large size of the dataset, ground truth is manually annotated every fifth frame, while the annotation of the intermediate frames is obtained by linear interpolation. ALOV++ is available online at http://www.alov300.org.

NUS people and rigid objects (NUS-PRO) dataset. Proposed in [Li et al., 2015a], it is the largest publicly available tracking dataset so far, and contains 365 video sequences collected from YouTube. All the sequences in NUS-PRO belong to five categories, namely, face, pedestrian, sportsman, rigid object and long sequences. The five categories contain 17 kinds of objects in all. Many video sequences in the NUS-PRO dataset are recorded by hand-held cameras which makes it close to real-life scenarios, e.g., videos contain abrupt object movement or motion blur. Moreover, occlusion, usually missing or casually marked in other tracking datasets, is elaborately considered and annotated in three categories: no occlusion, partial occlusion and full occlusion. The NUS-PRO dataset and the evaluation system are available at http://www.lv-nus.org/pro/nus_pro.html.

Princeton tracking benchmark (PTB) dataset. Song and Xiao [2013] constructed an RGBD tracking dataset of 100 video sequences, which are captured with a standard Microsoft Kinect 1.0. It is the first attempt to build a tracking dataset with depth information, which significantly reduces the ambiguity existing in RGB images [Shotton et al., 2013], and can be used to prevent model drifting and handle occlusion cases. However, due to the constraint of the recording device, the depth of the captured

Dataset	#Videos	Ground truth (rectangle)	Data
OTB [Wu et al., 2013]	50	Axis-aligned	RGB, gray
PTB [Song and Xiao, 2013]	100	Axis-aligned	RGBD
VOT2013 [Kristan et al., 2013]	16	Axis-aligned	RGB
ALOV++ [Smeulders et al., 2014]	315	Axis-aligned	RGB, gray
VOT2014 [Kristan et al., 2014]	25	Rotated	RGB
TB-50/100 [Wu et al., 2015]	100	Axis-aligned	RGB, gray
NUS-PRO [Li et al., 2015a]	365	Axis-aligned	RGB
VOT2015 [Kristan et al., 2015]	60	Rotated	RGB
VOTTIR2015 [Felsberg et al., 2015]	20	Axis-aligned	Thermal infrared

Table 2.2 – A summary of popular tracking datasets in chronological order.

object can only vary from 0.5 to 10 meters, and thus all the RGBD video sequences are captured indoors. The PTB dataset and the evaluation system are available at `http://tracking.cs.princeton.edu`.

We use the-state-of-the-art object tracking benchmark (OTB) [Wu et al., 2013] and the visual object tracking (VOT) challenge [Kristan et al., 2013, 2014, 2015, Felsberg et al., 2015] datasets extensively in this thesis, and will discuss them in §2.3.2. A summary of tracking datasets is shown in Table 2.2.

2.3.2 Datasets and evaluation methods used

Object tracking benchmark (OTB) dataset

The object tracking benchmark dataset [Wu et al., 2013], named OTB, is a collection of 50 commonly used tracking sequences, where the object varies in scale, has fast motion, or is occluded. The first frame of each sequence in OTB is illustrated in Figure 2.19.

In order to present the progress of tracking algorithms and set a general benchmark, 29 methods are compared in [Wu et al., 2013]. Two well-adopted evaluation methodologies are used: precision and success. Precision reflects the center location error. It is measured as the percentage of frames whose predicted object location (center of the predicted box) is within a distance varying between 0 and 50 pixels from the center of the ground truth box. The precision score is the percentage value when threshold distance is set to 20 pixels. The success measure is based on

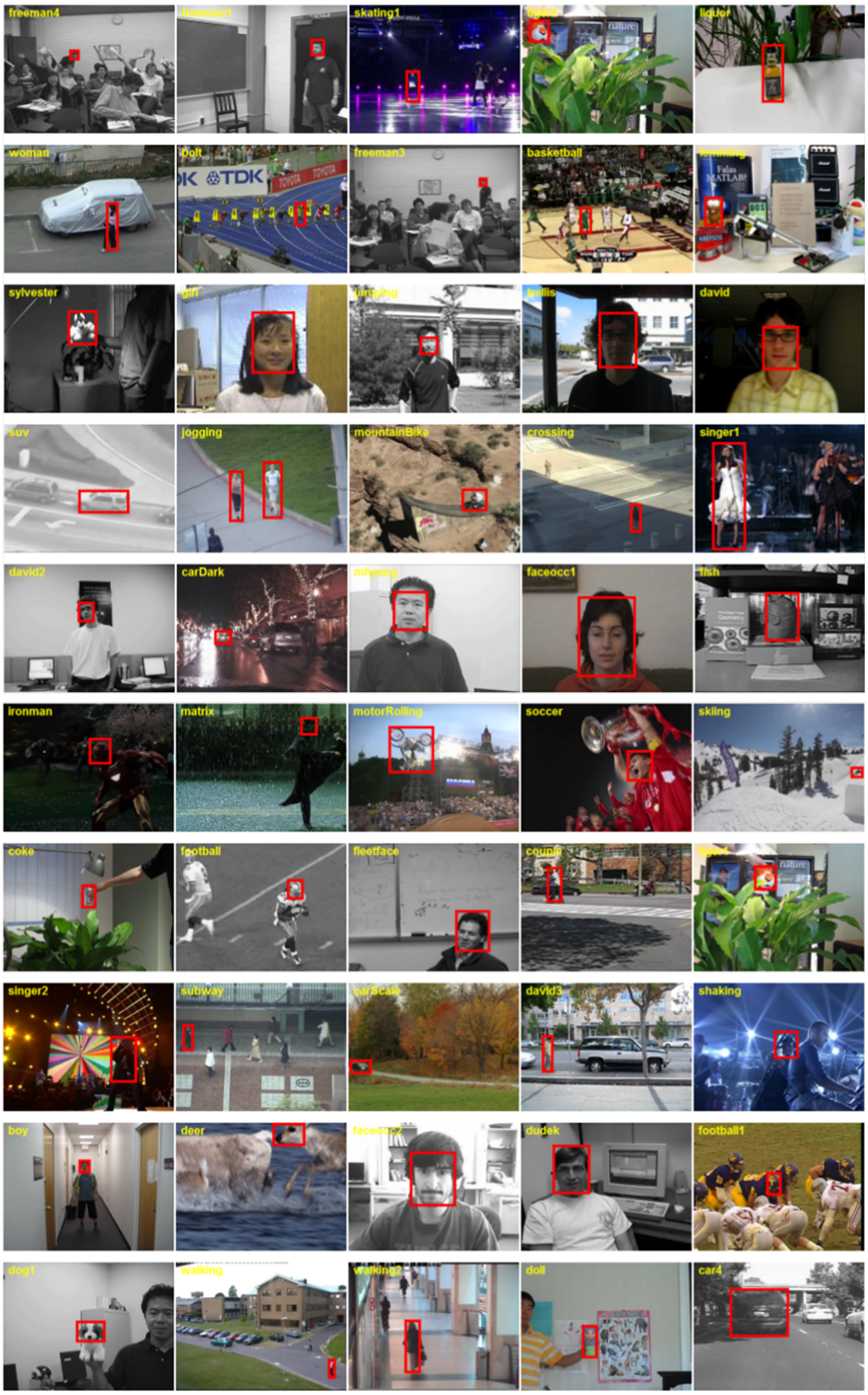


Figure 2.19 – Illustration of the OTB dataset [Wu et al., 2013], showing the first frame in each sequence along with the ground truth target annotation. Image courtesy of [Wu et al., 2013].

the bounding box overlap. It shows the percentage of frames whose intersection over union overlap with the ground truth annotation is over a threshold, varying between 0 and 1. Instead of using a fixed threshold, the area under curve (AUC) of the success plot determines the success score in order to rank the algorithms. The robustness of different trackers is evaluated with the following three procedures.

1. One-pass evaluation (OPE): It is a conventional method to evaluate trackers. All the trackers are run on the test sequences with the initializations from the ground truth position in the first frame, and the average precision and success scores are measured.
2. Temporal robustness evaluation (TRE): Each sequence for testing is divided uniformly into 20 segments. Each tracker is initialized at the beginning of a segment and evaluated until the end of the entire sequence. The tracking results of all the 20 tests are averaged to generate the precision and success scores.
3. Spatial robustness evaluation (SRE): For every sequence, each tracker is initialized in the first frame with shifted or scaled ground truth bounding box. As a default, each tracker is evaluated 12 times with different initial bounding box settings: eight spatial shifts including four center shifts and four corner shifts (10% of target size), and 4 scale variations (i.e., 0.8, 0.9, 1.1 and 1.2) with respect to the ground truth in the first frame. The precision and success scores are calculated from the average of all these 12 evaluations in order to rank the trackers.

Visual object tracking (VOT) challenge dataset

The visual object tracking (VOT) challenge was introduced in 2013 with the aim of providing a standardized platform to evaluate single-camera, single-target, model-free, causal short-term tracking algorithms. It has been organized as an annual workshop in conjunction with ICCV or ECCV conferences. In each workshop, a fully annotated dataset with several per-frame visual attributes is released. Each frame in the dataset is manually or semi-automatically labeled with six visual attributes, including occlusion, illumination change, motion change, size change, camera motion and unassigned. In addition to the dataset, an evaluation toolkit is also developed and actively maintained, which allows easy integration of third-party trackers for fair comparison. We used the datasets released for the two recent challenges, i.e., 2014 and 2015, in this thesis.

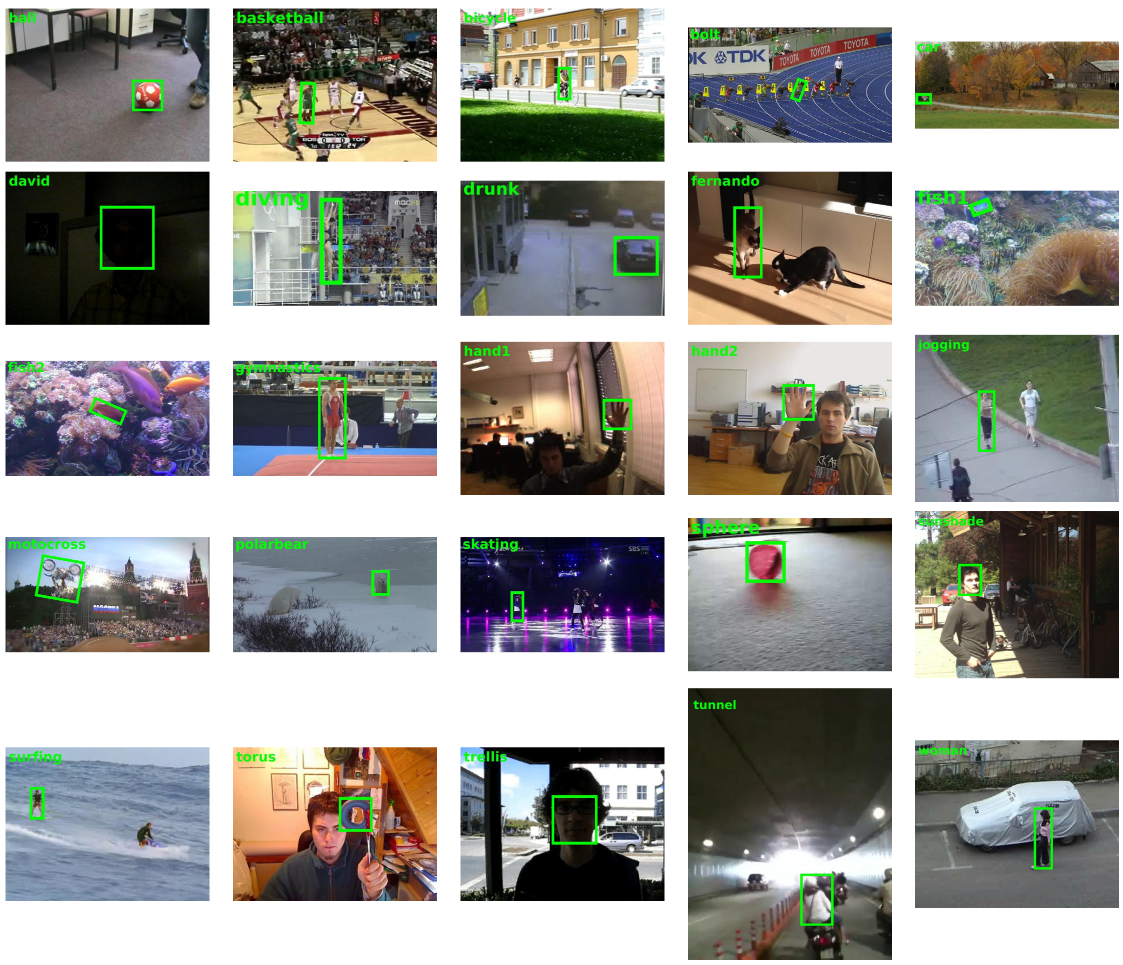


Figure 2.20 – Illustration of VOT2014 challenge dataset [Kristan et al., 2014], showing the first frame in each sequence along with the initial bounding box of the target object.

The evaluation scheme of VOT challenge uses accuracy and robustness measures to compare trackers, due to their high level of interpretability [Čehovin et al., 2014, 2015]. Raw accuracy is computed as the mean intersection over union score with the ground truth bounding box over the entire sequence (while discarding ten frames immediately following a tracking failure to further reduce the bias in accuracy measure), and raw robustness is the number of times the tracker has failed. A tracking failure is signaled in a frame t if the predicted box does not overlap with the ground truth annotation. In this case, the tracker is restarted from scratch in frame $t+5$ with the corresponding ground truth annotation in order to alleviate the bias in robustness measure. For a robust comparison,

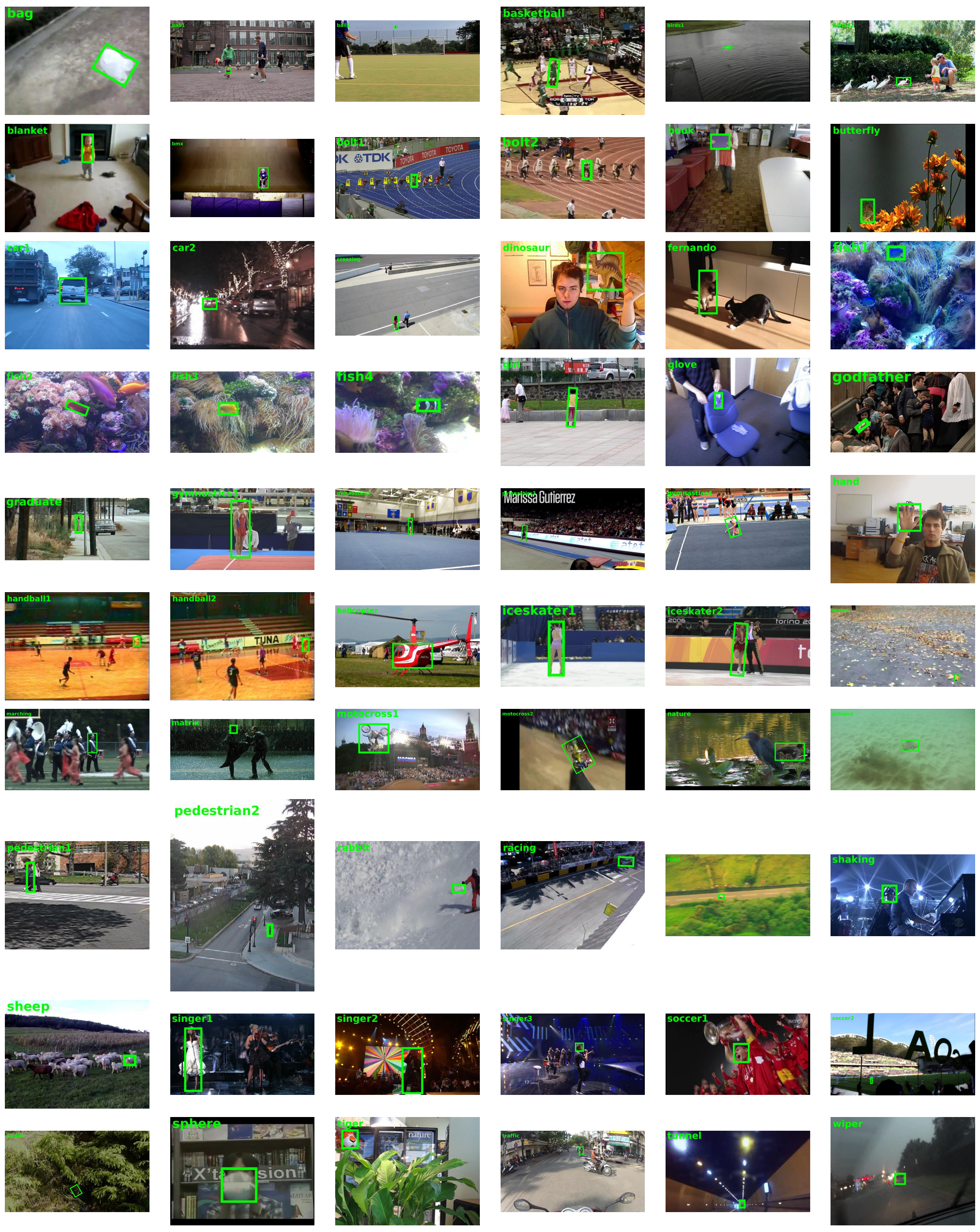


Figure 2.21 – Illustration of VOT2015 challenge dataset [Kristan et al., 2015], showing the first frame in each sequence along with the initial bounding box of the target object.

the scores are averaged over 15 runs of the tracker to account for any stochastic behavior.

VOT2014 challenge dataset. It contains a set of 25 challenging video sequences chosen from a candidate set of 394 sequences, which is composed of examples used in several previous works, such as VOT2013 challenge dataset [Kristan et al., 2013], OTB [Wu et al., 2013], ALOV++ [Smeulders et al., 2014], as well as a few unpublished ones. This choice was made by: (i) discarding sequences shorter than 200 frames, and (ii) ensuring that the selected sequences contain well-defined targets and represent challenging scenarios, such as clutter, object deformation, and change in aspect ratio. The frames in all the sequences are manually annotated. In a departure from annotations in other datasets, where the bounding box enclosing the target is axis-aligned, this dataset uses rotated boxes in order to handle targets that are deforming, as shown in Figure 2.20. This new annotation makes the dataset a more challenging setting to evaluate trackers. In addition, all the frames in VOT2014 challenge dataset are labeled with visual attributes: occlusion, illumination change, object motion, object size change, camera motion and neutral [Kristan et al., 2014].

The overall rank of a tracker is determined by first ranking trackers on accuracy and robustness separately on each attribute set, and then taking the mean rank over the two performance measures. If a set of trackers show similar performance (which is measured with statistical significance and practical difference tests), their ranks are set to the minimal rank of these equivalent trackers.

VOT2015 challenge dataset. Compared with VOT2014, two improvements are made in the VOT2015 challenge [Kristan et al., 2015]. The first one is a dataset more than twice as large, containing 60 sequences in total (cf., Figure 2.21). This dataset is fully annotated with rotated boxes and per-frame attributes. The second improvement is the evaluation strategy. The average of the accuracy and robustness ranks was used in VOT2013 and VOT2014. But it cannot be interpreted as a standalone result, since the evaluation is based on relative ranking, which is susceptible to the quality and the number of trackers compared. In other words, the performance of a tracker is determined not only by itself, but also all the trackers used for comparison.

To address this, a new measurement called *expected average overlap* is introduced, which combines the raw values of per-frame accuracies and

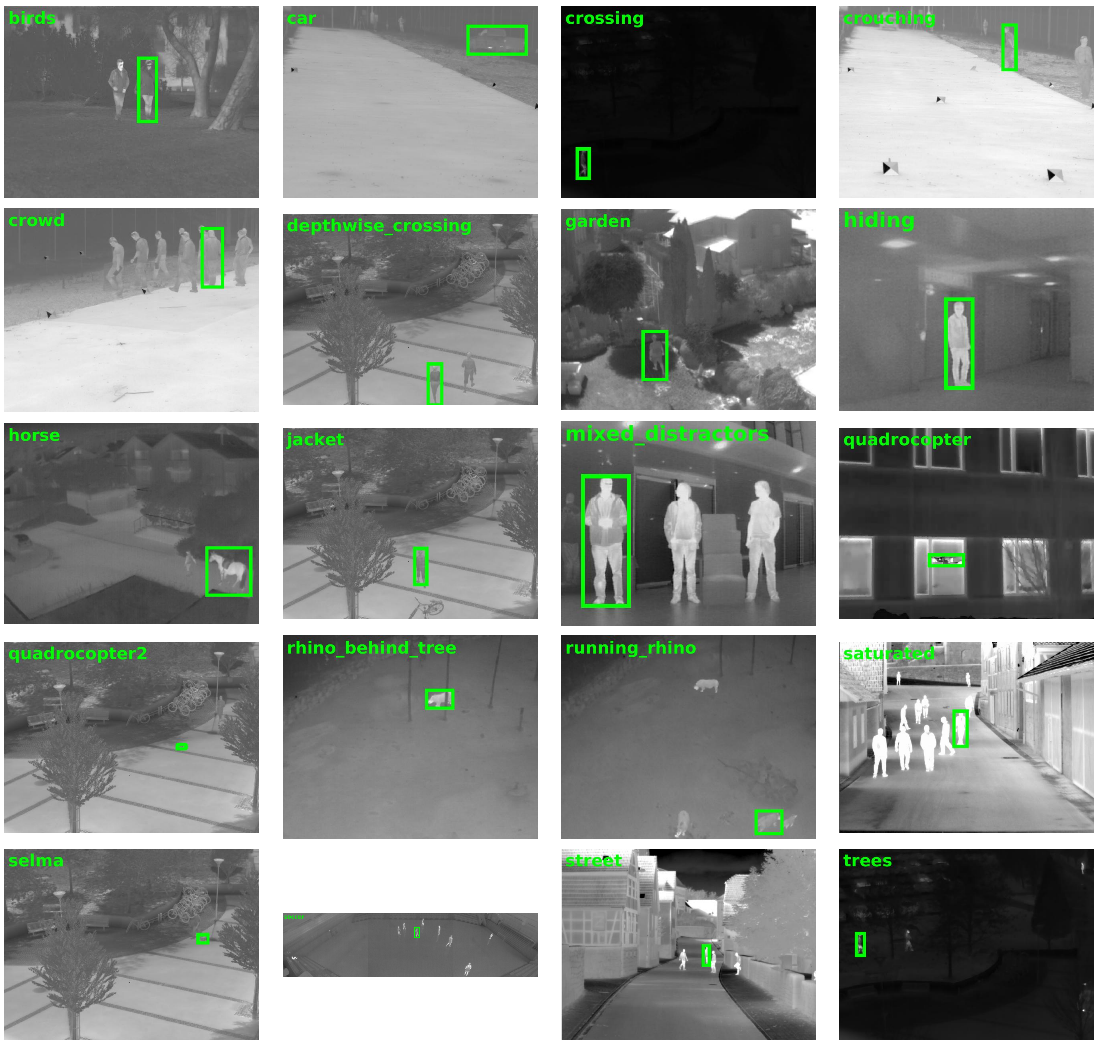


Figure 2.22 – Illustration of VOT-TIR2015 dataset [Felsberg et al., 2015], showing the first frame in each sequence along with the initial bounding box of the target object.

failures in a principled way. In other words, expected average overlap is a constant value to represent performance for each individual tracker, which is not dependent on other trackers. It is calculated as the average of the expected average overlap curve values over an interval $[N_{lo}, N_{hi}]$ of typical sequence lengths in the dataset. The intervals are computed by a kernel density estimate from the lengths of all the sequences in the dataset. For instance, the lengths of sixty sequences are used for estimating the range value for the VOT2015 dataset.

VOT-TIR2015 challenge dataset. A new sub-challenge was introduced along with the VOT2015 challenge, involving a thermal infrared (TIR) tracking dataset [Felsberg et al., 2015]. This dataset is annotated with common axis-aligned boxes and per-frame attributes. This new challenge adopts all the protocols from the VOT challenge, and utilizes the same toolkit for performance evaluation. The first frame of each sequence in VOT-TIR2015 is illustrated in Figure 2.22.

Chapter 3

Online Object Tracking with Proposal Selection

Contents

Tracking-by-detection approaches are some of the most successful object trackers in recent years [Wu et al., 2013, Smeulders et al., 2014, Pang and Ling, 2013, Kristan et al., 2014, Song and Xiao, 2013]. Their success is largely determined by the detector model they learn initially and then update over time. However, under challenging conditions where an object can undergo transformations, e.g., severe rotation, these methods are found to be lacking. Furthermore, most tracking-by-detection approaches usually determine tracking results only based on the best detection score, which is not always optimal and may gradually cause drift. In this chapter, we address these problems by formulating visual object tracking as a proposal selection approach. We make two contributions: (i) We introduce novel proposals estimated from the geometric transformations undergone by the object, and build a rich candidate set for predicting the object location. (ii) We devise a two-phase selection strategy using multiple cues, i.e., detection and edgebox scores. We evaluate our approach extensively on the visual object tracking (VOT) challenge and online tracking benchmark (OTB) datasets, and show top performance.

3.1 Introduction

The tracking-by-detection framework formulates the tracking problem as the task of detecting an object, which is likely to undergo changes in appearance, size or become occluded over time [Avidan, 2007, Grabner et al., 2008, Babenko et al., 2011, Mei and Ling, 2009, Godec et al., 2011, Hare et al., 2011, Kalal et al., 2012, Supancic III and Ramanan, 2013]. These approaches begin by training an object detector with an initialization (in the form of a bounding box) in the first frame, and then update this model over time. Naturally, the choice of exemplars used to update and improve the object model is critical [Supancic III and Ramanan, 2013, Hua et al., 2014, Zhang et al., 2014a].

Consider the *motocross* example shown in Figure 3.1. Here, the target (biker and his motorbike) undergoes several deformations as the biker performs an acrobatics routine, which leads to significant changes in the aspect ratio as well as the rotation angle of the bounding box. Traditional tracking-by-detection approaches, e.g., [Hare et al., 2011, Danelljan et al., 2014a], rely on axis-aligned bounding boxes and thus are ill-equipped to capture the accurate extents of the object in such scenarios. In this chapter, we aim to address this issue by tracking based on selecting the best proposal containing the object of interest from several candidates.

Here, we focus on the problem of tracking a single target in monocular video sequences. The target is provided as a ground truth annotation in the first frame, as in a standard tracking-by-detection setup [Hare et al., 2011, Danelljan et al., 2014a, Supancic III and Ramanan, 2013]. We learn an object detector model from this annotation and evaluate it in subsequent frames to propose candidate locations that are likely to contain the target. While these proposals are sufficient to track objects undergoing a small subset of transformations over time, such as translation shown in

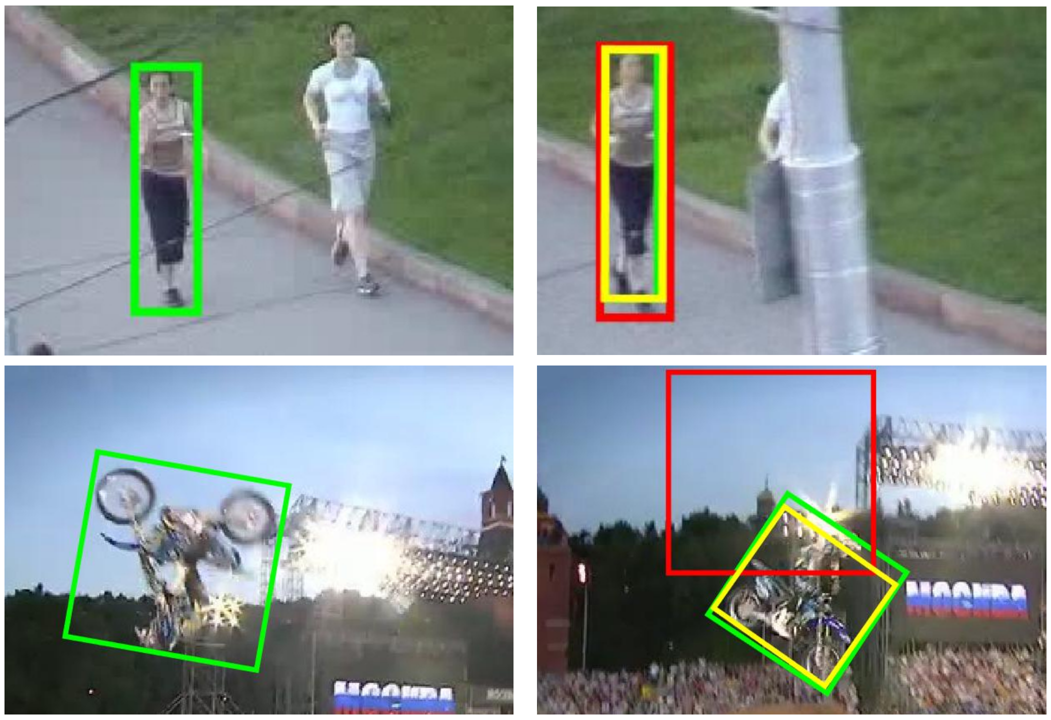


Figure 3.1 – Sample frames (cropped) from the *jogging* (top row) and *motocross* (bottom row) sequences [Kristan et al., 2014]. The ground truth annotation (green) in the first frame (left) is used to train our tracker and the winner [Danelljan et al., 2014a] of VOT2014 challenge. We show these two tracking results (right) on another frame in the sequence. Our method (yellow) successfully tracks objects undergoing geometric transformations unlike [Danelljan et al., 2014a] (red).

Figure 3.1-top, they cannot handle generic transformations, e.g., similarity transformation shown in Figure 3.1-bottom. Another issue is that such approaches typically determine tracking results with only detection scores — a suboptimal solution to localize the target. Subsequently, updating the detection model with these slightly inaccurate samples may accumulate localization errors and cause drift gradually.

We address these problems by: (i) introducing novel additional proposals to improve the candidate location set, and (ii) using multiple cues to select the proposal that is most likely to contain the target. Additional proposals are computed by robustly estimating the parameters of similarity transformation (i.e., scaling, translation, rotation) that the target is likely to have undergone, with a Hough transform voting scheme on the optical flow. With these parameters, we propose several candidate locations for the target. Thus, for every frame of the video sequence, our candidate set consists of object detector boxes as well as geometry estimation proposals. Note that state-of-the-art tracking-by-detection approaches are limited to detector proposals alone. The second contribution we make is in selecting the best proposal from the candidate set using multiple cues — detection score, and edgebox scores [Zitnick and Dollár, 2014] computed with edge responses and motion boundaries. In order to avoid setting explicit weights for combining multiple cues, we utilize a two-phase selection strategy. In the first phase, we select the best candidate merely based on detection score. If there are several candidates with similar detection scores, we then move to the second phase and make the final decision by comparing edgebox scores calculated from edge responses and motion boundaries. This two-phase selection strategy ensures that we find a high-quality proposal with sufficiently good detection score as well as a precise localization.

We evaluate the performance of our approach exhaustively on recent benchmark datasets, namely the visual object tracking (VOT) challenge dataset [Kristan et al., 2014, 2015] and the online tracking benchmark (OTB) [Wu et al., 2013]. Furthermore, we compare with all the 38 trackers evaluated as part of the VOT2014 challenge, and with the best performers [Hare et al., 2011, Jia et al., 2012, Zhong et al., 2012] among the 29 methods analyzed in the OTB comparison paper [Wu et al., 2013]. Our method shows the top performance on these challenging datasets, and it is in particular 29.5% better than the current leader [Danelljan et al., 2014a] of the VOT2014 challenge. Our approach submitted to the visual object tracking challenge in 2015 won one of the competitions. This submission, with a detailed description and parameter settings, is presented in Appendix A.

The remainder of the chapter is organized as follows. In Section 3.2 we discuss closely related work. The details of our novel proposals and the multiple cues to select the best candidate are explained in Section 3.3. Section 3.4 discusses the implementation details of the methods. In Section 3.5 we present an extensive evaluation of the proposed method and compare it with the state of the art. Concluding remarks are made in Section 3.6.

3.2 Related Work

Two of the key elements of any tracking algorithm are, how the object of interest is represented, and how this representation is used to localize it in each frame. Several methods have been proposed on these two fronts. Object representations have evolved from classical color histograms [Comaniciu et al., 2003] to models learned from generative [Black and Jepson, 1998, Ross et al., 2008, Mei et al., 2011] or discriminative [Avidan, 2004, Collins et al., 2005, Babenko et al., 2011, Hare et al., 2011, Danelljan et al., 2014a, Supancic III and Ramanan, 2013] approaches.

More recently, inspired by the success of object detection algorithms [Everingham et al., 2010, Felzenszwalb et al., 2010], the discriminative tracking-by-detection approach has gained popularity. In fact, methods based on this paradigm are ranked among the top performers on evaluation benchmarks [Wu et al., 2013, Smeulders et al., 2014, Pang and Ling, 2013, Kristan et al., 2014]. Standard discriminative tracking-by-detection methods learn an initial model of the object from the first frame in the sequence (e.g., with a support vector machine (SVM) [Hare et al., 2011, Supancic III and Ramanan, 2013]), evaluate it to detect the most likely object location in subsequent frames, and then update the object model with these new detections. Struck [Hare et al., 2011] is an interesting variant of this general framework, using a structured output formulation to learn and update the detector. DSST [Danelljan et al., 2014a] improves the discriminative correlation filtering scheme by estimating the scale of the target and combining intensity and HOG features.

Despite the general success of the discriminative tracking-by-detection approaches, they have an important shortcoming. They all rely solely on a detector, and are thus unable to cope with transformations an object can undergo. As a result, their models cannot be updated properly and cause drift gradually. On the contrary, key point based trackers like CMT [Nebehay and Pflugfelder, 2014] detect key points in a frame, match them to the next frame and can estimate transformations, such as changes in scale and rotation. However, they are highly sensitive to the success of key point detection algorithms and lack a learned model. Indeed, this is evident from the poor performance on the VOT dataset (see Section 3.5). Hua et al. [2014] proposed to use a set of trackers to maintain all the geometry transformations estimated by RANSAC based on frame-to-frame optical flow. In other words, if the target object undergoes obvious geometry transformations, a new discriminative tracker will be trained and added into the tracker pool. Nevertheless, this approach requires maintaining a set of trackers, all of which need to be evaluated independently in every frame.

The main focus of this chapter is to address the gap between aforementioned two paradigms, i.e., tracking-by-detection and key point-based approaches. Inspired by successful object proposal methods for detection and segmentation [Alexe et al., 2012, Carreira and Sminchisescu, 2010], we pose the tracking problem as the task of finding the best proposal from a candidate set, see Figure 3.2.

To some extent, our selection strategy aligns with several tracking by fusion approaches, such as [Isard and Blake, 1998b, Badrinarayanan et al., 2007, Du and Piater, 2008, Santner et al., 2010, Park et al., 2012]. Those approaches have incorporated multiple cues and fused their results with particle filtering [Isard and Blake, 1998b, Badrinarayanan et al., 2007], hidden Markov model [Du and Piater, 2008, Park et al., 2012] or pre-defined multi-cue selection strategy in a cascaded manner [Santner et al., 2010]. Our selection approach is similar to the fusion method used in [Santner et al., 2010]. However, the purpose of these two approaches are different. In order to improve the adaptivity while keeping robustness, Santner et al. [2010] proposed a cascaded multi-cue selection strategy, i.e., the tracking result of more stable cue can overrule the result from highly adaptive cue if conflicts happen. In contrast, our two-phase selection approach focuses on improving localization accuracy, while selecting candidates with sufficiently good detection score.

3.3 Proposal Selection Tracking

We now present all the components of our tracker (cf. Figure 3.2) in this section and then provide implementation details in Section 3.4.

3.3.1 Initial detector

We learn the initial detector model with a training set consisting of one positive sample, available as a bounding box annotation in the first frame, and several negative bounding box samples which are automatically extracted from the entire image. We use HOG features [Dalal and Triggs, 2005, Felzenszwalb et al., 2010] computed for these bounding boxes and learn the detector with a linear SVM, similar to other tracking-by-detection approaches [Supancic III and Ramanan, 2013, Hua et al., 2014]. The detector is then evaluated on subsequent frames to estimate the candidate locations of the object. Rather than make a suboptimal decision by choosing the best detection as the object location in a frame, we extract the top k detections and build a candidate set. We augment this

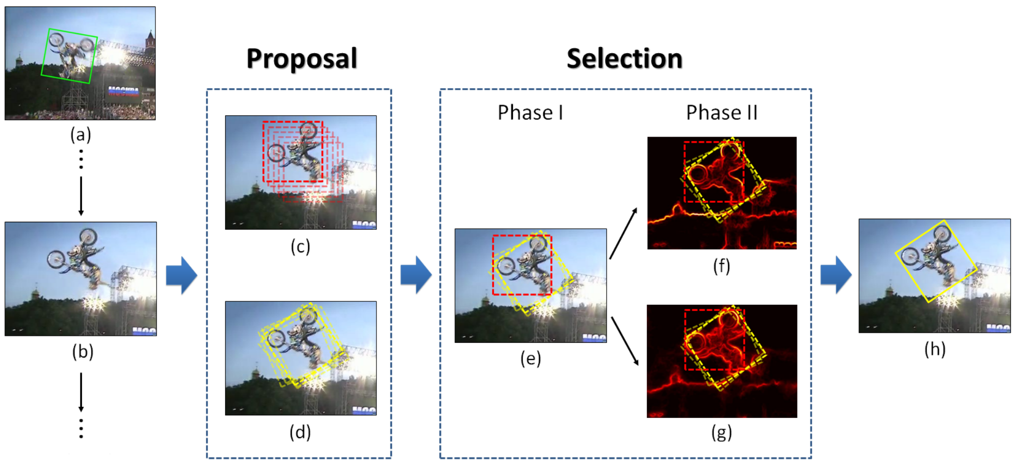


Figure 3.2 – The overall framework of proposal selection tracker: (a) Initialization; (b) A sample frame in which the object is to be tracked; (c) Tracking-by-detection proposals; (d) Geometry proposals; (e) Selection phase I: Detection score; Selection phase II: Edgebox score from (f) edge responses; (g) motion boundaries; (h) Selected tracking result and model updating.

set with proposals estimated from the transformations undergone by the object, as described in the following section.

3.3.2 Estimating geometry and proposals

In the examples shown in Figure 3.3, the object of interest is undergoing a geometric transformation (rotation in these examples). None of the top detector proposals can capture the object accurately in this case. To address this issue, we explicitly estimate the transformation undergone by the object and then use it to enrich the candidate set with additional geometric proposals.

We represent the geometric transformation with a similarity matrix. Note that other transformations like homography can also be used. The similarity transformation is defined by four parameters – one each for rotation and scale, and two for translation. In this work, we estimate them with a Hough transform voting scheme using frame-to-frame optical flow correspondences. We begin by computing the optical flow between frames $t-1$ and t with a state-of-the-art flow estimation method [Brox and Malik, 2011]. The pixels within the object bounding box with flow values in frame $t-1$ then give us corresponding pixels in frame t. With a

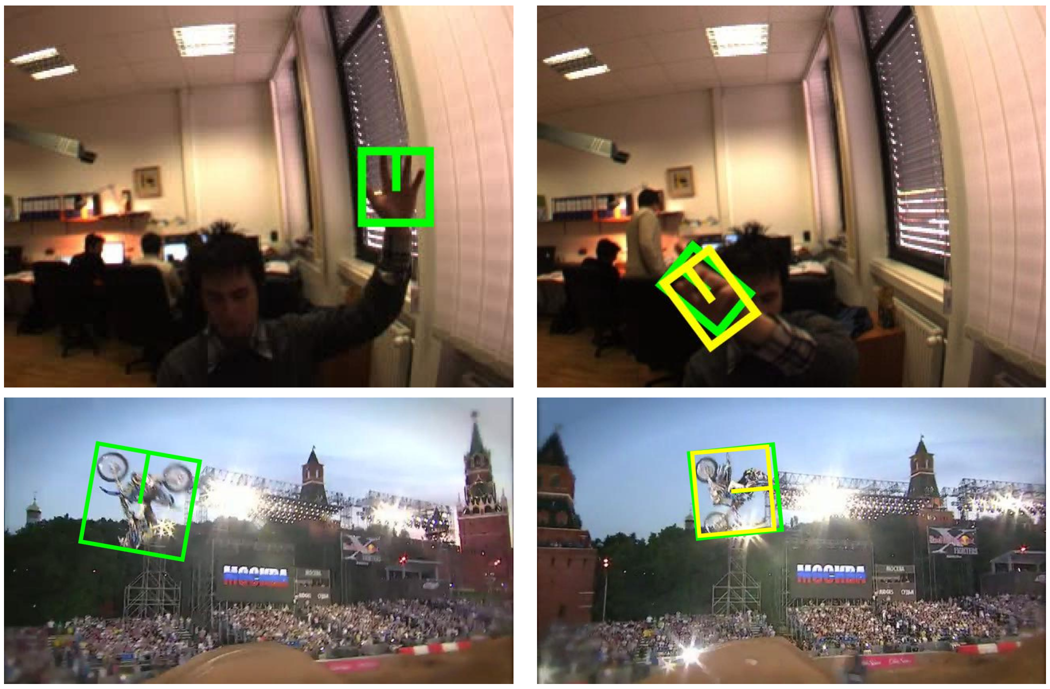


Figure 3.3 – Illustration of our geometric proposals on two sequences from the VOT2014 dataset. In each row, we show the ground truth (in green) in the first frame (left) and the best geometry proposal in a subsequent frame (in yellow in the right image). We show a vertical line on each box to visualize the rotation angle with respect to image edges.

pair of these matches, given by two pixels in $t-1$ which are sufficiently distant from each other, we estimate the four scalar parameters in the similarity matrix [Hartley and Zisserman, 2004]. Every choice of a pair of corresponding matches gives us an estimate of the four parameters. We then use the Hough transform [Hough, 1962], wherein each pair of point-matches votes for a (4D) parameter set, to find the top k consistent parameter sets. This voting scheme allows us to filter out the incorrect correspondences due to common errors in optical flow estimation.

To sum up, we estimate the k most likely geometric transformations undergone by the object as the parameters with the top k votes. Each one of these transformations results in a candidate location of the object in t, by applying the corresponding similarity matrix to the object bounding box in $t-1$. Before adding all these k proposals to the candidate set, we perform an additional filtering step to ensure further robustness. To this end, we discard all the proposals: (i) which have a low confidence, given by the number of votes, normalized by the total number of point-matches,

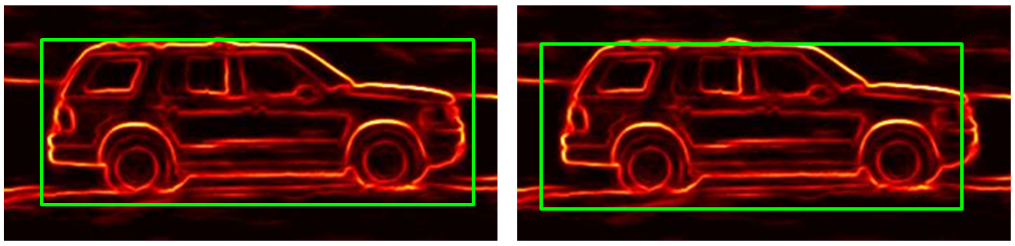


Figure 3.4 – Comparison of edgebox score calculated edge responses of two proposals. The proposal shown on the left is a better localization than the one on the right, which is supported by its higher edgebox score.

and (ii) those where the estimated angle is significantly different from estimations in earlier frames. We determine both these threshold values with Kalman filtering [Kalman, 1960]. A threshold τ_t, used for filtering proposals in frame $t+1$, is given by:

$$\tau_t = \alpha\tau_{t-1} + k_t(d_t - \alpha\beta\tau_{t-1}), \tag{3.1}$$

$$k_t = (p_{t-1} + q)/(p_{t-1} + q + r), \tag{3.2}$$

$$p_t = (1 - k_t)(p_{t-1} + q), \tag{3.3}$$

where α, β, q, r are scalar parameters set empirically (see Section 3.4), k_t is the Kalman gain, p_t is the error estimate. The filter is initialized with $\tau_0 = 0$, $p_0 = 0.1$. Here, d_t is either the confidence score or the estimated angle of the selected proposal in frame t to determine the respective threshold.

3.3.3 Selecting proposals

At the end of the geometry estimation step, we have k proposals from the detector and (at most) k proposals from the similarity transformation in frame t. Now, the task is to find the *best* proposal most likely to contain the object in this frame. We use three cues, detection confidence score, edgebox measures [Zitnick and Dollár, 2014] computed with object edge responses [Dollár and Zitnick, 2013] and motion boundaries [Weinzaepfel et al., 2015], for this task.

We propose a two-phase selection strategy to combine all the cues. In the first phase, we use the (normalized) detection confidence score computed for each proposal box with the SVM model. This provides information directly relevant to the object of interest in a given sequence. In

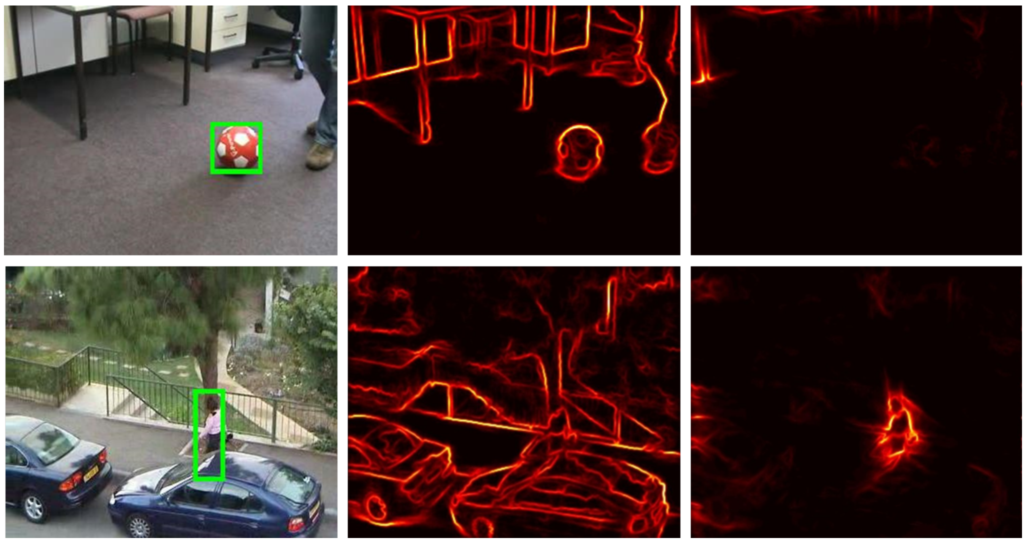


Figure 3.5 – Two examples of edge responses and motion boundaries. In each row, from left to right, we show the original image (with the object in a green box), edge responses and motion boundaries. Edges provide a stronger response in the example in the first row while motion boundaries are more effective in the example in the second row.

cases where the detection scores of all the candidate boxes are statistically similar, we then move to the second phase — use cues extracted from object edge responses and motion boundaries in the image, referred to as edgebox scores. In other words, when the detection scores are inconclusive to choose the best proposal, we rely on edgebox measure. Here, all the top candidates contain the object of interest to some extent, and the task is to choose the one that best contains the object. This scenario is well-suited for edgebox measure because a proposal box which accurately localizes the entire object has a higher edgebox score compared to a box which overlaps partially with the object, thus resulting in a lower edgebox score, as shown in Figure 3.4.

The edgebox score is given by the number of edge responses (or motion boundaries) within a proposal box, after discarding contours that intersect with the box's boundary, as in [Zitnick and Dollár, 2014]. We compute edgebox scores with edge responses and motion boundaries separately and use the one with a higher (normalized) score. As shown in the examples in Figure 3.5, these two cues are complementary. Then, the proposal with the best edgebox score is selected.

It is worth emphasizing that we follow this two-phase selection approach, rather than using a combined score, e.g., a weighted combination

Algorithm 1 : Our approach for online object tracking.

```
**Data**: Image frames $1 \ldots n$, ground truth $\text{box}_1$ in frame 1
**Result**: Object location $\text{box}_t$ in frames $t = 2 \ldots n$
Learn initial detector model in frame 1 (§3.3.1)
**for** $t = 2 \ldots n$ **do**
    candidate set $\mathcal{C}_t \leftarrow$ Top-$k$ detections in frame $t$ (§3.3.1)
    $\mathcal{H}_t \leftarrow$ Geometry proposals in frame $t$ (§3.3.2)
    $\mathcal{C}_t \leftarrow \mathcal{C}_t \cup \mathcal{H}_t$
    $\text{box}_t \leftarrow$ Best proposal in $\mathcal{C}_t$ (§3.3.3)
    Update detector model with $\text{box}_t$ (§3.3.4)
**end**
```

of SVM and edgebox scores, because setting this weight manually is suboptimal, and learning it is not possible due to the lack of standard training-validation-test datasets for tracking. In contrast, our approach uses object-specific SVM and generic edgebox scores effectively, and circumvents the need for a manually set weight to combine scores.

3.3.4 Updating the detector

Having computed the best proposal containing the object, box_t, we use it as a positive exemplar to learn a new object model. In practice, we update the current model incrementally with this new sample; see Section 3.4. This new detector is then evaluated in frame $t+1$, to continue tracking the object. Algorithm 1 summarizes our overall approach.

3.4 Implementation Details

Detector. The initial detector is trained with the one positive sample (provided as ground truth box in the first frame) and several negative examples extracted from the frame. We harvest bounding boxes that overlap less than 50% with the ground truth for the negative set. The regularization parameter in the SVM cost function is set to 0.1 in all our experiments. This cost function is minimized with LIBLINEAR [Fan et al., 2008]. We perform three rounds of hard negative mining to refine the detector. The detector scores are calibrated with Platt's method [Platt, 1999] using jittered versions of the positive sample (100 positive and negative samples each). For the VOT2014 dataset, an additional step is introduced to train with rotated bounding box annotations. We estimated the rotation angle between one edge of the box and the corresponding image axis,[1] rectified the image with this angle, and extracted an axis-aligned box as the positive sample. The detector was evaluated at seven scales: $\{0.980, 0.990, 0.995, 1.000, 1.005, 1.010, 1.020\}$, in every frame. This is done densely in each scale, with a step size of 2 pixels. In the rotated bounding box case, we rectify the image with the angle estimated during training, before evaluating the detector. We set $k = 5$ to select the top 5 detection results and also (at most) 5 geometry proposals to form the candidate set. The best proposal box$_t$ selected from all the candidates in a frame is used to update the detector with a warm-start scheme [Fan et al., 2008, Supancic III and Ramanan, 2013]. We then perform hard negative mining and calibrate the scores to complete the detector update step. For computational efficiency, the detector is evaluated in a region around the previously estimated object location, instead of the entire image.

Hough voting. The geometric transformation is estimated only when the mean ℓ_2 norm of optical flow in box$_{t-1}$ is larger than 0.5. This ensures that there is sufficient motion in the frame to justify estimating a transformation.

We use two pixels in $t - 1$ and their corresponding points in t to vote for the geometric transformation parameters. In practice, pixels that are at least 25 pixels apart from each other in $t - 1$ were chosen randomly for a reliable estimation. To aggregate the votes, a pseudo-random hash function of bin values is used to vote into a one-dimensional hash table, instead of a four-dimensional array, similar to [Lowe, 2004]. The bin size for scale is set to 0.1, and 2.0 for angle and the two translation parameters. Finally, we take the least-squares solution of all the candidates that vote for a particular bin, making our geometry estimation further robust.

Pruning proposals. The parameters of the Kalman filter to discard weak geometry proposals are set as: $\alpha = 1$, $\beta = 1$, process error variance $q = 0.001$, and measurement error variance $r = 0.01$ in all our experiments. Edgebox scores are used to determine the best proposal when the detection scores are inconclusive, i.e., when one or more proposals differ from the best detection score by at most 1%. However, edgebox scores can often be corrupted by noise, e.g., low-quality images that are common in benchmark datasets. We ensure that these scores are used only when

1. Note that this estimation was done only in the first frame or whenever the tracker is restarted after a failure, i.e., when we have access to the ground truth box.

they are comparable to the mean score of the previous 5 frames, i.e., the difference between the highest edgebox score and the mean is less than twice the variance. This filtering step is performed separately for edgebox computed with object edge response and motion boundaries.

3.5 Experiments

We now present our empirical evaluation on two state-of-the-art benchmark datasets, namely, VOT2014 challenge dataset [Kristan et al., 2014] and OTB [Wu et al., 2013], and compare with several recent methods. These two datasets are evaluated with their respective toolkits, and the comparison results are obtained directly from the corresponding project websites.

3.5.1 VOT2014 Results

The results in Table 3.1 correspond to the baseline experiment in the VOT2014 toolkit and the overall rank is computed as described in §2.3.2. Overall, our proposal selection method using detector and geometry proposals ("Our-ms-rot" in the table) performs significantly better than DSST [Danelljan et al., 2014a], the winner of the VOT2014 challenge, with an average rank of 7.33 vs 10.39. We also evaluated two variants of our approach. The first one "Our-ms" uses only multi-scale detector proposals and the second variant "Our-ss"is limited to proposals from the detector evaluated only at a single scale. The overall rank of these two variants is lower than our full method: 7.43 and 15.29 respectively. The variant based on multi-scale detector proposals performs better on the accuracy measure compared to the full method, but is significantly worse on the robustness measure. In other words, this variant fails on many more frames than the full method, and benefits from the resulting reinitializations. Due to significant changes in scale that occur in several frames in the dataset, the variant based on a single-scale detector performs poorly. CMT [Nebehay and Pflugfelder, 2014], the key point based tracker which estimates rotation and scale of the object, has an average rank of 24.43. With the evaluation protocol of reinitializing the tracker when it fails, we found that using edgebox measure did not show a significant difference compared to the detection score. Thus, to keep the comparison with respect to the large number of trackers (38) reasonable, we show a subset of variants of our methods on VOT2014.

No.	Method	Acc.	Robust.	Avg.
1	Our-ms-rot	6.07	8.58	7.33
2	Our-ms	4.73	10.13	7.43
3	DSST [Danelljan et al., 2014a]	6.78	13.99	10.39
4	SAMF [Li and Zhu, 2014]	6.46	15.65	11.06
5	DGT	12.67	10.13	11.4
6	KCF [Henriques et al., 2015]	6.16	16.71	11.44
7	PLT$_{14}$	16.04	6.98	11.51
8	PLT$_{13}$	19.74	4.00	11.87
9	eASMS	15.37	15.10	15.24
10	Our-ss	16.11	14.47	15.29
11	ACAT	15.10	16.78	15.94
12	MatFlow	23.82	9.67	16.74
13	HMMTxD	11.08	22.51	16.8
14	MCT	18.47	15.14	16.81
15	qwsEDFT	19.06	21.15	20.11
16	ACT	22.68	18.10	20.39
17	ABS	22.34	20.49	21.42
18	VTDMG	23.22	19.94	21.58
19	LGTv1	31.05	12.68	21.87
20	BDF	24.92	19.39	22.15
21	aStruck	23.92	20.98	22.45
22	Struck [Hare et al., 2011]	22.44	22.96	22.70
23	DynMS	24.38	21.24	22.81
24	Matrioska	23.85	22.48	23.17
25	ThunderStruck	24.43	21.93	23.18
26	OGT	16.24	32.03	24.14
27	EDFT	22.12	26.44	24.28
28	CMT [Nebehay and Pflugfelder, 2014]	21.67	27.19	24.43
29	SIR_PF	26.43	22.59	24.51
30	FoT	21.16	28.42	24.79
31	LT_FLO	18.44	32.85	25.64
32	IPRT	29.48	24.33	26.90
33	IIVTv2	27.64	27.44	27.54
34	NCC	19.96	37.00	28.48
35	PTp	35.05	23.05	29.05
36	IMPNCC	28.49	30.32	29.40
37	FRT	26.23	33.14	29.68
38	FSDT	26.39	34.16	30.27
39	IVT	29.79	31.65	30.72
40	MIL [Babenko et al., 2011]	36.85	26.85	31.85
41	CT	34.38	30.54	32.46

Table 3.1 – Performance comparison of our methods and all the entries in the VOT2014 challenge. We show the ranking on accuracy (Acc.) and robustness (Robust.) measures, as well as the overall rank (Avg.). The top performer in each measure is shown in red, and the second and third best are in blue and green respectively. We show three variants of our method – Our-ms-rot: uses multi-scale detector and geometry proposals, Our-ms: only multi-scale detector proposals, and Our-ss: only single-scale detector proposals.

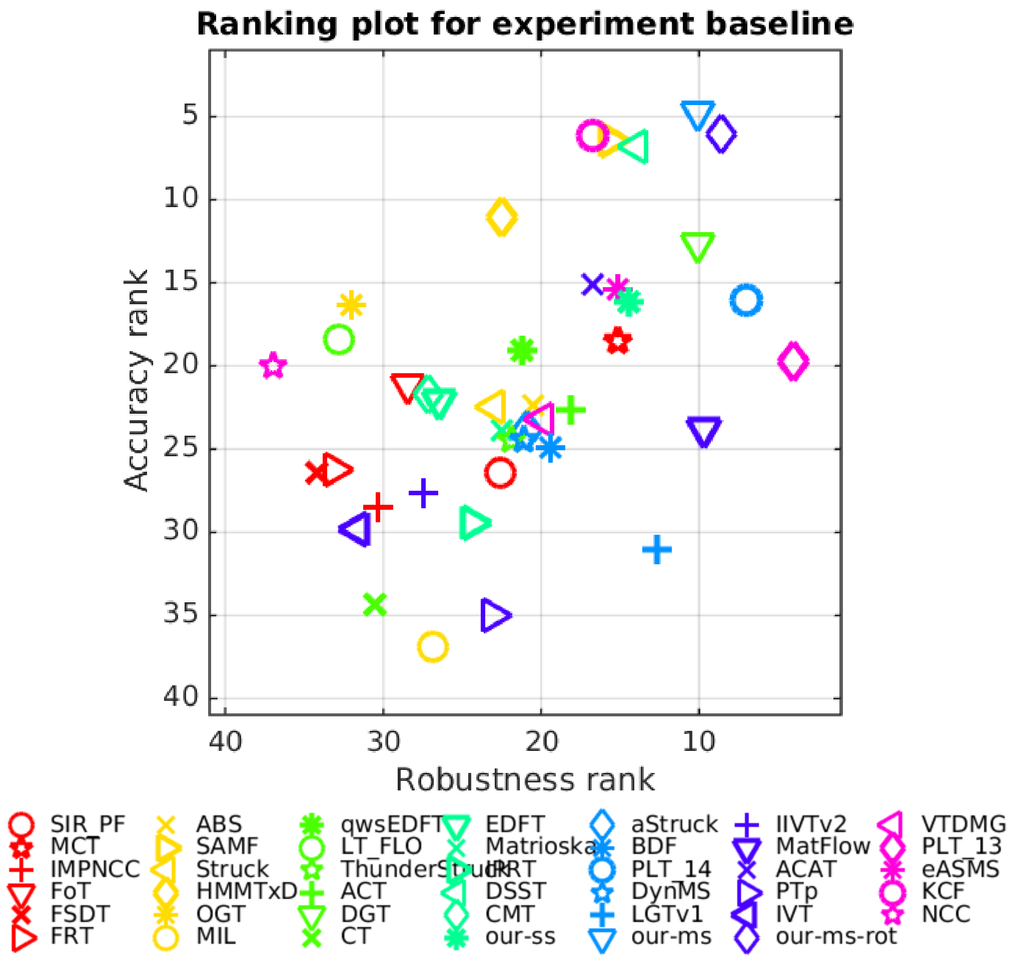


Figure 3.6 – AR ranking plot of all the trackers from the VOT2014 challenge and our methods. Trackers close to the top right corner of the plot are among the top performers.

On the accuracy measure, our approaches ("Our-ms-rot" and "Our-ms") are ranked first and second. KCF [Henriques et al., 2015] is third, with SAMF and DSST [Danelljan et al., 2014a] also performing well. The AR ranking plot in Figure 3.6 shows the similar performance of KCF, SAMF and DSST, where they form a cluster. KCF is a correlation filter based tracker. It is essentially a regression method trained with HOG features computed on several patches densely sampled around the object. This method, however, is significantly inferior on robustness measure (16.71 compared to our result 8.58) as it cannot handle object deformations other than translation. SAMF and DSST are extensions of the correlation

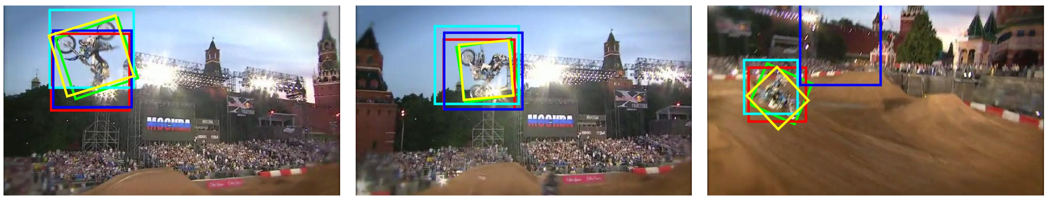


Figure 3.7 – Sample result on the *motocross* sequence in VOT2014 challenge dataset. Ground truth: green, Our result: yellow, DSST [Danelljan et al., 2014a]: red, PLT$_{14}$: cyan, Struck [Hare et al., 2011]: blue.

filter-based methods. SAMF proposes an adaptive scale for the patches and also integrates HOG and color attribute features. It is more robust than KCF, but still poorer than our result (15.65 vs 8.58). DSST [Danelljan et al., 2014a], the VOT2014 challenge winner, improves the correlation filtering scheme by estimating the scale of the target and combining intensity and HOG features. It trains two separate filters: one for estimating translation and another for scale. While DSST shows better performance than both SAMF and KCF, it also cannot handle cases where the target is rotating. Our method addresses this issue and significantly improves over DSST (8.58 vs 13.99 in robustness rank).

Our tracker is ranked third on the robustness measure. PLT$_{13}$ and PLT$_{14}$, which are two variants of Struck [Hare et al., 2011], are first and second respectively. PLT$_{13}$, the winner of the VOT2013 challenge, uses object-background color histograms to weight the features within the bounding box during SVM training. In essence, it attempts to refine the object representation from an entire bounding box to an object segmentation. PLT$_{14}$ is the multi-scale version of PLT$_{13}$, and it shows better accuracy at the cost of a lower robustness. The segmentation cue used in these methods makes them accurate because it provides a more precise object representation than a bounding box. However, in many situations, e.g., fast motion, lack of strong object-background color priors, it is likely to fail. This is evident from the significantly lower accuracy of the PLT methods – 16.04, 19.74 compared to 6.07 of our approach (see Table 3.1).

We present a sample qualitative result in Figure 3.7. The *motocross* sequence is considered as one of the most challenging sequences in this benchmark, based on the number of trackers that fail on it. Our approach performs significantly better than other methods. We compare to the results of DSST and PLT$_{14}$, which performed well on the VOT2014 challenge, and also Struck, the top performer on several datasets.

3.5.2 OTB Results

The results in Figure 3.8 correspond to the one-pass evaluation (OPE) discussed in §2.3.2. We show success and precision plots of our method in Figure 3.8 and compare it with the top 10 methods. Our overall method "Our-ms-rot-em" and its four variants outperform all the trackers on both the measures. More precisely, our tracker achieves 0.798 precision score compared to 0.656 of Struck [Hare et al., 2011], and 0.580 success score compared to 0.499 of SCM [Zhong et al., 2012] — the top two performers on this benchmark dataset [Wu et al., 2013]. We also show precision and success plots on the 11 attributes of the OTB dataset in Figures 3.9 and 3.10. These include attributes, such as "fast motion", "background clutter", "motion blur", "deformation", "illumination variation", "in-plane rotation", "low resolution", "occlusion", "out-of-plane rotation", "out of view", and "scale variation".

Our single-scale tracker "Our-ss" is inferior to the other four variants: 0.733 precision score and 0.523 success score, due to significant changes in scale and/or rotation in several sequences. The performance of our multi-scale version without geometry proposals "Our-ms", and the multi-scale one with geometry proposals "Our-ms-rot" is very similar. This is expected, despite "Our-ms-rot" performing better than "Our-ms" on the VOT2014 dataset, because the ground truth annotations in the OTB dataset are axis-aligned and not rotated boxes, as is the case in VOT2014. Thus, a rotating object can be enclosed (although imprecisely) by a larger axis-aligned box. This makes "Our-ms" as effective as "Our-ms-rot" following the OTB evaluation protocol. In Figure 3.8 we also show the influence of scores computed with edges "e" and motion boundaries "m". The precision result using (multi-scale) detector confidence alone to select from the candidate set ("Our-ms-rot") is 0.754. Incorporating the measure computed with edges ("Ours-ms-rot-e") improves this by 2.1% to 0.770, and the cue computed with motion boundary ("Our-ms-rot-em") gives a further 3.6% improvement to 0.798. A similar improvement pattern can be seen on the success plot.

A couple of recent papers [Hua et al., 2014, Supancic III and Ramanan, 2013] also evaluated their trackers on the OTB dataset. However, they are not online trackers, in that they use the information for subsequent frames to infer the object location in the current frame. We, nevertheless, compare to them using mean F_1 score (with 50% overlap threshold), following the protocol in [Hua et al., 2014]. Our tracker ("Our-ms-rot-em") performs significantly better with a score of 0.757 compared to 0.657 and 0.661 of

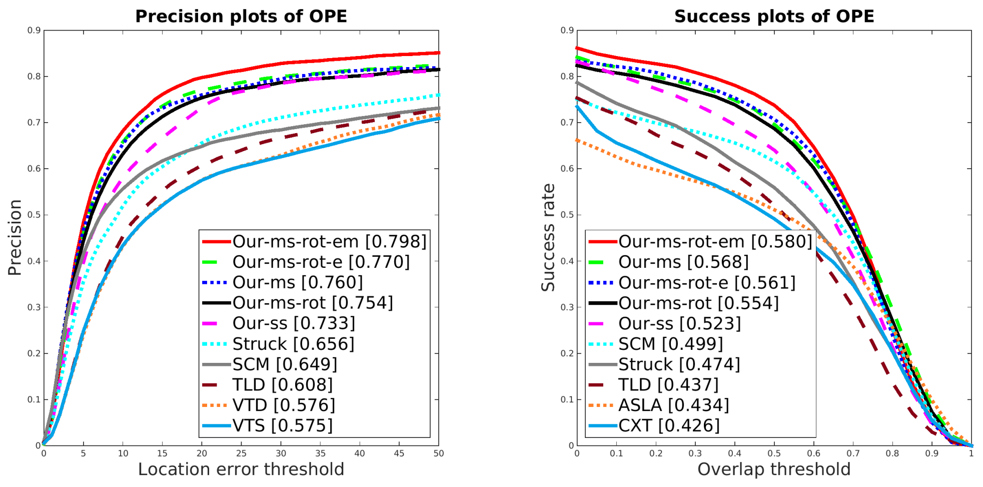


Figure 3.8 – Precision plots (left) and success plots (right) of OPE on the OTB dataset. For clarity we compare to the top 10 algorithms in each figure. The precision plots are summarized with the precision score at 20 pixel error threshold and the success plots with the area under the curve. These values are shown in the legend.

[Hua et al., 2014] and [Supancic III and Ramanan, 2013] respectively.

Qualitative results. In Figure 3.11, we compare our results with those of the top three methods from the standard evaluation toolkit [Wu et al., 2013]. For the CarScale sequence in Figure 3.11(a), our method (shown in yellow) successfully estimates the change in scale showing the best performance over the entire sequence. Struck (magenta) cannot cope with change in scale. TLD (red) uses a set of fixed-size template patches as the object model, which is restrictive, and cannot handle these severe changes. SCM tracker (cyan) suffers from lack of sufficient model updates as the appearance of the object changes significantly. Trackers using a dense sampling approach, i.e., Struck (magenta), TLD (red), get stuck on background regions in the David3 sequence shown in Figure 3.11(b). Once again SCM (cyan) cannot handle the appearance changes, when the person is partially occluded by the tree. On the contrary, our approach (yellow) continues to track the person.

For the Dog1 sequence in Figure 3.11(c), SCM (cyan) provides an accurate location of the object, but fails to estimate its scale precisely. In contrast, our tracker (yellow) has a near-perfect overlap with the ground truth. For the Dudek sequence in Figure 3.11(d) all the three other trackers

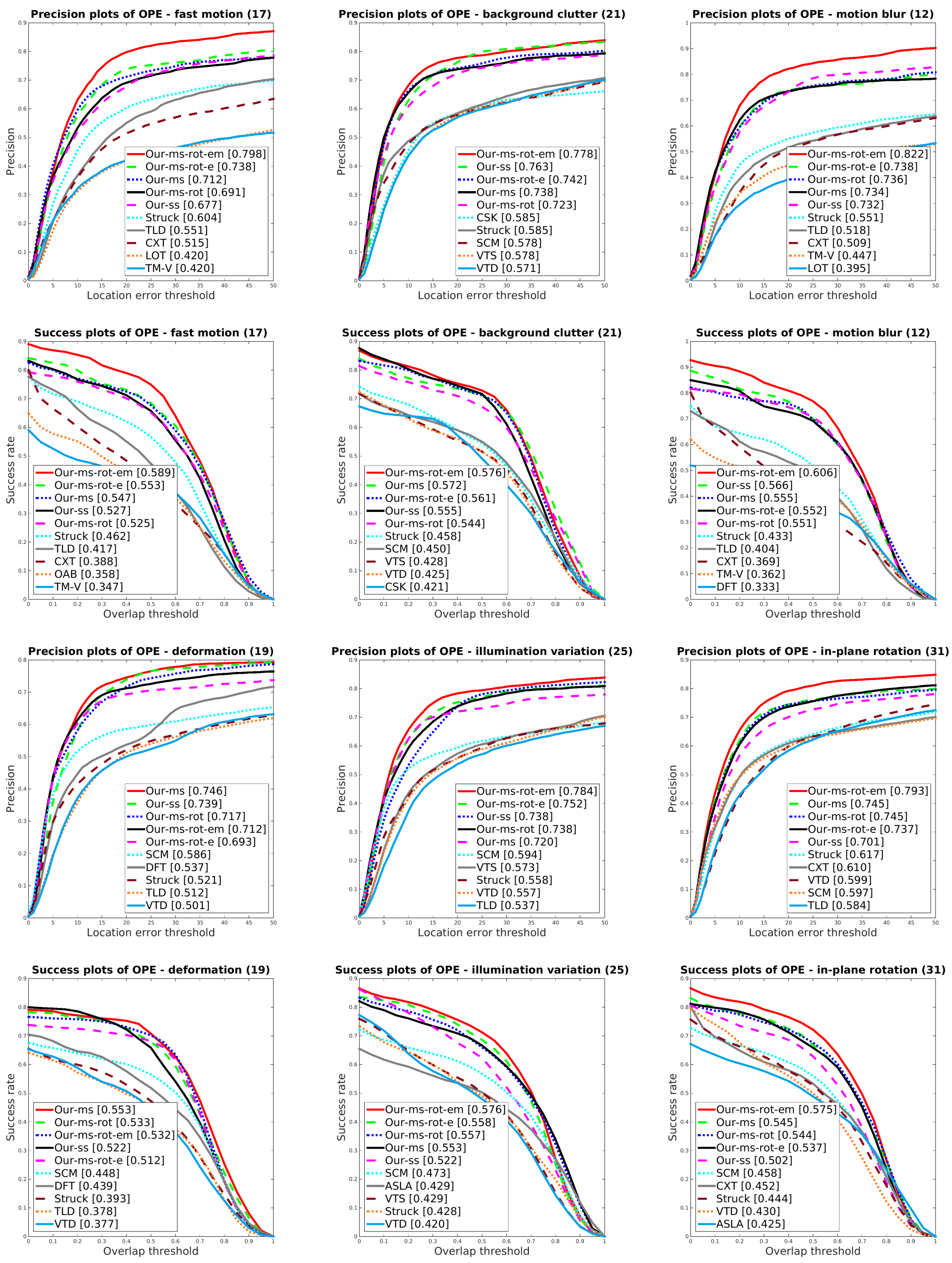


Figure 3.9 – Precision and success plots on the 11 attributes of the OTB dataset (part 1). For clarity we show the top 10 algorithms in each figure. The precision plots are summarized with the precision score at 20 pixel error threshold and the success plots with the area under the curve. These values are shown in the legend.

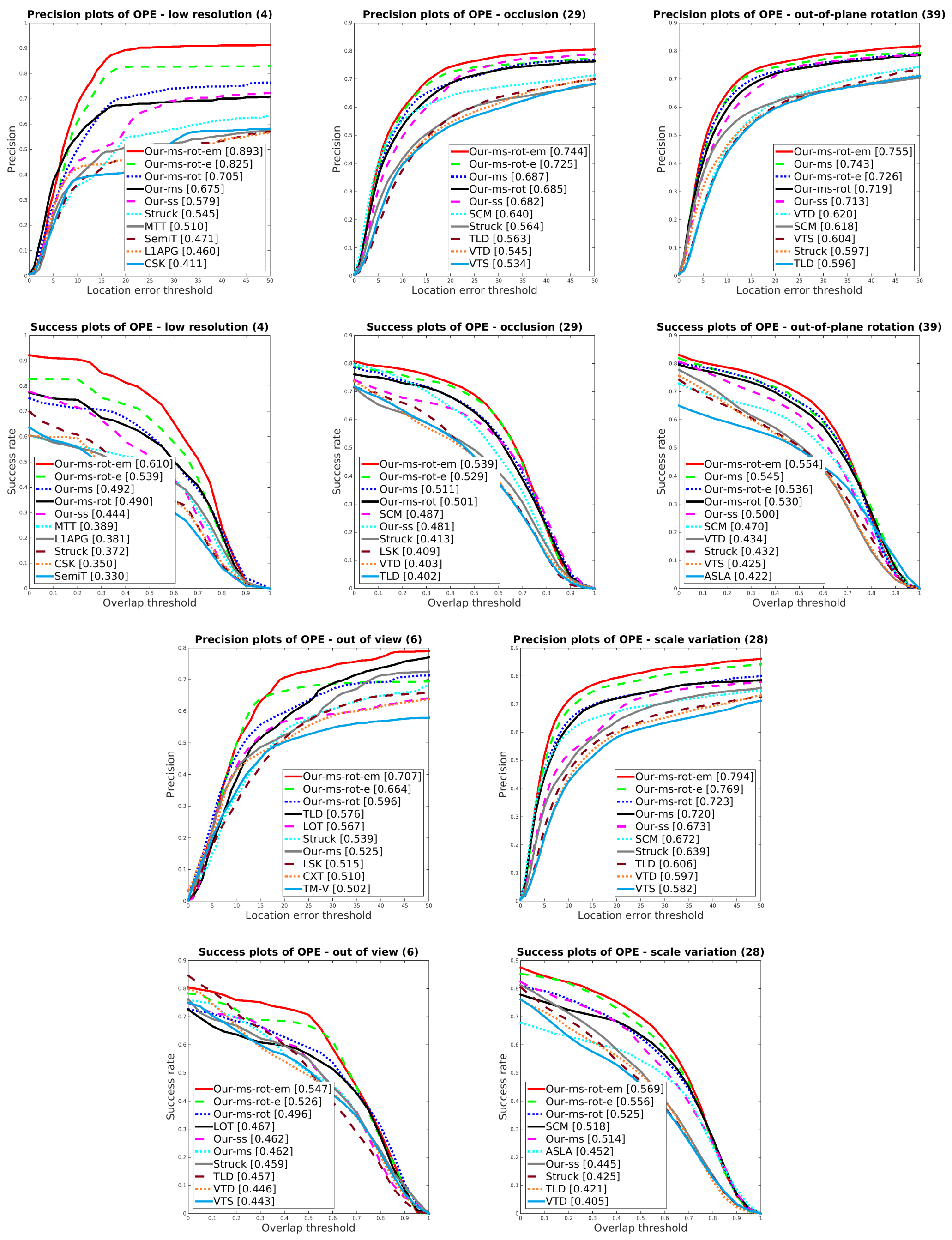


Figure 3.10 – Precision and success plots on the 11 attributes of the OTB dataset (part 2). For clarity we show the top 10 algorithms in each figure. The precision plots are summarized with the precision score at 20 pixel error threshold and the success plots with the area under the curve. These values are shown in the legend.

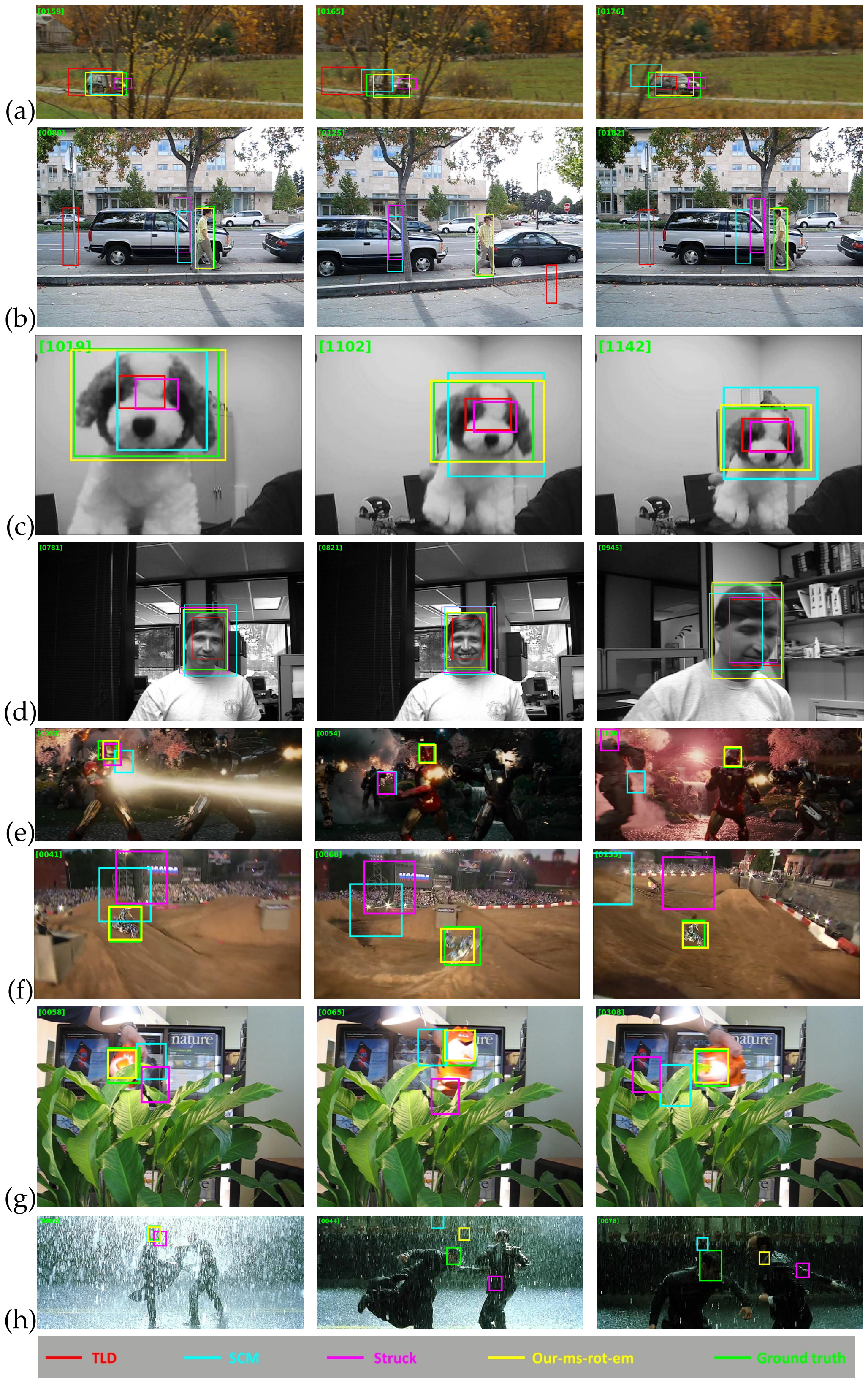


Figure 3.11 – Tracking results of our method ("Our-ms-rot-em" shown in yellow) on CarScale, David3, Dog1, Dudek sequences from the OTB dataset. We also show the ground truth (green) and results of SCM (cyan), Struck (magenta) and TLD (red) for comparison.

(cyan, magenta, red) drift onto a part of the object as they lack an accurate rotation angle estimation (see column 3 in the figure), unlike our tracker (yellow). The three trackers (SCM, Struck, TLD) also fail on the Ironman example shown in Figure 3.11(e), which has significant illumination changes, object undergoing geometric transformations and/or becoming occluded. Our framework (yellow) overcomes these challenges.

For the MotorRolling sequence in Figure 3.11(f), given the significant change in appearance of the object, TLD (red) cannot re-start the tracker successfully when it fails. Struck (magenta) remains in a region around the initial location of the object due to its dense sampling scheme. The restrictive update used in the SCM tracker (cyan), i.e., only updating negative samples in the discriminative model and object template histogram in the generative model, is also ineffective. Our estimation of geometry transformations to track and then update the model performs best. In the case of the Tiger1 sequence shown in Figure 3.11(g), TLD (red), Struck (magenta) and SCM (cyan) track the object successfully in several frames, but are less effective when the object undergoes motion blur or partial occlusion. All the trackers perform poorly on the Matrix sequence shown in Figure 3.11(h). This sequence is particularly challenging, with imaging conditions changing dramatically from the frames at the beginning of the sequence (column 1: heavy rain) to subsequent frames (columns 2 and 3: little to almost no rain). Struck and TLD suffer from drift in frame 2 (column 1). Our method and SCM perform better than these two, and track the object at the beginning of the sequence, but drift or miss it later on (columns 2 and 3).

In Figure 3.12, we compare our complete method “Our-ms-rot-em”with four variants on two sample sequences – Lemming and Skating respectively. For the Lemming sequence, depicted in Figure 3.12(a-c), we observe that the full method (shown in yellow) and the variant using object edge cues (“Our-ms-rot-e”shown in magenta) can handle partial occlusion of the object (column 2 in (a)), while the three other variants, which rely solely on detection scores to select from the proposal set, cannot. The strength of the object edges and motion boundaries are shown in rows (b) and (c) respectively. We observe a similar behavior in the Skating sequence, shown in Figure 3.12(d-f), where the two versions using edgebox scores perform the best.

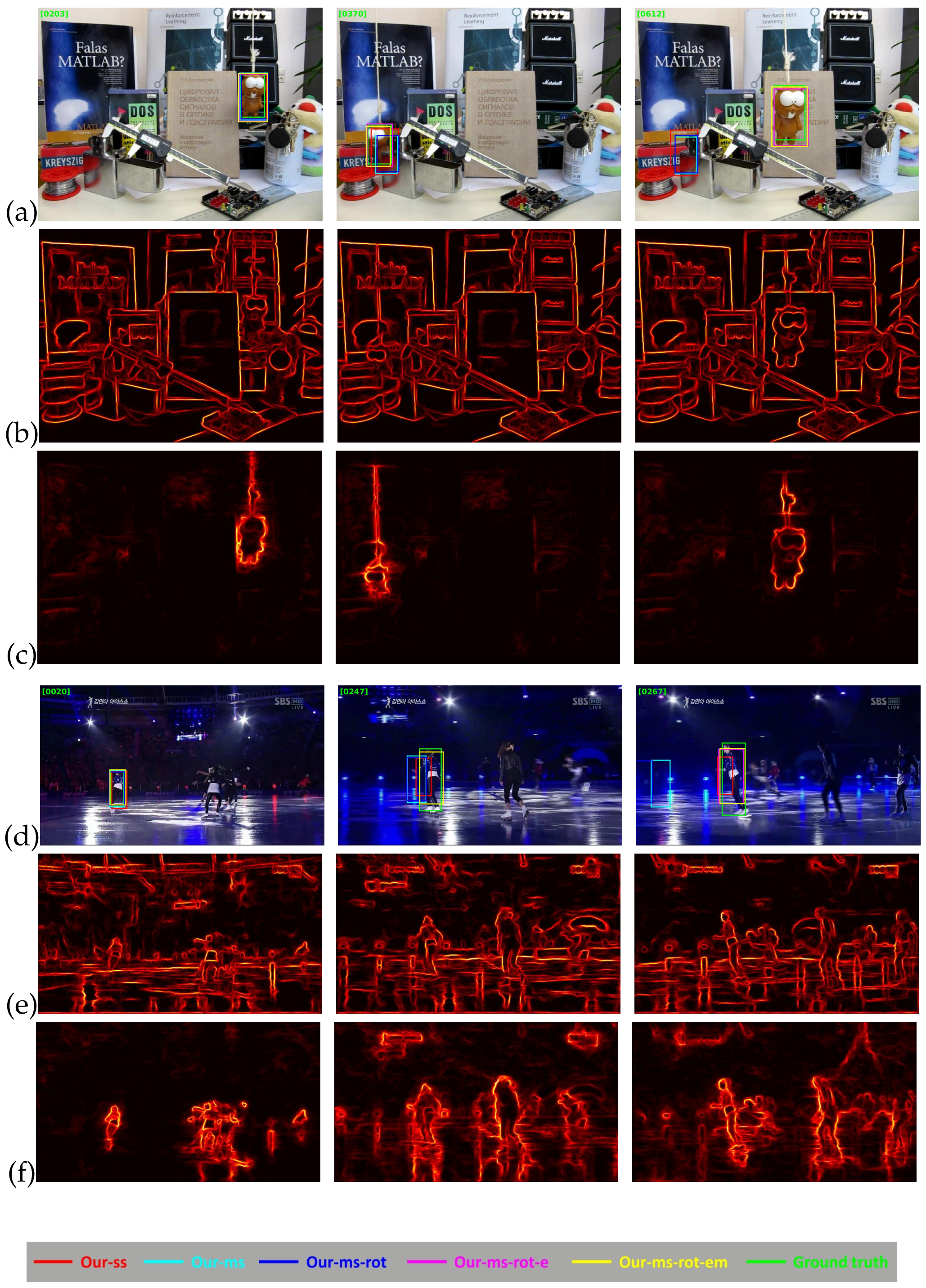


Figure 3.12 – Comparison of our full method “Our-ms-rot-em” with four variants on Lemming and Skating sequences – “Our-ss”: Single scale detector proposals only, “Our-ms”: Multi-scale detector proposals, “Our-ms-rot”: Multi-scale detector + geometry proposals, “Our-ms-rot-e”: Multi-scale detector + geometry proposals + edgebox score from object edges. The full method additionally uses edgebox score from motion boundaries with “Our-ms-rot-e”. (a, d) Tracking results on sample frames. (b, e) Object edge responses. (c, f) Motion boundaries.

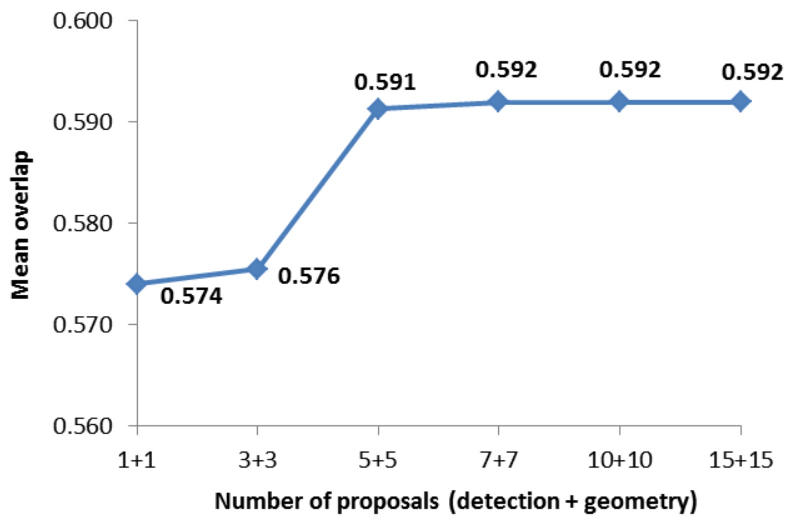


Figure 3.13 – Influence of the number of (detection and geometry) proposals on the tracking performance on the OTB dataset.

3.5.3 Discussion

Influence of the number of proposals. In Figure 3.13 we evaluate the performance of our approach ("Our-ms-rot-em") with respect to the number of candidate proposals on the OTB dataset. Here, we measure the performance as mean overlap between the predicted and the ground truth bounding boxes over the entire sequence. We observe that the mean overlap score increases initially with the number of proposals, but then quickly saturates. In other words, we require a minimum number of proposals (5 each for detection and geometry, which we use in this chapter) for tracking an object effectively. Adding further candidate proposals beyond this does not affect the performance, as the new candidates are either very similar to or less accurate than those already in the proposal set.

Computation time. Our Matlab and mex implementation is un-optimized and not real-time. It performs as follows (average fps on VOT2014+OTB datasets): our-ss: 6.4, our-ms: 4.1, our-ms-rot: 3.1, our-ms-rot-e: 2.1, our-ms-rot-em: 1.9 on a cluster node with 20 cores. The last three variants require pre-computed optical flow [Brox and Malik, 2011], which runs at 0.3 fps on a GPU. Our implementation can be sped up in several ways, e.g., by scaling the images down, improving the feature extraction process.

3.6 Summary

This chapter presents a new tracking-by-detection framework for online object tracking. Our approach begins with building a candidate set of object location proposals extracted using a learned detector model. It is then augmented with novel proposals computed by estimating the geometric transformation(s) undergone by the object. We localize the object by selecting the best proposal from this candidate set using multiple cues: detection confidence score, edge responses and motion boundaries. The performance of our tracker is evaluated extensively on the VOT challenge and the OTB datasets, and we show significant improvement over the top performers.

The proposal-selection framework is flexible and can be adopted to different user scenarios by adding or changing the source of proposals and the strategy of selection. For instance, in order to handle extreme deformable objects, proposals from object segmentation can be added.

Chapter 4

Occlusion and Motion Reasoning for Long-term Tracking

Contents

Although discriminative tracking-by-detection approaches, in particular Struck [Hare et al., 2011], have shown competitive results, they perform poorly in the presence of long-term occlusions, as well as severe viewpoint changes of the object. In this chapter, we propose a principled way to combine occlusion and motion reasoning with a tracking-by-detection approach. Our occlusion and motion reasoning is based on state-of-the-art long-term trajectories which are labeled as object or background tracks with an energy-based formulation. The overlap between labeled tracks and detected regions allows to identify occlusions. Further, the tracks are used to estimate the geometric transformations of the target over time. Experimental results show that our tracker obtains state-of-the-art results, and handles occlusion and viewpoint changes better than competing tracking methods.

4.1 Introduction

Many of the previous works have approached the problem of tracking from the perspective of either a motion tracker or an appearance tracker. As discussed in previous chapters, tracking-by-detection, i.e., approaches that treat the tracking problem as a task of detecting the object over time [Avidan, 2007, Grabner et al., 2008, Babenko et al., 2011, Mei and Ling, 2009, Godec et al., 2011, Hare et al., 2011, Kalal et al., 2012, Supancic III and Ramanan, 2013], has become an increasingly popular method for object tracking. In fact, the latest evaluation papers [Wu et al., 2013, Pang and Ling, 2013] have shown such an approach, namely Struck [Hare et al., 2011], to be the best-performer on a diverse set of examples. One of the critical steps in these methods is to update and improve the object model over time. Consider the example in Figure 4.1. It shows a scenario where the object of interest — a car — is occluded when it goes under a bridge. If the model is updated in every frame, this results in the well-known issue of drifting [Matthews et al., 2004]. In other words, a part of the bridge might be tracked instead of the car in the latter frames in the sequence. In this section we present an algorithm which can handle such occlusions as well as significant viewpoint changes in a principled way, based on state-of-the-art quasi-dense long-term trajectories [Ochs et al., 2014, Sundaram et al., 2010]. These trajectories rely on dense optical flow [Brox and Malik, 2011] in combination with a forward-backward matching criterion. Sundaram et al. [2010] showed that they significantly outperform the Kanade-Lucas-Tomasi (KLT) tracker [Lucas and Kanade, 1981], often used in tracking approaches [Everingham et al., 2009, Kalal et al., 2012].

Our main contribution is to use these long-term trajectories in combination with graph-cut based track labeling to identify the state of the object, e.g., no, partial or full occlusion, as well as change in viewpoint, and to choose and adapt positive and negative object samples accordingly to improve the model.

The goal of this chapter is to track the object, given a bounding box initialization in the first frame. Our approach begins by learning an initial

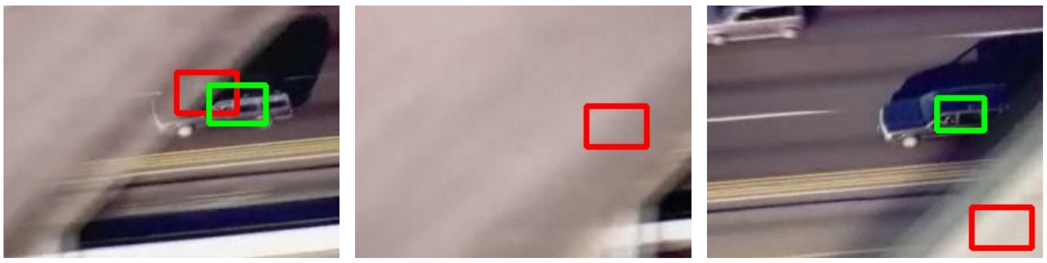


Figure 4.1 – Three sample frames from the Carchase sequence in the TLD dataset [Kalal et al., 2012]. Each image also shows the result of our method (in green) and Struck [Hare et al., 2011] (in red). As the car starts to move to a severely occluded state, updating the appearance model with the prediction result leads to model drift, as shown by the Struck result. Our proposed method estimates such states, and does not update the model in such cases.

appearance model from the annotation in the first frame, similar to [Hare et al., 2011, Kalal et al., 2012, Supancic III and Ramanan, 2013], and uses it to propose candidate object locations in the latter frames. We incorporate motion cues to refine the candidate region search space and avoid incorrect object proposals (Section 4.3). In order to determine whether a candidate location in a frame contains the object, we compute long-term motion tracks in the video [Sundaram et al., 2010], and use them to predict the state of the object, i.e., the transformation it has undergone with respect to the previous frames (Section 4.4). More specifically, we estimate states such as, no, partial or full occlusion, change in appearance of the object. This is achieved with an energy-based formulation, where the task is to assign each track an object or background label. When a significant part of the tracks within a candidate region belong to the background, the object is identified to be occluded. We will show that other types of change in state, such as a significant change in viewpoint, can also be estimated with our formulation. With this additional cue in hand, we build a temporally-evolving object model which deals with these state changes by updating the initially learned detector accordingly (See §4.4.2). In essence, our tracker proposes a new way to interleave the motion-based and tracking-by-detection paradigms. Its performance is evaluated on sequences from a recent benchmark dataset [Wu et al., 2013] and video sequences used in [Kalal et al., 2012, Supancic III and Ramanan, 2013] (Section 4.6).

4.2 Related Work

A variant of the tracking-by-detection approach, known as adaptive discriminative tracking-by-detection, updates the object model over time [Grabner et al., 2008, Babenko et al., 2011, Hare et al., 2011, Kalal et al., 2012, Supancic III and Ramanan, 2013]. It typically consists of two phases: (i) tracking, where the detector is used to predict the object location in each frame; and (ii) learning, where the estimated object locations (in the form of bounding boxes) are used to generate training examples to improve the learned object model. Observing that many of the adaptive methods lack a principled way of generating the training samples, a joint structured output formulation (Struck) was proposed in [Hare et al., 2011]. This work learns a function to predict the object location in a frame based on its location in the previous frame and a predefined search space. The prediction function is a dot product of a weight vector and a joint kernel map. This weight vector is learned and updated with an online structured output framework. Our approach is based on a similar philosophy, in that, we learn and update a prediction function. However, we use state-of-the-art long-term motion tracks [Sundaram et al., 2010] to determine the state of the object and produce an effective set of training exemplars for the update step. In the example shown in Figure 4.1, we predict that the object is severely occluded in the middle frame and thus, do not update our detector. Note that Struck (result shown in red in Figure 4.1) drifts onto a part of the bridge in this example.

Our method is also closely related to two other recent approaches [Kalal et al., 2012, Supancic III and Ramanan, 2013] on long-term tracking. The TLD algorithm [Kalal et al., 2012] aims to combine the benefits of tracker- and detector-based approaches. It decomposes the tracking task into specialized components for tracking, learning and detection, which are run simultaneously. The result of this algorithm is a combination of predictions from the frame-to-frame tracking based on median optical flow and a detection component. The two components mutually update each other. Specifically, the results of the tracker provide training data to update the detector, and the detector re-initializes the tracker when it fails, for example, when the object is occluded or leaves the field of view. While this is an interesting approach, it is somewhat restrictive as the object model is a set of fixed-size template patches, i.e., TLD cannot handle severe changes in object size, such as the scenario shown in Figure 4.2. Furthermore, the motion information is limited to frame-to-frame constraints. Supancic III and Ramanan [2013] presented a tracking-by-detection method, where

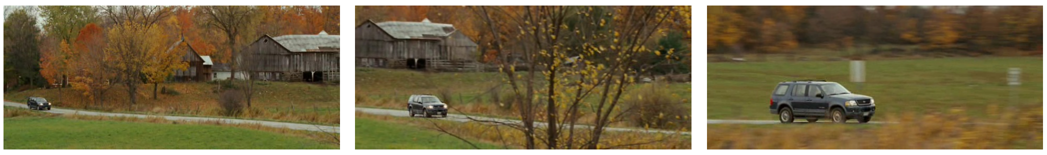


Figure 4.2 – Three frames from the CarScale sequence in [Wu et al., 2013] are shown to highlight the severe change in object size over time in some of the videos.

the detector is updated using a pre-fixed number of frames, i.e., the top-k frames chosen according to an SVM objective function, irrespective of the state of the object. This does not handle long-term occlusions. In contrast, our algorithm updates the model with only the frames that show a significant presence of the object, as it relies on long-term motion cues to choose the training exemplars, unlike [Supancic III and Ramanan, 2013] which uses only the detector.

We experimentally compare with these related works [Hare et al., 2011, Kalal et al., 2012, Supancic III and Ramanan, 2013], and show the benefits of our approach in Section 4.6.

4.3 Overview

In line with the tracking-by-detection approach, our tracker comprises three stages. First, a detector is learned from a given training set of positive and negative exemplars. Second, we track with this learned detector. Third, we update the object model with a carefully selected subset of frames. We now present an overview of these stages and then provide more details in Sections 4.4 and 4.5.

Initial detector and tracking. An initial object detector is required to set off our tracker. It is learned with a training set, where the positive example is the ground truth annotation in the first frame of the sequence, and the negative samples are harvested from the frame, similar to [Supancic III and Ramanan, 2013]. The initial model is then learned with HOG features [Dalal and Triggs, 2005, Felzenszwalb et al., 2010] extracted from the bounding boxes, and a linear SVM.

The detector is used to predict candidate locations of the object in other frames. In each frame, we find the most likely location of the object by evaluating the detector in a region estimated from motion cues

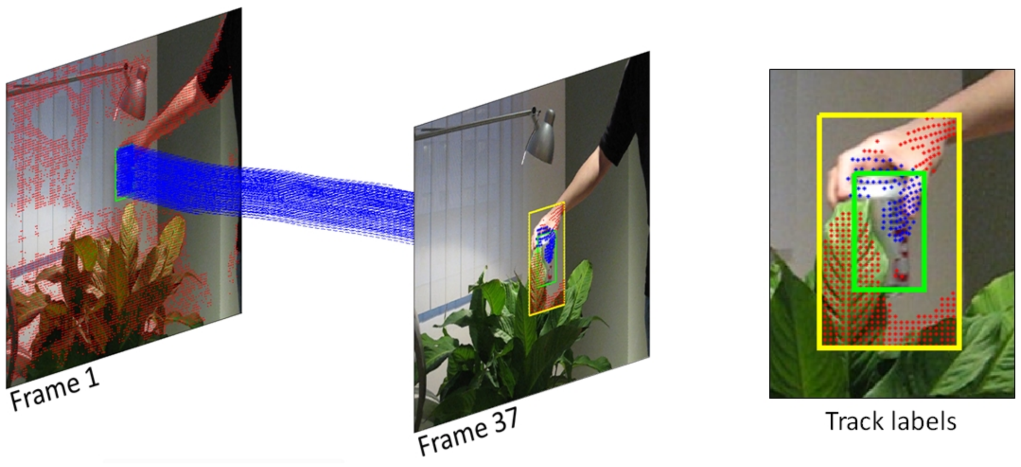


Figure 4.3 – *Left*: Long-term tracks beginning in frame 1 of the Coke sequence [Wu et al., 2013]. The yellow box shows the search region used to compute the bounding box most likely to contain the object (green box). We use tracks to estimate the object state. *Right*: Close-up of the track labels in frame 37. Here, less than 60% of the tracks within the predicted bounding box are assigned to the object (blue), and the remaining are labeled as background (red). Thus, the object is predicted to be in an occluded state.

(optical flow computed from the previous frame), and then choosing the bounding box with the best detection score, as shown in Figure 4.3. The motion-refined search is not only computationally efficient, but also avoids incorrect detections due to background clutter. Note that the bounding box obtained from this step is not labeled as the object yet.

We compute and analyze the motion cues to make the object label assignment in each frame. To this end, we extract long-term point tracks which extend over many frames in the video [Sundaram et al., 2010], see Figure 4.3-Left. At this stage, we discard tracks less than 5 frames long, which are typically less reliable. We then propose an energy-based formulation to label the remaining tracks as object or background. This is related to the labeling framework used in [Lezama et al., 2011] for motion-clustering tracks. The tracks within the bounding box in the first frame, i.e., the ground truth annotation, are initialized with the object label, and those that lie outside are given the background label. With these initial assignments and pairwise energy terms (which measure track similarity), we optimize the energy function and label all the new tracks, i.e., tracks that begin in the second or latter frames, see Figure 4.3-Right.

Occlusion. If a significant part (40% or more) of the tracks within the bounding box take the background label (as in Figure 4.3), we consider the object to be in an occluded state. In this case, the object model is not updated with the detection result. We continue to track the object with the non-updated detector as long as there are object tracks and a detection response. This step avoids model drift [Matthews et al., 2004, Supancic III and Ramanan, 2013]. For example, in the sequence in Figure 4.1, the model is not updated with the frame shown in the middle, to avoid the tracker drifting onto a part of the bridge, which occludes the car. To handle cases where the object re-appears after a full occlusion (e.g., the frame on the right in Figure 4.1), the detector is evaluated over the entire image in subsequent frames.

Temporally-evolving detector. When the object is not occluded in a frame, the long-term tracks are used to measure geometric transformations that the object may have undergone, such as change in scale, rotation (see Figures 4.2 and 4.4). In this work, we approximate these transformations with a similarity matrix [Hartley and Zisserman, 2004], and estimate it with track-point correspondences between consecutive frames. The bounding box is then refined with the estimated transformation and is assigned the object label. This is illustrated on an example in Figure 4.4. Based on the severity of the transformation, we either: (i) update the existing detector; or (ii) learn a new detector. In summary, our detector model evolves temporally to account for changes in object appearance by updating, i.e., learning, with new positive instances.

4.4 Motion Cues in the Tracker

Motion cues serve two purposes in our algorithm: (1) to determine the search region for evaluating the detector; and (2) to estimate the state of the object. We use dense optical flow computed between two frames for the first task, and cues extracted from long-term tracks for the second.

Given the bounding box labeled as the object in a frame, we compute optical flow [Brox and Malik, 2011] for all the pixels within the box, and obtain the median flow. With this flow estimate, the bounding box is translated onto the following frame, and the area surrounding it (an enlarged box) is considered as the search region for the detector. In other words, we restrict the search space for the object detector when finding the most likely location of the object in a new frame. An example is illustrated in Figure 4.3. This useful cue is inspired by many of the traditional motion-based trackers, but is limited to providing only local motion information. We argue that these local cues are insufficient to reliably estimate (e.g., when the optical flow measurements are poor) whether the new bounding box contains the object or not. Our work integrates richer cues computed from long-term tracks into the framework to make a robust estimation of the state of the object. We achieve this with an energy-based formulation involving the long-term tracks.

Each track is represented with a random variable X_i and takes a label $x_i \in \{0, 1\}$, where 0 denotes the background and 1 is the object. Let n denote the number of tracks, and $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ be the set of random variables. A labeling $\mathbf{x}$ refers to any possible assignment of labels to the random variables, and takes values from the set $\{0, 1\}^n$. The cost of a label assignment $E(\mathbf{X} = \mathbf{x})$, or $E(\mathbf{x})$ in short, is defined as:

$$E(\mathbf{x}) = \sum_{i=1}^{n} \phi_i(x_i) + \lambda \sum_{(i,j)\in\mathcal{E}} \phi_{ij}(x_i, x_j), \tag{4.1}$$

where $\phi_i(x_i)$ is the unary term to measure how likely it is for the track i to take label x_i. The function $\phi_{ij}(x_i, x_j)$ is a smoothness term to encourage similar tracks to take the same label. The set of pairs of interacting tracks is denoted by $\mathcal{E}$, and λ is a parameter to regulate the relative strength of the unary and the pairwise terms. The energy function (4.1) is minimized exactly to obtain the globally optimal labels for the tracks.

The pairwise smoothness term takes the form of a generalized Potts model [Boykov and Jolly, 2001] and is given by:

$$\phi_{ij}(x_i, x_j) = \begin{cases} \exp(-\lambda_d d(i,j)) & \text{if } x_i \neq x_j, \\ 0 & \text{otherwise,} \end{cases} \tag{4.2}$$

where $d(i, j)$ measures the dissimilarity between the two tracks i and j and λ_d is a parameter set to 0.1. This term is defined between pairs of neighbouring tracks, and it assigns a low cost for two dissimilar tracks to take different labels. We use the popular dissimilarity measure [Brox and Malik, 2010] computed as the maximum distance between time-corresponding spatial coordinates $\mathbf{p}_t^i, \mathbf{p}_t^j$ and velocity values $\mathbf{v}_t^i, \mathbf{v}_t^j$ as:

$$d(i,j) = \max_t \ ||\mathbf{p}_t^i - \mathbf{p}_t^j||_2^2 \frac{||\mathbf{v}_t^i - \mathbf{v}_t^j||_2^2}{5\sigma_t^2}. \tag{4.3}$$

This maximum is computed over points in time where the two tracks overlap. The first term, $||\mathbf{p}_t^i - \mathbf{p}_t^j||_2^2$, measures the spatial Euclidean distance, and the second term is the vector difference of the velocities estimated over 5 frames, i.e., $\mathbf{v}_t^i = \mathbf{p}_i^{t+5} - \mathbf{p}_i^t$.

For the unary terms, all the tracks that begin within the ground truth annotation in the first frame are assigned a very high cost to take the background label. This prevents them from changing their label in the latter frames, and is essentially a hard assignment of the object label. Inversely, tracks that lie outside the annotation in the first frame are given a very high cost to take the foreground label. The hard label assignment within the ground truth annotation can be refined, for example, by assigning a subset of the tracks with the object label using Grabcut-like segmentation techniques [Rother et al., 2004]. We found this refinement step to be non-essential in our case, since the track labels are used in combination with other cues, and not directly to determine the object location. The unary term for any new tracks starting in the second frame and beyond is defined as:

$$\phi_i(x_i = 1) = \begin{cases} 1 - \frac{1}{1+\exp(-(\alpha_t d_t + \beta_t))} & \text{if track } i \in \text{box}_t, \\ 0.5 & \text{otherwise,} \end{cases} \tag{4.4}$$

and $\phi_i(x_i = 0) = 1 - \phi_i(x_i = 1)$. Here, d_t is the SVM detection score for box_t, the bounding box estimate in frame t. The scalars α_t and β_t map this score into a probabilistic output, and are computed using Platt's method [Platt, 1999]. The intuition behind this unary term is that new tracks within a strong detection are likely to belong to the object. For tracks that lie outside the detection box, we allow the pairwise similarity terms to decide on their labels by giving an equal unary cost for assigning object or background labels.

In order to minimize the energy function (4.1) we apply the min-cut/maxflow algorithm [Hammer, 1965, Kolmogorov and Zabih, 2004, Boykov and Kolmogorov, 2004] on a corresponding graph, where each track is represented as a node. All the tracks within the search region in frame t (shown as a yellow box in Figure 4.4) are added as nodes. Additionally, tracks labeled in the previous frames which continue to exist in the frame are added. The unary and pairwise costs, computed as described earlier, are added as weights on the edges in the graph. We then perform *st*-mincut on the graph to get the optimal labels for all the nodes. Building the graph and performing inference on it in every frame allows us to update the labels of existing tracks based on new evidence from neighboring tracks. An illustration of track labels is shown in Figure 4.3.

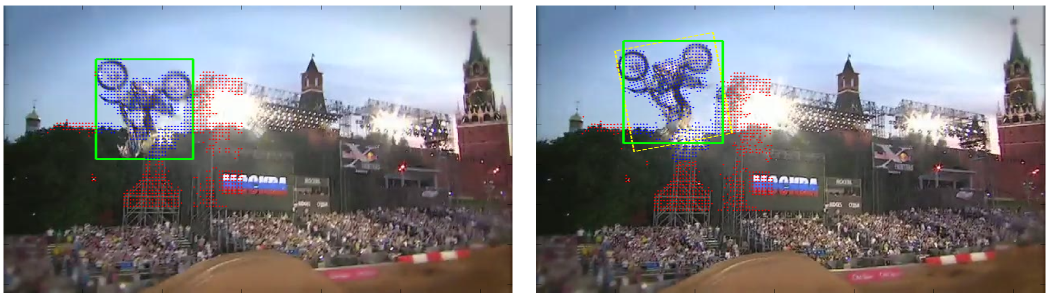
4.4.1 Predicting the State

With the track labels in hand, we determine whether the object has been occluded or a change in viewpoint has occurred. If more than 40% of the tracks within box_t belong to the background, it is marked as a partial occlusion. We identify a full occlusion of the object if more than 80% of the tracks are assigned the background label. In other cases where a majority of the tracks continue to belong to the object, we verify if there have been any other transformations, see Figures 4.2 and 4.4 for two such examples. We model these transformations with a similarity matrix. It is estimated with a RANSAC approach [Hartley and Zisserman, 2004], using points on the tracks (inside the box_{t-1}) in frames $t-1$ and t as correspondences. Since it is feasible to obtain more reliable point correspondences between consecutive frames, we compute frame-to-frame similarity matrices, and then accumulate them over a set of frames. For example, the transformation S_3^1, from frame 1 to 3, is computed as the product of the transformations S_3^2 and S_2^1. When a similarity matrix shows a significant change in scale or rotation, fixed empirically as 15% and 10° respectively in all the experiments, we mark the state as change in viewpoint.

To sum up, the candidate region box_t is labeled as occluded when a full occlusion state is predicted. When a change in viewpoint is estimated, box_t is transformed with the similarity matrix S_t^1 to obtain box_t^S, which is then assigned the object label. In other cases, i.e., neither occlusion nor change in viewpoint, box_t takes the object label.

4.4.2 Re-training the Model

Re-training (or updating) the model is crucial to the success of a tracking algorithm to handle situations where the object may change in some form over time. We use the predicted state of the object to precisely define the update step as follows.



(a) Frame 1 (b) Frame 4

Figure 4.4 – Sample frames from the MotorRolling sequence [Wu et al., 2013], where the object undergoes a significant transformation (rotates counter clockwise). (a) Frame 1, showing the bounding box in the first frame. (b) The result of our tracker is shown is green. We show the bounding box transformed with the estimated similarity in yellow.

Case 1: No change in state. The model update is straightforward, if the object is neither occluded, nor has undergone any of the other transformations. The new bounding box, box_t, is treated as a positive exemplar, and is used to update the SVM classifier.

Case 2: Occlusion. When the object is in a (partial or fully) occluded state, the classifier is not updated.

Case 3: Change in viewpoint. The detection result box_t in this case is transformed with the estimated similarity matrix to box_t^S. We then fit an image-axes-aligned box that encloses box_t^S, as illustrated in the example in Figure 4.4. This transformation changes either the scale or the aspect ratio of the bounding box containing the object. Recall that our initial detector is trained from a single positive example at one scale, and adding other samples with different scales (or aspect ratios) will deteriorate it. We choose to train a new detector with the new bounding box in frame t, and maintain a set of detectors which capture various scales and aspect ratios of the object, as it changes over time. This idea of maintaining multiple detectors for an object category is similar in spirit to exemplar SVMs [Malisiewicz et al., 2011].

A summary of our method is given in Algorithm 2. Note that in the case of a full occlusion, the best detection is obtained by evaluating the detector over the entire image. If the score of the best detection result in

Algorithm 2 : Our approach for tracking an object and estimating its state.

```
Data: Image frames $1 \dots n$, object location $\text{box}_1$ in frame 1
Result: Object location $\text{box}_t$ and $\text{state}_t$ in frames $t = 2 \dots n$
Learn initial detector in frame 1 (§4.3)
Compute long-term tracks
for $t = 2 \dots n$ do
    if $\mathit{state}_{t-1}$ is Full occlusion then
        $\text{box}_t$, $\text{score}_t \leftarrow$ Best detection over the entire frame $t$
        if $\mathit{score}_t > \mathit{restart\_threshold}$ then
            $\text{state}_t \leftarrow$ No change
            Update detector model (§4.4.2)
        else
            $\text{box}_t \leftarrow \emptyset$
            $\text{state}_t \leftarrow$ Full occlusion
        end
    else
        $\text{box}_t \leftarrow$ Best detection in frame $t$
        Compute track labels (§4.4)
        $\text{state}_t \leftarrow$ Estimate object state in frame $t$ (§4.4.1)
        switch $\mathit{state}_t$ do
            case Full occlusion
                $\text{box}_t \leftarrow \emptyset$
                No detector update
            end
            case Partial occlusion
                No detector update
            end
            case Change in viewpoint
                $S_t^1 \leftarrow$ Estimate the transformation
                $\text{box}_t^S \leftarrow \text{Transform}(\text{box}_t, S_t^1)$ (§4.4.2)
                $\text{box}_t \leftarrow \text{box}_t^S$
                Learn new detector model (§4.4.2)
            end
            case No change
                Update detector model (§4.4.2)
            end
        endsw
    end
end
```

the image is greater than a pre-determined threshold, we consider it as a full recovery from the occlusion state.

4.5 Implementation Details

Detector. We chose a linear SVM and HOG features to learn the object detector in this work, following a recent approach [Supancic III and Ramanan, 2013] which showed its efficacy on the tracking problem. The regularization parameter in SVM is fixed to 0.1 for all our experiments. The SVM objective function is minimized with LIBLINEAR [Fan et al., 2008]. The initial detector is learned with one positive sample in the first frame and many negative examples harvested from bounding boxes (sampled from the entire image) that do not overlap with the true positive by more than 10%. We also perform 5 iterations of hard negative mining, similar to [Supancic III and Ramanan, 2013]. The learned detector is run at its original scale in the motion-predicted search region. Recall that we handle severe changes in object state (change in scale, rotation) by building a set of detectors (Section 4.3). For all the experiments, we fixed the maximum size of this set to 4, and replaced the worst performing detector (i.e., the detector with the lowest score when evaluated on the new exemplar), whenever necessary. We found this approach to work better in practice, compared to one where a single multi-scale detector is used. To update the detector efficiently with new samples, we use the standard warm-start strategy [Fan et al., 2008, Supancic III and Ramanan, 2013].

State prediction. The parameter λ in the energy function (4.1), which controls the strength of the unary and pairwise terms is set to 1 in all our experiments. Pairwise terms are added between pairs of tracks that are less than a distance of 5 pixels in a frame. We minimize (4.1) with the graph cut algorithm [Hammer, 1965, Kolmogorov and Zabih, 2004, Boykov and Kolmogorov, 2004]. The thresholds for determining a partial or full occlusion are empirically fixed to 40% and 80% respectively in all our experiments.

4.6 Experimental Analysis

4.6.1 Datasets

To compare with the most relevant tracking-by-detection approaches, we use the test videos and ground truth annotations from the OTB dataset [Wu et al., 2013] and TLD dataset [Kalal et al., 2012]. We show a sample set of frames from these videos in Figures 4.5, 4.6 and 4.8. In particular, we evaluate on all the videos in the OTB dataset, many of which contain motion blur, fast motion, rotation, background clutter, and the following sequences from the TLD dataset: Pedestrian2, Pedestrian3, Carchase, which contain challenging scenarios with pose, scale and illumination changes, full or partial occlusion. We note that the sequences in the OTB dataset, do not annotate occlusion states. For example, frames in the Coke sequence where the Coke can is completely occluded by a leaf are still annotated with a bounding box. As a result, our method is at a disadvantage in cases where we estimate an occlusion and do not output a bounding box until the object is re-detected with a sufficiently good detection score.

4.6.2 Evaluation Measures

Some of the previous works in tracking have used mean displacement error in pixels to evaluate the accuracy quantitatively. As argued in [Supancic III and Ramanan, 2013], this measure is not scale-invariant and is not precise for cases when a method loses track of the object. We follow [Kalal et al., 2012, Supancic III and Ramanan, 2013] and treat an estimated object location as correct if it sufficiently overlaps with the ground truth annotation. We then compute precision, recall and the F_1 score. In the results shown in Table 4.1, we use 50% as the overlap threshold.

4.6.3 Experimental Results

In this section we compare our approach with the top-performing methods, namely TLD (2012) [Kalal et al., 2012], SPLTT (2013) [Supancic III and Ramanan, 2013], and the best tracker evaluated on OTB dataset [Wu et al., 2013] — Struck (2013) [Hare et al., 2011]. We used the original implementation provided by the respective authors. For TLD, we set the size of the initial object bounding box as 15, since it did not run with the default value of 24 for some of the sequences.

No.	Sequence	Struck	TLD	SPLTT	Our approach	
					plain	*occ.+vpoint*
1	Football1	0.378	0.351	0.554	1.000	1.000
2	Trellis	0.821	0.455	0.701	0.838	0.919
3	Walking	0.585	0.379	0.541	0.476	0.922
4	MotorRolling	0.146	0.110	0.128	0.134	0.512
5	MountainBike	0.908	0.355	0.908	1.000	1.000
6	Basketball	0.072	0.025	0.396	0.425	0.554
7	Car4	0.404	0.003	0.314	0.401	0.398
8	Suv	0.587	0.913	0.904	0.531	0.907
9	Woman	0.936	0.829	0.891	0.935	0.920
10	Coke	0.948	0.694	0.804	0.801	0.880
11	Freeman4	0.177	0.134	0.145	0.095	0.004
12	Soccer	0.166	0.094	0.133	0.126	0.143
13	Pedestrian2	0.175	0.500	0.950	0.107	0.979
14	Pedestrian3	0.353	0.886	0.989	0.424	1.000
15	Carchase	0.036	0.340	0.290	0.098	0.312

Table 4.1 – Comparison of our approach with the state-of-the-art methods using F_1 measure (higher is better). "*Plain*" and "*occ.+vpoint*" refer to our variants without and with object state estimation respectively.

OTB Results. When evaluated on all the 50 sequences from OTB dataset [Wu et al., 2013], our approach results in 0.657 mean F_1 score (with 50% overlap threshold), whereas Struck [Hare et al., 2011], TLD [Kalal et al., 2012] and SPLTT [Supancic III and Ramanan, 2013] achieve 0.565, 0.513 and 0.661 respectively. We illustrate a selection of these sequences in Table 4.1 (row 1-12), Figures 4.5 and 4.6.

Our method shows a significant improvement on several sequences (rows 1-6 in Table 4.1). For the Football1 sequence (row 1, Table 4.1), our F_1 score is 1.000 compared to 0.554 (SPLTT). In Figure 4.5(1), we see that Struck (columns 2, 3: red box) tends to drift because the model is not selectively updated. SPLTT also performs poorly (column 2: yellow box, column 3: loses track) as it only relies on frame-to-frame optical flow between candidate detections computed in each frame. If either the optical

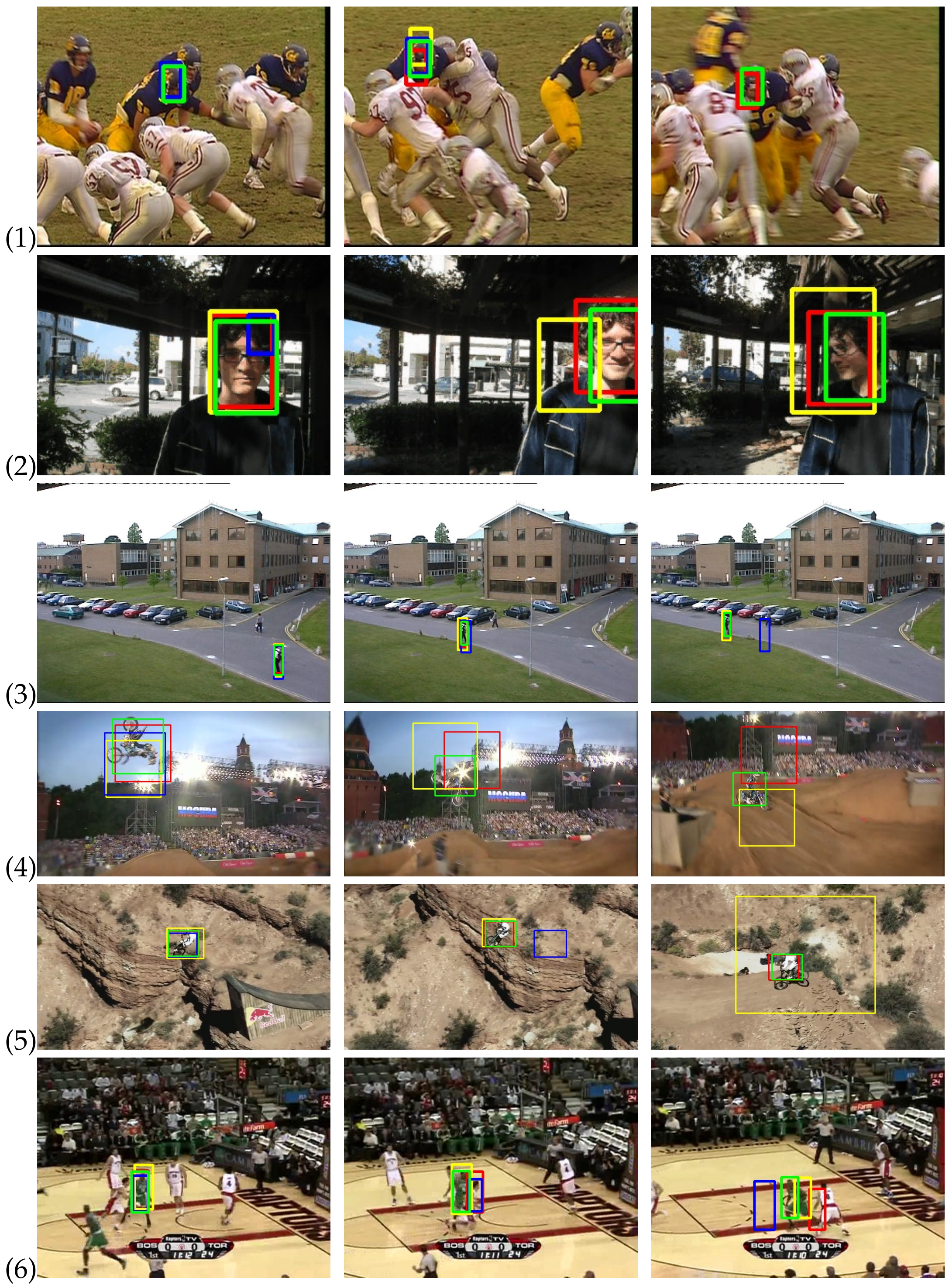


Figure 4.5 – Qualitative results on (1) Football1, (2) Trellis, (3) Walking, (4) MotorRolling, (5) MountainBike, and (6) Basketball sequences from the OTB dataset [Wu et al., 2013]. Green: Our result, Red: Struck, Blue: TLD, Yellow: SPLTT. See text for details.

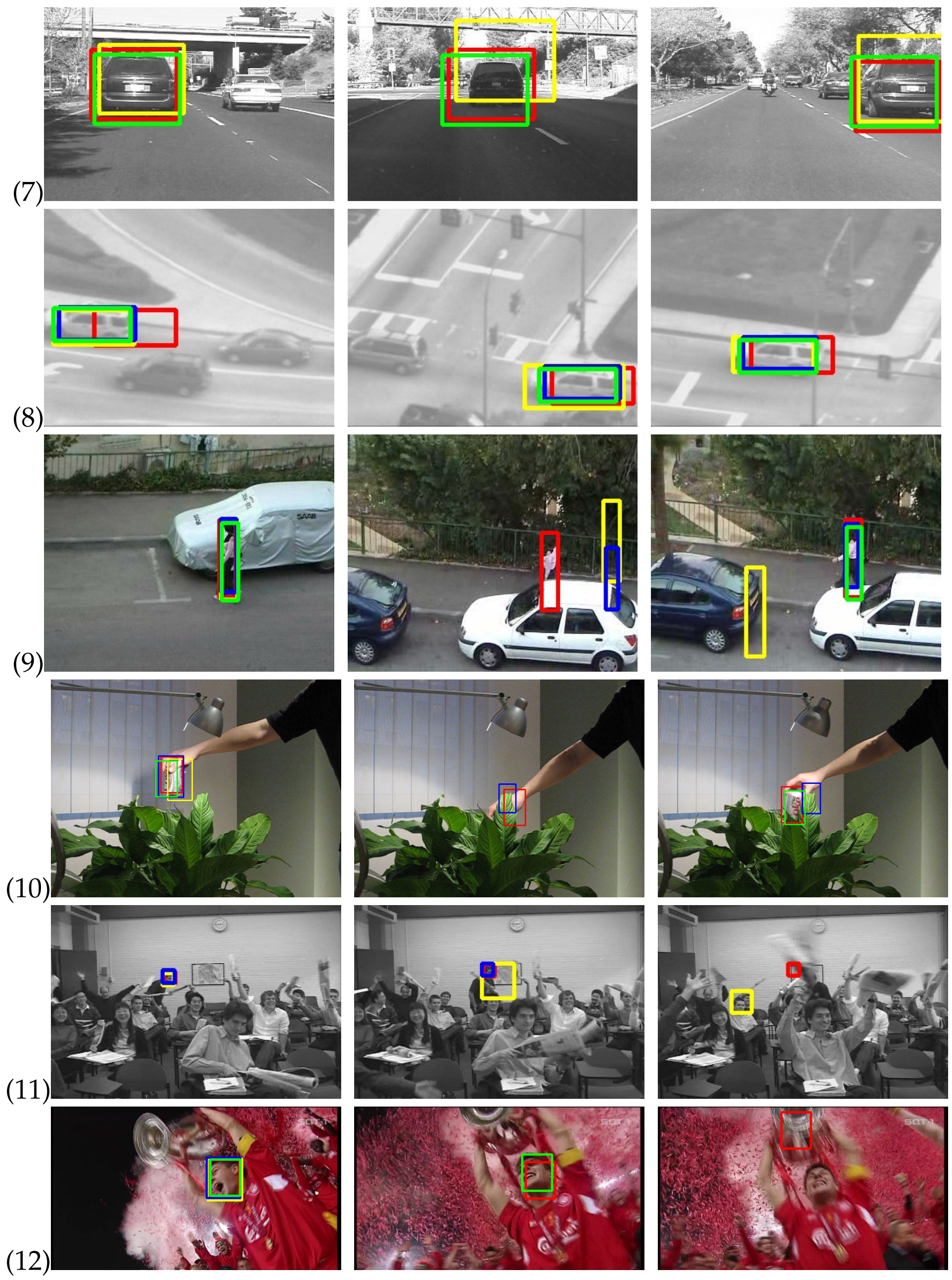


Figure 4.6 – Qualitative results on (7) Car4, (8) Suv, (9) Woman, (10) Coke, (11) Freeman4, and (12) Soccer sequences from the OTB dataset [Wu et al., 2013]. Green: Our result, Red: Struck, Blue: TLD, Yellow: SPLTT. See text for details.

flow or the detection is weak, SPLTT loses track. TLD (column 3: loses track) also uses frame-to-frame optical flow tracking and is prone to drift. In contrast, our method uses long-term tracks and updates the model selectively, which results in better performance. For the Trellis sequence, our method shows nearly 10% improvement over Struck (0.919 vs 0.821, see row 2, Table 4.1). Sample frames are shown in Figure 4.5(2). Here, TLD is confused by the illumination changes, drifts (blue box in column 1) onto a part of the object (the face), and eventually loses track (columns 2 and 3: no blue box). This is potentially due to the weaker object model. The partial occlusions (column 2) and change in viewpoint (column 3) lead to incorrect model updates, and thus poorer results for SPLTT (yellow box) and Struck (red box). Our method (green box) estimates the state of the object (occlusion or change in viewpoint) and performs a correct update step.

For the Walking sequence, we achieve an F_1 score of 0.922 compared to 0.585 from Struck (row 3, Table 4.1). From Figure 4.5(3) we observe that our tracker (green box) adapts to changes in object size (with the help of long-term tracks) better than SPLTT (yellow box) and Struck (red box). TLD (blue box) tracks the object initially, but drifts at about half-way through the sequence (frame 202). For the Basketball sequence (row 6, Table 4.1), our $F1$ score is 0.554 compared to 0.396 (SPLTT). As seen in the sample frames in Figure 4.5(6), the other methods track the object, the basketball player in green, initially (column 1). However, they drift onto another player who runs in front of the object (column 2: red (Struck) and blue (TLD) boxes) or nearby regions (column 3: yellow (SPLTT) and blue (TLD) boxes). Struck updates the model without considering the state of the object, e.g., occlusion, and is thus more prone to drift. Our detector model is updated temporally only when the object is not occluded. It is also stronger than the model used in TLD (a set of image patches). Hence, we continue to track the object accurately. SPLTT recovers from the partial drift (column 3, Figure 4.5(6)), but is less accurate overall.

The performance of our method is comparable on some sequences (rows 7-12 in Table 4.1). For example, an F_1 score of 0.398 compared to 0.404 (Struck) for the Car4 sequence (see row 7, Table 4.1). For the Suv sequence, shown in Figure 4.6(8), our result (0.907) is comparable to SPLTT (0.904), TLD (0.913), and is better than Struck (0.587). In Figure 4.6(9) we show sample frames from the Woman sequence, where our method identifsies that the object is occluded (column 2). Due to the lack of occlusion labeling in the ground truth annotation, our method is penalized

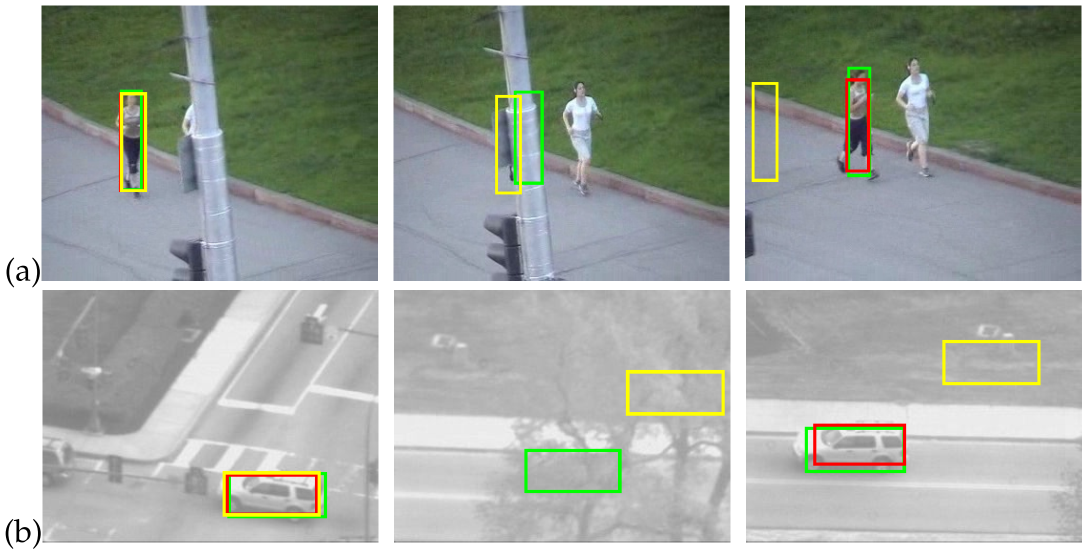


Figure 4.7 – Qualitative results comparing our occlusion reasoning (this chapter) and proposal selection tracker (Chapter 3) on (a) Jogging and (b) Suv sequences from the OTB dataset [Wu et al., 2013]. Yellow: Occlusion reasoning tracker, Red: Proposal selection tracker, Green: Ground truth. See text for details.

for frames where we estimate occlusion, and hence our result is slightly worse (0.920 vs 0.936 (Struck), shown in row 9, Table 4.1). The Coke sequence (row 10, Table 4.1) is another such case, where our method (0.87) performs significantly better than TLD (0.69) and SPLTT (0.80), but is inferior to Struck (0.95).

In a few cases, our method performs worse than the trackers we compare with. For example, on the Freeman4 sequence, our method fails to track the object (0.004 F_1 score). Struck, TLD and SPLTT perform better than this (0.177, 0.134 and 0.145 respectively), but are still significantly inferior to their average performance on the entire benchmark dataset. As shown in Figure 4.6(12), none of the methods show a noteworthy performance, and drift or miss the object often. We observed that the minimum size of our detector was not ideal to find the object in this sequence, which is only 15×16 pixels large. All the trackers also perform poorly on the Soccer sequence — 0.166 is the best performance, which is comparable to our score, 0.143. In Figure 4.6(12) we see that the player's face in this sequence is tracked initially, but due to severe motion blur, fails in the latter frames.

The overall performance of our method presented in this chapter is sim-

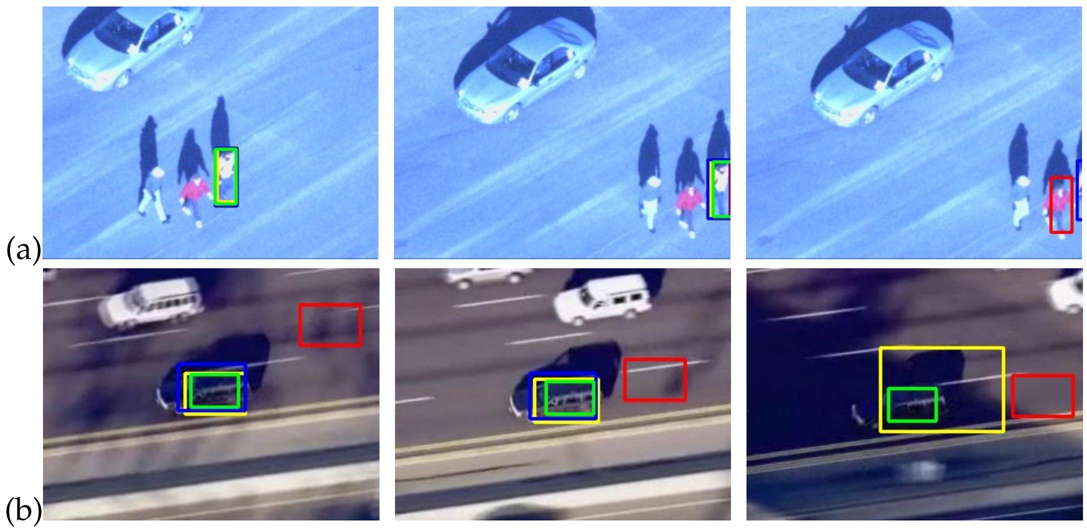


Figure 4.8 – Qualitative results on (a) Pedestrian2, and (b) Carchase sequences from the TLD dataset [Kalal et al., 2012]. Green: Our result, Red: Struck, Blue: TLD, Yellow: SPLTT. See text for details.

ilar to the proposal selection tracker (single-scale) described in Chapter 3. However, the occlusion reasoning method performs significantly better on certain sequences. From the qualitative results shown in Figure 4.7, the method described in this chpater stops tracking when occlusion occurs, and resumes tracking when the target reappears. In contrast, the proposal selection tracker (single-scale) continues to track even when the target is occluded and drifts to the background due to incorrect updates.

TLD Dataset Results. We show tracking results on the most challenging sequences from the TLD dataset (Pedestrian2, Pedestrian3 and Carchase) Table 4.1 (rows 13-15). Results on the Pedestrian2 and Carchase long-term sequences, in Figure 4.8(a) and Figure 4.8(b), show that Struck cannot handle cases where the object re-enters the field of view after occlusion, unlike our method, TLD and SPLTT.

Discussion. Table 4.1 also shows a component-level evaluation of our method, namely "plain" and "occ.+vpoint". The former one refers to our basic approach without predicting the state of the object, while the latter one uses the predicted state. Estimating the state of the object improves the performance in most cases (e.g., Walking sequence in row 3). In some cases we observe a slight decrease in performance over the plain vanilla method (e.g., Woman sequence in row 9) due to lack of occlusion labeling in the ground truth annotation.

Note that long-term tracks are used as an additional information in our work. If there are insufficient point tracks within the bounding box (i.e., less than 10), we do not estimate the state, and continue in a tracking-by-detection mode. For estimating the object state, we observed two cases. (1) Object and camera motion: In this case, tracks from [Sundaram et al., 2010] do not suffer from significant drift as they tend to be relatively short in length. For example, on the Deer sequence (71 frames), the average length of the track is 10.1, and less than 10% of tracks drift. This does not affect our state estimation. (2) Object or camera motion only: Here, tracks can drift, and then result in incorrect occlusion estimates (e.g., Crossing sequence: 120 frames; average track length 77, 50% drift). In the worst case, our tracker predicts full occlusion and misses the object for a few frames, but recovers when the detector is run over the entire image to overcome this occlusion state. In essence, failures in long-term tracks have a limited impact on our system overall. However, a limiting case of our approach is when an object undergoes occlusion, and re-appears in a viewpoint which has not been seen before the occlusion (i.e., no template is learned).

Computation time of our method depends on the image size and the number of tracks in the sequence. For sequences in Table 4.1, it takes 6.7 seconds/frame on average, with our unoptimized Matlab code (which does not include time to precompute optical flow — 3.4 seconds/frame on GPU).

4.7 Summary

This chapter describes a principled way to identify the state of an object to address the model update problem in visual object tracking. In particular, we consider the long-term tracking problem, where the object may become occluded or leave the field of view. To address this, we utilize long-term trajectories in combination with graph-cut based track labeling scheme to determinate the state of the target. We also estimate geometry transformations between consecutive frames based on these trajectories to perform an appropriate update step. We evaluate our approach on multiple datasets and show comparable performance with several state-of-the-art trackers.

Chapter 5

Conclusion

Contents

In this thesis we have focused on the problem of visual object tracking in challenging scenarios, where the object undergoes significant transformations, becomes occluded or leaves the field of view. We proposed two approaches for addressing such cases: (i) We presented a general proposal-selection framework, building on the discriminative tracking-by-detection paradigm for short-term tracking. (ii) We utilized motion cues to identify the state of an object in a principled way in order to address the model update issue for long-term tracking.

The following concludes the thesis with a summary of contributions in Section 5.1, and potential directions for future research in Section 5.2.

5.1 Summary of contributions

In Chapter 3, we investigated challenging scenarios in visual object tracking, where an object undergoes severe geometric transformations. Although a discriminative tracking-by-detection framework has become one of the most successful paradigms for tracking in recent years, it is ill-equipped to handle such challenging conditions. Moreover, it also determines tracking results based only on the best detection score, which is not always an optimal strategy. In order to address these issues, we presented a proposal-selection tracking approach, building on the traditional discriminative tracking-by-detection framework. In the proposal stage, besides candidates from a standard tracking-by-detection framework, we introduced additional candidates that capture geometric transformations undergone by the object to form an enlarged candidate pool. In the selection stage, we then determined the best proposal from this set using multiple cues — detection score, and edgebox scores computed with edge responses and motion boundaries. We utilized a two-phase selection strategy to combine these cues, wherein edgebox scores are used when the detection scores are inconclusive to choose the best proposal (i.e., the detection scores of several candidates are statistically similar). Our extensive experiments showed that the proposal-selection tracking approach achieves top performance on diverse datasets with the same set of parameters.

In Chapter 4, we considered the problem of model update, which is one of the most critical components in long-term tracking. In order words, we studied the problem of determining when to update the model of a target object, as it may become occluded or leave the field of view in the long-term tracking scenario. To address this, we first calculated long-term trajectories from optical flow in the video sequence. Then, we proposed a method to identify the state of the object based on a graph-cut scheme to label the trajectories as object or background. This estimated state of the object allows us to update the model selectively. We evaluated our approach on multiple datasets and showed significant performance gain over state-of-the-art trackers.

In Appendix A, we presented our submission to the visual object tracking challenge organized in 2015. Our tracker, based on our proposal-selection framework, was the winner in the VOT-TIR2015 challenge, and it was ranked sixth in the VOT2015 challenge.

5.2 Future work

In the following, we present potential extensions of our work, which are based on observations from the experimental results, and also inspired by recent progress in computer vision and machine learning. We address extensions of the proposal-selection tracking framework in §5.2.1 and extensions of the occlusion reasoning for long-term tracking in §5.2.2. Finally, in §5.2.3, we discuss potential extensions based on deep features.

5.2.1 Proposal-Selection Tracking Framework

Our proposal-selection tracking approach has achieved state-of-the-art results. However, it can not handle deformable objects. There are two main reasons. First, we only consider candidates undergoing geometry transformations (i.e., similarity transformations) in the proposal stage. In other words, if the objects do not undergo rigid transformations, they will not be included in the candidate pool and thus cannot be selected by the tracker. Second, in the selection stage, we determine the tracking result using detection scores. They are calculated with HOG features and a linear SVM, which tend to perform poorly on deformable objects. To address these issues, the proposal-selection approach can be improved with additional proposals from object segmentation based methods [Son et al., 2015, Li et al., 2013a, Wen et al., 2015, Tsai et al., 2012] and deformable object trackers [Godec et al., 2011, Liu et al., 2015, Nebehay and Pflugfelder, 2015]. Furthermore, as suggested by [Zhu et al., 2015], including candidates from object proposals [Hosang et al., 2016] extracted for the entire frame, can improve tracking robustness, in particular for fast moving objects and low-frame-rate videos. During the selection phase, other cues can be investigated, based on matching [Cho et al., 2014, Revaud et al., 2015a], multiple experts [Zhang et al., 2014a], and multihypothesis trajectory analysis [Lee et al., 2015].

For real-world applications, real-time processing is mandatory for visual object tracking. Currently, the computation time of our proposal-selection pipeline is limited by the optical flow estimation which is used for generating the geometry proposals and motion boundaries, and also the object detector, which is evaluated with small step size. It is worth exploring how faster optical flow methods (e.g., [Tao et al., 2012] and [Revaud et al., 2015b]) or matching based methods (e.g., [Revaud et al., 2015a]) can improve speed without decreasing performance. Furthermore, high-speed trackers, such as [Vojir et al., 2014, Henriques et al., 2015], can replace the HOG and linear SVM based detector in our framework. However, these high-speed trackers may cause loss in performance, as shown in several recent evaluations [Kristan et al., 2014, 2015]. To address this, inspired by [Viola and Jones, 2001], we suggest a cascaded proposal-selection framework with different trackers, taking into account the trade off between performance and computational cost. In particular, in every frame, fast trackers and a simple selection strategy can be first applied to build a pool of pre-selected candidates, i.e., reject unreasonable candidates at a low computational cost. Then, high-performance trackers and better selection strategies can be utilized to determine the final tracking result, from the pre-selected candidate pool.

5.2.2 Occlusion Reasoning for Long-term Tracking

We utilized long-term trajectories to determine the state of a tracked object, which decides when to update the model in long-term tracking. For easy integration, we adopted long-term trajectories that are pre-calculated on the entire video sequence [Ochs et al., 2014]. As a result, our occlusion reasoning tracker cannot run in an online manner, as it relies on trajectories calculated with future frames. We can adapt our framework to the online scenario with trajectories calculated using all the frames until the current frame. Given the problem setting for object tracking, we can use the ground truth annotation in the first frame (or the re-detection result when the tracker recovers from an occlusion) for guiding long-term trajectory segmentation, in order to determine the state of the target. This is more reliable than using additional trajectories from future frames.

As in the case of the proposal-selection framework, the computational bottleneck of our occlusion reasoning tracker is the calculation of optical flow for extracting long-term trajectories. According to our experimental results, quasi-dense long-term trajectories supply sufficient information for identifying the state of the object. These trajectories are built with dense optical flow [Brox and Malik, 2011] calculated at every pixel. Instead of this dense optical flow, we can use matching based methods, e.g., efficient GPU implementation of deep matching [Revaud et al., 2015a], to decrease the computational cost. Moreover, matching based methods can also be applied to get correspondences not only between consecutive frames but also over more distant frames. This can further improve the robustness of long-term trajectories under challenging scenarios, such as frames containing motion blur, where traditional optical flow methods perform poorly. One way to address this is to skip the frames containing motion blur, and find long-range correspondences using matching techniques [Revaud et al., 2015a].

Another extension is to combine the proposal-selection tracking approach and occlusion reasoning into one unified framework, to handle more diverse circumstances, e.g., target objects undergoing significant transformations, and becoming occluded or leaving the field of view. In particular, the proposal-selection tracking approach can be used to handle the short-term tracking scenario, while occlusion reasoning can be applied to determine when to update the model, in order to avoid drifting and improve robustness. This unified framework can be further extended by recent works [Hong et al., 2015b, Ma et al., 2015b], which suggest maintaining multiple trackers with different (aggressive or conservative) updating strategies.

5.2.3 Deep Learning for Tracking

In the past year or so, supervised deep learning methods have been applied to visual object tracking successfully, and have achieved good performance on many tracking datasets [Li et al., 2014, Wang et al., 2015a, Nam and Han, 2016]. However, these methods have not dominated model-free tracking, unlike what is observed on other computer vision problems, e.g., object classification [Krizhevsky et al., 2012], object detection [Girshick et al., 2014], and face verification [Taigman et al., 2014]. One of the reasons is the lack of proper training data in the standard setup of model-free tracking, i.e., the tracking target can be any object or any part of an object, annotated only in the first frame of a video sequence. What is even worse is that, "the same kind of objects can be considered as a target in a sequence and as a background object in another" [Nam and Han, 2016]. Therefore, supervised deep learning methods, which focus on learning an appearance representation, are suboptimal for direct use in model-free tracking. Instead, these approaches, e.g., faster R-CNN [Ren et al., 2015], can be: (i) used to compute object proposals to enrich our proposal-selection framework, and (ii) applied to model-specific tracking problems.

As shown in Chapter 4, long-range motion patterns in video sequences can be used to identify the object state and guide model update for model-free object tracking. Inspired by this work and [Dosovitskiy et al., 2015], we propose to adapt supervised deep learning methods, e.g., temporal convolutional networks learned from multi-frame dense optical flow [Simonyan and Zisserman, 2014] and long-term recurrent convolutional networks learned from video sequences [Donahue et al., 2015], for representing different long-range motion patterns. These motion patterns, e.g., occlusion and self-occlusion, are not necessarily dependent on the appearance of objects. For instance, in the case where a person walking is occluded by a tree or by a building, the appearance of the two objects (i.e., tree and building) is very different. However, from a motion point of view, these two events have the same motion pattern — the motion trajectory from a moving object (person) is terminated by a stationary object (tree or building). This approach can exploit additional training data as it is not restricted by the type of object, thus making it applicable to model-free tracking.

Appendix A

Participation in VOT2015 & VOT-TIR2015 challenges

Contents

In this appendix, we present the details of our submission and the evaluation results on both the visual object tracking (VOT) 2015 and the thermal infrared visual object tracking (VOT-TIR) 2015 challenges. The goal of these challenges is to compare short-term model-free single-object trackers, and serve as the de factor state-of-the-art evaluation platform for visual object tracking. In particular, the VOT challenge focuses on natural RGB video sequences with rotated rectangle ground truth boxes, while the VOT-TIR challenge consists of thermal infrared video sequences with axis-aligned ground truth boxes, see examples in Figures 2.21 and 2.22. For more details of the two challenges, we refer the reader to the challenge reports [Kristan et al., 2015, Felsberg et al., 2015]. In the following, we describe our submitted tracker in Section A.1, and present evaluation results in Section A.2.

A.1 Description of the tracker

We submitted a simplified version of our proposal-selection tracker, referred as to sPST. Compared to the full version of our proposal-selection tracker, described in Chapter 3, we excluded geometry proposals and motion boundaries selection in sPST, due to the computational cost of the optical flow method. Similar to the full version of the proposal-selection tracker, sPST proceeds in two stages – proposal followed by selection. In the proposal stage, we generate a set of candidates computed by the tracking-by-detection framework, where we use the frame as is, or rotate it according to the ground truth annotation in the initial frame to handle rotated bounding box annotation. In the selection stage, we determine the best candidate as the tracking result with detection and edgebox scores. We follow the two-phase selection strategy to combine these two cues, as described in Chapter 3. It is worth noting that in order to show the generality of sPST, we set identical parameters for both VOT2015 and VOT-TIR2015 challenges, despite the target video domains of these two challenges being different.

A.1.1 Experimental environment

We implemented sPST with Matlab 2014b and mex files. For evaluation on the challenge datasets, we followed the guidelines, and integrated sPST into the VOT challenge toolkit.[1] We performed all the experiments on a workstation with an Intel Xeon CPU at 2.4GHz and 48G memory, running Fedora 21 64bit operation system.

A.1.2 Implementation details

We adopted all the parameters of sPST from Chapter 3, which are fixed or calculated based on the ground truth annotation in the first frame of all the sequences in VOT2015 and VOT-TIR2015 challenges.

Object template and HOG feature. The object template and HOG feature parameters are set according to the area of the ground truth bounding box in the initial frame. If the area of the bounding box is larger than 10000 pixels, the object template resize scale is set to 0.5. If the area of the bounding box is between 400 pixels and 10000 pixels, the object template

1. Available at `https://github.com/votchallenge/vot-toolkit`.

resize scale is set to 0.8. If the area of the bounding box is smaller than 400 pixels, the object template resize scale is set to 1.0.

After resizing, if the area of object template is larger than 4000 pixels, the cell size of HOG feature is set to 8. If the area of object template is between 1000 pixels and 4000 pixels, the cell size is set to 6. For an area less than 1000 pixels, the cell size is set to 4.

Detector. The initial detector is trained in the first frame with one positive sample and several negative examples that have less than 50% overlap with the ground truth annotation. In order to make our experimental results repeatable, we fixed the training sample order randomly to learn the SVM. In every frame that follows, the detector is evaluated at seven scales: $\{0.980, 0.990, 0.995, 1.000, 1.005, 1.010, 1.020\}$ with dense-scanning at a step size of 2 pixels. The detector is updated with the tracking result every frame, except when the result in a frame is very similar to that in the previous frame (i.e., the normalized cross-correlation score between current and previous frame results is larger than 0.95).

Candidate proposals. The top 5 detection results are added to the candidate pool. Moreover, if the ground truth bounding box in the initial frame is rotated by more than 15 degrees (clockwise or anti-clockwise), another top 5 detection results on the rotated image are added to enrich the pool.

Candidate selection. We adopted the two-phase selection strategy, discussed in Chapter 3, in the selection stage. First, we check the normalized detection confidence score of all proposals. When the detection scores of some or all the proposals are statistically similar (i.e., the differences between these detection scores and the maximum detection score are less than 1% of the maximum score), we collect all these similar proposals for the following selection step. Otherwise, we choose the proposal with the maximum detection score as tracking result. Second, we check the edgebox scores [Zitnick and Dollár, 2014] of remaining proposals. If all the edgebox scores are less than 0.075 or are not comparable to the mean of the last five edgebox scores of the tracking predictions, which implies a low quality score, we select the candidate with the highest detection score. Otherwise, we choose the proposal with the highest edgebox score as tracking result.

Handling small bounding box. If the initial ground truth box contains less than 300 pixels, we still train the detector as usual. But before evaluating the detector in the new frame, we check the pixel difference between current and previous frames. If more than 40 percent of pixels in the search region are changed, we apply the ordinary proposal-selection scheme to determinate tracking result. Otherwise, we set the region, which has the same size as the previous tracking box and contains the largest percent of changed pixels, as tracking result.

A.2 Evaluation results

According to [Kristan et al., 2015], sPST was ranked sixth among 62 trackers in the VOT2015 challenge. For the VOT-TIR2015 challenge [Felsberg et al., 2015], sPST was ranked second among 24 trackers and received the "winning tracker" title. The tracker ranked first, with a slightly better performance that sPST, was submitted by the organizers themselves, and was disqualified according to the competition rules.

A.2.1 The performance of sPST on VOT2015

All the raw results of each sequence in the VOT2015 dataset are generated by VOT challenge toolkit, and are shown in Table A.1. According to these results from the toolkit, the average accuracy of sPST is 0.54, the average number of failures is 1.42, and it runs at 5.80 fps on average.

	Overlap	**Failures**	**Speed**
bag	0.45	1.00	4.14
ball1	0.85	0.00	2.24
ball2	0.56	0.00	1.93
basketball	0.63	0.00	7.71
birds1	0.42	2.00	3.04
birds2	0.56	0.00	5.54
blanket	0.66	0.00	13.23
bmx	0.35	0.00	2.49
bolt1	0.70	1.00	7.10
bolt2	0.68	0.00	9.93

book	0.36	5.00	3.45
butterfly	0.25	0.00	4.17
car1	0.67	2.00	8.15
car2	0.83	0.00	15.74
crossing	0.70	0.00	3.15
dinosaur	0.48	2.00	6.83
fernando	0.43	1.00	5.25
fish1	0.42	4.00	6.90
fish2	0.38	4.00	4.33
fish3	0.58	0.00	7.79
fish4	0.40	1.00	10.86
girl	0.68	1.00	7.36
glove	0.62	3.00	2.47
godfather	0.37	1.00	7.92
graduate	0.55	3.00	9.93
gymnastics1	0.49	4.00	8.83
gymnastics2	0.57	3.00	2.14
gymnastics3	0.28	3.00	1.31
gymnastics4	0.42	1.00	2.91
hand	0.52	5.00	6.76
handball1	0.62	3.00	9.60
handball2	0.52	3.00	4.24
helicopter	0.41	0.00	5.97
iceskater1	0.46	2.00	6.68
iceskater2	0.57	2.00	7.01
leaves	0.24	5.00	1.44
marching	0.71	0.00	3.58
matrix	0.56	2.00	4.58
motocross1	0.56	1.00	4.81
motocross2	0.34	2.00	0.98
nature	0.33	5.00	5.91

octopus	0.57	0.00	4.96
pedestrian1	0.68	1.00	7.67
pedestrian2	0.45	0.00	9.49
rabbit	0.22	6.00	3.24
racing	0.58	0.00	5.28
road	0.63	1.00	3.97
shaking	0.78	0.00	7.42
sheep	0.58	0.00	9.08
singer1	0.70	0.00	6.62
singer2	0.76	1.00	6.27
singer3	0.24	0.00	4.38
soccer1	0.39	2.00	5.81
soccer2	0.63	1.00	2.39
soldier	0.50	1.00	1.42
sphere	0.69	0.00	6.78
tiger	0.78	0.00	5.72
traffic	0.87	0.00	2.64
tunnel	0.71	0.00	9.49
wiper	0.74	0.00	6.85
mean	0.54	1.42	5.80

Table A.1 – The performance of sPST on the VOT2015 challenge dataset.

A.2.2 The performance of sPST on VOT-TIR2015

All the raw results of each sequence in the VOT-TIR2015 dataset, which are generated by the VOT challenge toolkit, are shown in Table A.2. According to these results from the toolkit, the average accuracy of sPST is 0.70, the average number of failures is 0.35 and it runs at 11.07 fps on average.

	Overlap	**Failures**	**Speed**
birds	0.74	0.00	6.64
car	0.55	0.00	13.02

crossing	0.85	0.00	8.80
crouching	0.67	0.00	10.05
crowd	0.78	0.00	3.98
depthwise_crossing	0.72	0.00	9.58
garden	0.65	3.00	14.69
hiding	0.66	0.00	14.57
horse	0.74	0.00	12.70
jacket	0.82	0.00	11.73
mixed_distractors	0.74	0.00	7.16
quadrocopter	0.52	1.00	5.88
quadrocopter2	0.54	0.00	20.38
rhino_behind_tree	0.71	0.00	22.83
running_rhino	0.54	1.00	22.95
saturated	0.79	0.00	6.72
selma	0.74	0.00	7.08
soccer	0.59	0.00	4.44
street	0.75	1.00	6.65
trees	0.81	1.00	11.49
mean	0.70	0.35	11.07

Table A.2 – The results of the VOT-TIR2015 challenge for our sPST tracker.

As shown in Figure 2.22, thermal infrared (TIR) video data shows more clear edge responses than RGB video data. To highlight the usefulness of this cue for tracking, we evaluated our sPST tracker on VOT-TIR2015 without two-phase selection, i.e., the tracking result in each frame was determined only by the detection score. This results in 0.67 for the average accuracy and 0.35 for the average number of failures, both of which are inferior to sPST with two-phase selection (0.70 and 0.35 respectively).

Publications & Award

This thesis has led to several publications summarized below.

International conferences

- Y. Hua, K. Alahari, C. Schmid. Occlusion and motion reasoning for long-term tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

- Y. Hua, K. Alahari, C. Schmid. Online object tracking with proposal selection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

Other publications

- M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, R. Pflugfelder, *et al*. The visual object tracking VOT2015 challenge results. In *ICCV Workshop on Visual Object Tracking Challenge*, 2015.

- M. Felsberg, A. Berg, G. Hager, J. Ahlberg, M. Kristan, J. Matas, A. Leonardis, L. Cehovin, G. Fernandez, T. Vojir, G. Nebehay, R. Pflugfelder, *et al*. The thermal infrared visual object tracking VOT-TIR2015 challenge results. In *ICCV Workshop on Visual Object Tracking Challenge*, 2015.

Award

- Y. Hua, K. Alahari, C. Schmid. The "winning tracker" award at the Visual Object Tracking challenge VOT-TIR2015. In *ICCV Workshop on Visual Object Tracking Challenge*, 2015.

Bibliography

A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, 2006. 17.

B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, 2012. 50.

M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002. 18.

S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004. 12 and 49.

S. Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, 2007. 10, 12, 46, and 72.

B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011. 10, 13, 14, 46, 49, 58, 72, and 74.

V. Badrinarayanan, P. Pérez, F. L. Clerc, and L. Oisel. Probabilistic color and adaptive multi-feature tracking with dynamically switched priority between cues. In *ICCV*, 2007. 50.

Q. Bai, Z. Wu, S. Sclaroff, M. Betke, and C. Monnier. Randomized ensemble tracking. In *ICCV*, 2013. 10.

C. Bailer, A. Pagani, and D. Stricker. A superior tracking approach: Building a strong tracker through fusion. In *ECCV*, 2014. 25.

S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004. 17.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009. 32.

M. J. Black and A. D. Jepson. EigenTracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998. 10 and 49.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998. 13.

V. N. Boddeti, T. Kanade, and B. Kumar. Correlation filters for object alignment. In *CVPR*, 2013. 19.

D. S. Bolme, B. Draper, and J. R. Beveridge. Average of synthetic exact filters. In *CVPR*, 2009a. 19.

D. S. Bolme, Y. M. Lui, B. Draper, and J. R. Beveridge. Simple real-time human detection using a single correlation filter. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2009b. 19.

D. S. Bolme, J. R. Beveridge, B. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010. 6, 18, 19, and 22.

J.-Y. Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. Technical report, Intel corporation, Microprocessor research labs, 2001. 17.

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004. 79 and 83.

Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *ICCV*, 2001. 78.

T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. 78.

T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011. 51, 68, 72, 77, and 96.

K. Cannons. A review of visual tracking. Technical Report CSE-2008-07, Dept. Comput. Sci. Eng., York Univ., Toronto, Canada, 2008. 10.

J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, 2010. 50.

L. Čehovin, M. Kristan, and A. Leonardis. Is my new tracker really better than yours? In *WACV*, 2014. 39.

L. Čehovin, A. Leonardis, and M. Kristan. Visual object tracking performance measures revisited. *arXiv preprint arXiv:1502.05803*, 2015. 39.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. The MIT Press, 2006. 12.

Z. Chen, Z. Hong, and D. Tao. An experimental survey on correlation filter-based tracking. *arXiv preprint arXiv:1509.05520*, 2015. 19.

M. Cho, J. Sun, O. Duchenne, and J. Ponce. Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers. In *CVPR*, 2014. 95.

D. M. Chu and A. W. Smeulders. Thirteen hard cases in visual tracking. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2010. 35.

B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4):271–288, 1998. 1.

R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, and L. Wixson. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, The Robotics Institute, Carnegie Mellon University, USA, 2000. 1.

R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1631–1643, 2005. 12 and 49.

D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, 2000. 5 and 17.

D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5): 564–577, 2003. 12 and 49.

E. V. Cuevas, D. Zaldivar, and R. Rojas. Kalman filter for vision tracking. Technical Report B 05-12, Institute of Computer Science, Free University of Berlin, Berlin, Germany, 2005. 18.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 10, 21, 50, and 75.

M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014a. 6, 21, 22, 34, 46, 47, 48, 49, 57, 58, 59, and 60.

M. Danelljan, F. S. Khan, M. Felsberg, and J. van de Weijer. Adaptive color attributes for real-time visual tracking. In *CVPR*, 2014b. 21 and 22.

M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015. 6, 20, 21, and 22.

K. G. Derpanis. Characterizing image motion. Technical Report CS-2006-06, Dept. Comput. Sci., York Univ., Toronto, Canada, 2006. 5.

T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1):31–71, 1997. 13.

T. B. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *CVPR*, 2011. 22 and 23.

P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 6 and 53.

J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 97.

A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 97.

W. Du and J. Piater. A probabilistic approach to integrating multiple cues in visual tracking. In *ECCV*, 2008. 24 and 50.

S. Duffner and C. Garcia. PixelTrack: A fast adaptive algorithm for tracking non-rigid objects. In *ICCV*, 2013. 16.

A. Elgammal, R. Duraiswami, and L. S. Davis. Probabilistic tracking in joint feature-spatial spaces. In *CVPR*, 2003. 17.

M. Everingham, J. Sivic, and A. Zisserman. Taking the bite out of automated naming of characters in TV video. *Image and Vision Computing*, 27(5):545–559, 2009. 72.

M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 34 and 49.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008. 55, 56, and 83.

L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4): 594–611, 2006. 34.

M. Felsberg, A. Berg, G. Hager, J. Ahlberg, M. Kristan, J. Matas, A. Leonardis, L. Čehovin, G. Fernandez, T. Vojíř, G. Nebehay, R. Pflugfelder, et al. The thermal infrared visual object tracking vot-tir2015 challenge results. In *ICCV Workshop on Visual Object Tracking Challenge*, 2015. 6, 8, 34, 36, 42, 43, 99, and 102.

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. 10, 21, 49, 50, and 75.

K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975. 17.

J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with gaussian processes regression. In *ECCV*, 2014. 10.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 26, 29, and 97.

M. Godec, P. M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. In *ICCV*, 2011. 15, 16, 46, 72, and 95.

H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, 2006. 10, 12, and 14.

H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008. 13, 14, 46, 72, and 74.

H. Grabner, J. Matas, L. Van Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *CVPR*, 2010. 22.

S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987. 31.

P. L. Hammer. Some network flow problems solved with pseudo-boolean programming. *Operations Research*, 13(3):388–399, 1965. 79 and 83.

B. Han, Y. Zhu, D. Comaniciu, and L. Davis. Kernel-based bayesian filtering for object tracking. In *CVPR*, 2005. 18.

B. Han, S.-W. Joo, and L. S. Davis. Probabilistic fusion tracking using mixture kernel-based bayesian filtering. In *ICCV*, 2007. 26.

S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. 10, 14, 15, 24, 46, 48, 49, 58, 60, 61, 71, 72, 73, 74, 75, 84, and 85.

I. Haritaoglu, D. Harwood, and L. S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, 2000. 1.

R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, second edition, 2004. 52, 77, and 80.

J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, 2012. 20 and 22.

J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. 6, 21, 22, 34, 58, 59, and 95.

D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15, 2008. 22.

S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, 2015a. 28 and 29.

Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Multi-Store tracker (MUSTer): a cognitive psychology inspired approach to object tracking. In *CVPR*, 2015b. 21, 33, 34, and 97.

J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):814–830, 2016. 95.

P. V. C. Hough. Method and means for recognizing complex patterns, 1962. US Patent 3,069,654. 52.

Y. Hua, K. Alahari, and C. Schmid. Occlusion and motion reasoning for long-term tracking. In *ECCV*, 2014. 7, 46, 49, 50, 61, and 62.

Y. Hua, K. Alahari, and C. Schmid. Online object tracking with proposal selection. In *ICCV*, 2015. 7.

M. Isard and A. Blake. Condensation – Conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1): 5–28, 1998a. 18 and 25.

M. Isard and A. Blake. ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. In *ECCV*, 1998b. 50.

X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, 2012. 48.

Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *ICPR*, 2010. 17.

Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7): 1409–1422, 2012. 5, 17, 31, 46, 72, 73, 74, 75, 84, 85, and 90.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. 5 and 53.

M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988. 5.

V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004. 79 and 83.

M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Čehovin, G. Nebehay, G. Fernandez, T. Vojíř, et al. The visual object tracking VOT2013 challenge results. In *ICCV Workshop on Visual Object Tracking Challenge*, 2013. 6, 34, 36, and 41.

M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojíř, G. Fernandez, A. Lukežič, A. Dimitriev, et al. The visual object tracking VOT2014 challenge results. In *ECCV Workshop on Visual Object Tracking Challenge*, 2014. 6, 34, 36, 39, 41, 45, 47, 48, 49, 57, and 95.

M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Čehovin, G. Fernandez, T. Vojíř, G. Hager, G. Nebehay, R. Pflugfelder, et al. The visual object tracking VOT2015 challenge results. In *ICCV Workshop on Visual Object Tracking Challenge*, 2015. 6, 8, 34, 36, 40, 41, 48, 95, 99, and 102.

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, preprint*, 2016. 12.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 26 and 97.

M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *NIPS*, 2010. 32.

S. Kumar and M. Hebert. A hierarchical field framework for unified context-based classification. In *ICCV*, 2005. 21.

D.-Y. Lee, J.-Y. Sim, and C.-S. Kim. Multihypothesis trajectory analysis for robust visual tracking. In *CVPR*, 2015. 24 and 95.

I. Leichter, M. Lindenbaum, and E. Rivlin. A general framework for combining visual trackers–the "black boxes" approach. *International Journal of Computer Vision*, 67(3):343–363, 2006. 25.

C. Leistner, H. Grabner, and H. Bischof. Semi-supervised boosting using visual similarity learning. In *CVPR*, 2008. 13.

J. Lewis. Fast normalized cross-correlation. *Vision Interface*, 10(1):120–123, 1995. 17.

J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*, 2011. 76.

A. Li, M. Lin, Y. Wu, M.-H. Yang, and S. Yan. NUS-PRO: A new visual tracking challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):335–349, 2015a. 6, 34, 35, and 36.

F. Li, T. Kim, A. Humayun, D. Tsai, and J. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013a. 95.

H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In *BMVC*, 2014. 6, 10, 26, 28, and 97.

X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel. A survey of appearance models in visual object tracking. *ACM Transactions on Intelligent Systems and Technology*, 4(4):58, 2013b. 2 and 10.

Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV Workshop on Visual Object Tracking Challenge*, 2014. 21, 22, and 58.

Y. Li, J. Zhu, and S. C. Hoi. Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In *CVPR*, 2015b. 21.

T. Liu, G. Wang, and Q. Yang. Real-time part-based visual tracking via adaptive correlation filters. In *CVPR*, 2015. 21 and 95.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 56.

B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981. 5, 17, and 72.

W. Luo, J. Xing, X. Zhang, X. Zhao, and T.-K. Kim. Multiple object tracking: A literature review. *arXiv preprint arXiv:1409.7618*, 2014. 10.

C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015a. 21, 30, and 31.

C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In *CVPR*, 2015b. 21, 33, and 97.

A. Mahalanobis, B. Vijaya Kumar, and D. Casasent. Minimum average correlation energy filters. *Applied Optics*, 26(17):3633–3640, 1987. 19.

A. Mahalanobis, B. Vijaya Kumar, S. Song, S. Sims, and J. Epperson. Unconstrained correlation filters. *Applied Optics*, 33(17):3751–3759, 1994. 19.

T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *ICCV*, 2011. 81.

P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu. SemiBoost: Boosting for semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2000–2014, 2009. 13.

O. Masoud and N. P. Papanikolopoulos. A novel method for tracking and counting pedestrians in real-time using a single camera. *IEEE Transactions on Vehicular Technology*, 50(5):1267–1278, 2001. 1.

I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6): 810–815, 2004. 4, 31, 72, and 77.

X. Mei and H. Ling. Robust visual tracking using ℓ_1 minimization. In *ICCV*, 2009. 10, 11, 46, and 72.

X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai. Minimum error bounded efficient ℓ_1 tracker with occlusion detection. In *CVPR*, 2011. 11 and 49.

D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert. Contextual classification with functional max-margin markov networks. In *CVPR*, 2009. 22.

K. Murphy, A. Torralba, and W. Freeman. Using the forest to see the trees: A graphical model relating features, objects and scenes. *NIPS*, 2003. 22.

H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 6, 27, 29, and 97.

G. Nebehay and R. Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *WACV*, 2014. 49, 57, and 58.

G. Nebehay and R. Pflugfelder. Clustering of static-adaptive correspondences for deformable object tracking. In *CVPR*, 2015. 95.

P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1187–1200, 2014. 72 and 96.

Y. Pang and H. Ling. Finding the best from the second bests-inhibiting subjective bias in evaluation of visual tracking algorithms. In *ICCV*, 2013. 45, 49, and 72.

D. W. Park, J. Kwon, and K. M. Lee. Robust visual tracking using autoregressive hidden Markov model. In *CVPR*, 2012. 50.

S. Paschalakis and M. Bober. Real-time face detection and tracking for mobile videoconferencing. *Real-Time Imaging*, 10(2):81–94, 2004. 1.

V. I. Pavlovic, R. Sharma, and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997. 1.

P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *ECCV*, 2002. 10.

P. Perez, J. Vermaak, and A. Blake. Data fusion for visual tracking with particles. *Proceedings of the IEEE*, 92(3):495–513, 2004. 26.

J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74, 1999. 55 and 79.

H. Possegger, T. Mauthner, and H. Bischof. In defense of color-based model-free tracking. In *CVPR*, 2015. 10.

A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *ICCV*, 2007. 22.

S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 97.

J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Deep convolutional matching. *arXiv preprint arXiv:1506.07656*, 2015a. 95 and 96.

J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015b. 95.

D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1): 125–141, 2008. 10 and 49.

C. Rother, V. Kolmogorov, and A. Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004. 79.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 28 and 34.

V. Salari and I. K. Sethi. Feature point correspondence in the presence of occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):87–91, 1990. 5.

J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST: Parallel robust online simple tracking. In *CVPR*, 2010. 17, 24, 25, and 50.

I. K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56–73, 1987. 5.

J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994. 17.

J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006. 22.

J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013. 35.

G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah. Part-based multiple-person tracking with partial occlusion handling. In *CVPR*, 2012. 10.

T. Sikora. The MPEG-4 video standard verification model. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):19–31, 1997. 1.

K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 97.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 30.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*, 2014. 29.

A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, 2014. 6, 12, 34, 35, 36, 41, 45, and 49.

J. Son, I. Jung, K. Park, and B. Han. Tracking-by-segmentation with online gradient boosting decision tree. In *ICCV*, 2015. 95.

S. Song and J. Xiao. Tracking revisited using RGBD camera: Unified benchmark and baselines. In *ICCV*, 2013. 35, 36, and 45.

Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan. Contextualizing object detection and classification. In *CVPR*, 2011. 22.

M. Spengler and B. Schiele. Towards robust multi-cue integration for visual tracking. *Machine Vision and Applications*, 14(1):50–58, 2003. 26.

B. Stenger, T. Woodley, and R. Cipolla. Learning to track with multiple observers. In *CVPR*, 2009. 26.

N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, 2010. 72, 73, 74, 76, and 91.

J. S. Supancic III and D. Ramanan. Self-paced learning for long-term tracking. In *CVPR*, 2013. 10, 32, 46, 49, 50, 56, 61, 62, 72, 73, 74, 75, 77, 83, 84, and 85.

J.-C. Tai, S.-T. Tseng, C.-P. Lin, and K.-T. Song. Real-time image tracking for automatic traffic monitoring and enforcement applications. *Image and Vision Computing*, 22(6):485–501, 2004. 1.

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. 26 and 97.

F. Tang, S. Brennan, Q. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. In *ICCV*, 2007. 10 and 13.

M. Tao, J. Bai, P. Kohli, and S. Paris. SimpleFlow: A non-iterative, sublinear optical flow algorithm. In *Computer Graphics Forum*, 2012. 95.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. 11.

C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, School of Computer Science, Carnegie Mellon University, USA, 1991. 5 and 17.

A. Torralba. Contextual priming for object detection. *International Journal of Computer Vision*, 53(2):169–191, 2003. 22.

D. Tsai, M. Flagg, A. Nakazawa, and J. M. Rehg. Motion coherent tracking using multi-label MRF optimization. *International Journal of Computer Vision*, 100(2):190–202, 2012. 95.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, volume 6, pages 1453–1484, 2005. 14.

J. Van De Weijer, C. Schmid, J. Verbeek, and D. Larlus. Learning color names for real-world applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523, 2009. 21.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 10 and 95.

T. Vojir, J. Noskova, and J. Matas. Robust scale-adaptive mean-shift for tracking. *Pattern Recognition Letters*, 49:250–258, 2014. 95.

L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015a. 6, 29, 30, and 97.

N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013. 27.

N. Wang and D.-Y. Yeung. Ensemble-based tracking: Aggregating crowd-sourced structured time series data. In *ICML*, 2014. 25.

N. Wang, S. Li, A. Gupta, and D.-Y. Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015b. 28.

X. Wang. Intelligent multi-camera video surveillance: A review. *Pattern recognition letters*, 34(1):3–19, 2013. 10.

N. Wax. Signal-to-noise improvement and the statistics of track populations. *Journal of Applied Physics*, 26(5):586–595, 1955. 4.

P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Learning to detect motion boundaries. In *CVPR*, 2015. 6 and 53.

L. Wen, Z. Cai, Z. Lei, D. Yi, and S. Z. Li. Online spatio-temporal structural context learning for visual tracking. In *ECCV*, 2012. 23.

L. Wen, D. Du, Z. Lei, S. Z. Li, and M.-H. Yang. JOTS: Joint online tracking and segmentation. In *CVPR*, 2015. 95.

J. Whitehill, T.-F. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, 2009. 25.

C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997. 5.

J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009. 11.

Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 6, 12, 34, 36, 37, 41, 45, 48, 49, 57, 61, 62, 72, 73, 75, 76, 81, 84, 85, 86, 87, and 89.

Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. 36.

C. Yang, R. Duraiswami, and L. Davis. Efficient mean-shift tracking via a new similarity measure. In *CVPR*, 2005. 17.

H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823–3831, 2011. 10.

M. Yang, Y. Wu, and G. Hua. Context-aware visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1195–1209, 2009. 22.

A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):1–45, 2006. 1 and 10.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 29.

J. Zhang, S. Ma, and S. Sclaroff. MEEM: Robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014a. 32, 46, and 95.

K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M.-H. Yang. Fast visual tracking via dense spatio-temporal context learning. In *ECCV*, 2014b. 21 and 23.

S. Zhang, H. Yao, X. Sun, and X. Lu. Sparse coding based visual tracking: review and experimental comparison. *Pattern Recognition*, 46(7):1772–1788, 2013. 12.

T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Low-rank sparse learning for robust visual tracking. In *ECCV*, 2012a. 11.

T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via multi-task sparse learning. In *CVPR*, 2012b. 11.

B. Zhong, H. Yao, S. Chen, R. Ji, T.-J. Chin, and H. Wang. Visual tracking via weakly supervised learning from multiple imperfect oracles. *Pattern Recognition*, 47(3):1395–1410, 2014. 25.

W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, 2012. 48 and 61.

X. Zhou, L. Xie, P. Zhang, and Y. Zhang. An ensemble of deep neural networks for object tracking. In *ICIP*, 2014. 27.

G. Zhu, F. Porikli, and H. Li. Tracking randomly moving objects on edge box proposals. *arXiv preprint arXiv:1507.08085*, 2015. 95.

C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 6, 48, 53, 54, and 101.