Georg Nebehay

A Deformable Part Model for One-Shot Object Tracking

DOCTORAL THESIS

submitted to
Graz University of Technology

Supervisor

Prof. Dr. Horst Bischof
Graz University of Technology

Co-Supervisor

Univ. Lect. Dr. Roman Pflugfelder
Austrian Institute of Technology

Examiner

Ass. Prof. Dr. Matej Kristan
University of Ljubljana

Graz, Austria, Sep. 2016

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. The text document uploaded to TUGRAZonline is identical to the presented doctoral thesis.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Ort	Datum	Unterschrift

Abstract

As cameras become ubiquitously available, the need for analyzing video sequences on-the-fly arises. An important class of applications requires algorithms that are able to continuously track an a-priori unknown object of interest as it makes its way through the scene. This problem is difficult, as no training data can be used beforehand to create an object model. In this thesis, this problem is referred to as one-shot object tracking. Extensive literature about the topic of one-shot object tracking is available, still the performance of state-of-the-art one-shot tracking algorithms on realistic sequences leaves much to be desired. In this thesis the viewpoint is taken that the deformation of objects of interest acts as a major obstacle for achieving satisfactory results.

While approaches have been proposed in the literature for dealing with this challenge, they either are too simple to be of use for complex objects or require a considerable amount of training data to work. However, in one-shot object tracking there is by definition only one training example available. More training examples can be collected from the video sequence in an online manner, however this process is error-prone and can lead to the undesired effect of accumulating errors so that the object model is no longer a good representation of the object of interest.

In this thesis, a deformable part model for one-shot object tracking is proposed, aiming at providing a robust model for deformable objects that does not rely on model updates to work. Instead, it operates on the basic assumption that object parts are connected by mediating parts, like an arm might connect a hand to the torso of a person. One advantage of the proposed model is the independence on the actual parts representation. We suggest to leverage the synergies between two very different methods for establishing parts correspondences. These methods consist on the one hand of static correspondences, which are based on training information only. This type of correspondences is robust, but unable to adapt to new object appearances. A complementary method can be found in adaptive correspondences, which are computed from a very recent appearance of the object. Adaptive correspondences lack the robustness of static correspondences, but can provide necessary accuracy.

To assess the usefulness of the proposed model in practice, we conduct a rigorous evaluation on a dataset of 77 sequences. This evaluation includes a comparison to state-of-the art tracking algorithms, the effect of employing different part representations as well as additional experiments that reveil insights about internal workings of the proposed model. We find that the proposed deformable part model gives a significant performance improvement over the state-of-the art.

Kurzfassung

Das Bedürfnis, Videosequenzen automatisiert auszuwerten, nimmt im selben Maße zu wie die Verfügbarkeit von Kameras. Ein wichtiger Anwendungstyp benötigt Algorithmen, die ein zuvor unbekanntes Zielobjekt kontinuierlich verfolgen können. Dieses Problem ist deswegen schwierig, weil keine Trainingsdaten verwendet werden können, um im Vorhinein ein Objektmodell zu erstellen. In dieser Dissertation wird dieses Problem One-Shot-Tracking genannt. Obwohl es unzählige Publikationen zu diesem Thema gibt, lassen die Ergebnisse des Standes der Technik auf realistischen Sequenzen sehr zu wünschen übrig. In dieser Dissertation wird der Standpunkt eingenommen, dass die Deformation von Zielobjekten das größte Hindernis darstellt, um zufriedenstellende Ergebnisse zu erzielen.

Es gibt zwar Ansätze in der Literatur, diese Herausforderung zu meistern, diese sind jedoch zu simpel für komplexe Objekte oder benötigen eine beträchtliche Menge an Trainingsdaten. Es gibt aber per Definitionem in One-Shot-Tracking nur ein einziges Trainingsbeispiel. Zwar können aus der Videosequenz weitere Trainingsdaten extrahiert werden, dieser Prozess ist allerdings fehleranfällig und kann zu dem unerwünschten Effekt der Fehlerakkumulierung führen, so dass das Objektmodell keine gute Repräsentation des Zielobjekts mehr darstellt.

In dieser Dissertation wird ein Modell für One-Shot-Tracking vorgeschlagen, das auf keinerlei Modellaktualisierung beruht. An deren Stelle tritt die Annahme, dass einzelne Objektteile durch vermittelnde Objektteile verbunden sind, so wie ein Arm die Hand und den Torso einer Person verbindet. Ein Vorteil des vorgeschlagenen Modells ist die Unabhängigkeit von der Art, wie die Teile repräsentiert werden. Wir schlagen vor, die Synergien zwischen zwei verschiedenen Methoden zur Herstellung von Teilkorrespondenzen auszuschöpfen. Diese Methoden bestehen zum Einen aus statischen Korrespondenzen, die auf Trainingsdaten beruhen. Dieser Korrespondenztyp ist robust, kann aber keine neue Ansichten des Objekts aufnehmen. Eine komplementäre Methode dazu sind adaptive Korrespondenzen, die anhand einer aktuellen Ansicht des Objekts berechnet werden. Adaptiven Korrespondenzen fehlt die Robustheit von statischen Korrespondenzen, sie können aber eine höhere Genauigkeit liefern.

Um den Nutzen des vorgeschlagenen Modells zu überprüfen, führen wir eine hieb- und stichfeste Evaluierung auf einem Datensatz mit 77 Sequenzen durch. Diese Evaluierung beinhaltet einen Vergleich mit dem Stand der Technik, den Vergleich von unterschiedlichen Teilmodellen sowie zusätzliche Eperimente, die Einsichten in die Interna des vorgeschlagenen Modells enthüllen. Wir stellen fest, dass das vorgeschlagene Modell eine signifikante Verbesserung gegenüber dem Stand der Technik bringt.

Acknowledgment

Obtaining a PhD degree is not for the faint of the heart. It requires personal commitment, endurance and austerity, but even more important is the support of people providing advice, support and comfort. I would like to take this opportunity and express my deepest gratitude to Roman Pflugfelder, who paved the way on which I began my journey by giving me the opportunity to participate in his research ideas and, more importantly, believing in that I could eventually provide a contribution. The constant support I received was more than I could have ever asked from a supervisor, opening my eyes to new directions and helping me understand when I had taken a wrong turn. Having arrived at my destination and looking back at the time we spent together fulfills me with a touch of nostalgia, as I clearly see it as a time not only of professional achievements, but also as a time of personal transformation, in which Roman assumed an important role.

I would also like to thank Horst Bischof for acting as the main supervisor during this work and for sharing valuable gems from his vast knowledge. Markus Kommenda provided me with excellent working conditions and kept me free from distractions, which cannot be taken for granted and cannot be praised highly enough. I would also like to thank Branislav Micusik who spotted TLD at ICCV and set the course for my future work. Dan Shao provided extremely valuable emotional support during difficult times before submissions deadlines, maybe one day there can be even two tigers on a mountain. More thanks for endless discussions, lunch-time getaways and chess go out to Gustavo Fernandez, Andreas Zoufal, Csaba Beleznai, Christoph Weiß, Peter Gemeiner, David Schreiber, Cristina Picus, Michael Rauter, Andreas Opitz, Markus Hofstätter and the rest of SVT, even though some of them are suckers for proprietary software. This is a good moment to thank the open source community for sharing their tools.

The members of the EPiCS project have been a great source of inspiration for me, opening the door to international collaborations and broadening my horizon. Special thanks to Peter Lewis for introducing me to the power of vi, without which everything would certainly have been less exciting. Similarly, participating in the organization of the VOT challenge was great fun for me and I would like to thank Matej Kristan, Luka Čehovin, Jiří Matas, Aleš Leonardis, and Tomáš Vojíř (I hope I got all the diacritics right) for this special experience. Hopefully there will be many more successful challenges.

More special thanks go out to Helmut Karg, who subconsciously convinced me to study computer science. I hope that one day he finishes his BSc. Last but not least I would like to thank my parents for providing me with a comfortable chaise longue for my office and the fire protection department of Techgate for not complaining about it.

Notation

Symbol	Meaning
$\|.\|$	L_2 norm
α	rotation
b	bounding box
c	descriptor
d	dimension
δ	deformation threshold
f	measure
γ	ratio threshold
h	vote
θ	threshold
m	correspondence
μ	center/mean
p	distance
s	scale
t	current iteration
v	displacement vector
x	part
ϕ	valid configuration
ω	consensus
ψ	overlap
C	candidate descriptors
D	dissimilarity
H	transformation matrix
I	image
M	number of correspondences

Symbol	Meaning
N	number of parts
P	reference descriptors
R	rotation matrix
T	sequence length
Z	reference part configuration
$\mathcal{I}$	indicator function
$\mathcal{L}$	part correspondences
$\mathcal{L}^A$	adaptive correspondences
$\mathcal{L}^S$	static correspondences
$\mathcal{L}^\omega$	consensus set
GT	Ground Truth
LK	Lucas-Kanade method
NN	Nearest Neighbor
ALG	Algorithmic output
CMT	Consensus-based Matching and Tracking
GHT	Generalized Hough Transform
OPE	One-Pass Evaluation
ECDF	Empirical Cumulative Distribution Function
SNNDR	Second Nearest Neighbor Distance Ratio
DPMOST	Deformable Part Model for One-Shot Object Tracking
RANSAC	Random Sample Consensus

Contents

Chapter 1

Introduction

Visual object tracking research has gone a long way since its seminal use in military applications (Wax, 1955; Kalman, 1960). Due to the ubiquitous availability of cameras and computing power, more people than ever are striving to employ object tracking methods in more peaceful usage scenarios. While the demand for out-of-the-box tracking solutions remains high, object tracking algorithms have failed to stand up to their promise of delivering accurate, robust and resource-aware results. Object tracking is still considered to be one of the major unsolved problems in computer vision. This introductory chapter discusses the very basics of one-shot object tracking, a sub-area of tracking.

1.1 Definition of One-shot Object Tracking

According to Maggio and Cavallaro (2011), the general goal of a tracking algorithm is to estimate the state of one or more objects of interest in a video sequence composed of images $I_1, \ldots, I_T$. The state is an abstraction of the output of the algorithm, which may consist of points, ellipses, axis aligned bounding boxes, rotated bounding boxes, polygons or silhouettes (Yilmaz et al., 2006)[1]. Different state models represent objects at different levels of granularity. For instance, the silhouette of an object is a more fine-grained representation than an ellipse the object was fitted into. From an application's point of view, the most accurate state representation is a pixel-wise segmentation of the object of interest. In practice however, virtually every single published tracker employs bounding boxes as the representation of state for reasons of simplicity. A bounding box b is a rectangle defined by its top left corner and its width and height. While this representation is simple and easy to use, it is clear that most objects encountered in real-world scenarios are essentially non-rectangular. This leads to the undesired situation that either certain object parts protrude beyond the bounding box or background objects extend into the bounding box. In spite of these effects, we still adopt the ubiquitous use of bounding boxes as general tracking output to remain comparable to the state of the art. Still, most of the content in this work is also applicable to other state representations. We will now narrow down the general definition of Maggio and Cavallaro to a more specific branch of object tracking that we focus on in this work.

[1] On an even more abstract level, the state may consist of general properties of the object, e.g. visibility, size, etc.

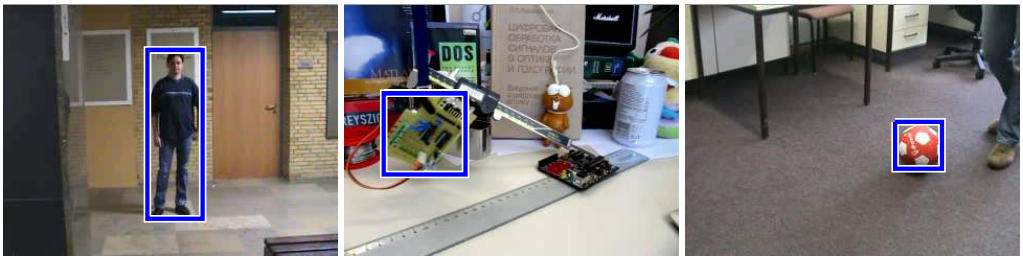


Figure 1.1: In one-shot object tracking, no prior information about the object of interest is available except for the initialization region in the first frame of the video sequence. Images are from Klein et al. (2010) and Santner et al. (2010).

Many authors have studied how to track classes of objects, for instance cars (Koller et al., 1994) or humans (Gavrila, 1999; Shotton et al., 2011). In these scenarios, it is possible to incorporate prior knowledge about object of interests into the tracking algorithm. For instance, one cue that could be used in car tracking is the license plate, which provides a means of uniquely identifying an object. However, when the object class is unknown beforehand, class-specific cues are no longer available and more general assumptions have to be made that ideally apply to all object classes equally well. For example, Lucas and Kanade (1981) employ the asssumption that the object of interest changes its position from frame to frame only slightly. Another popular assumption made by Comaniciu et al. (2000) is that objects are uniquely identifiable by their color composition. Clearly, it is much harder to find assumptions that work for each and every object compared to assumptions that need to work for an object class only. At the same time, the tracking of unknown objects is of particular interest, as a solution to it would have a great impact to both the scientific community and practitioners.

For a long time authors have not bothered to employ a distinct term to distinguish between different variants of the object tracking problem and referred to them collectively as *visual* object tracking. More recently, authors have adopted the term *model-free* object tracking[2] to distinguish tracking of unknown objects from the case when a model can be obtained beforehand (model-based), for instance by making use of training data. This term however is the source of perennial confusion, as every tracking algorithm requires some kind of object model. To avoid this unpleasant situation, we introduce in this work the term *one-shot* object tracking, borrowed from one-shot object learning (Fei-Fei et al., 2006). The term one-shot object tracking is meant to convey the idea that exactly one training example of the object is available in the form of the initial bounding box b_1 in the first frame I_1 of the video sequence. Examples for this are shown in Figure 1.1, where a person, a circuit board and a ball are selected as the object of interest. The variety in this small selection of objects already hints at the difficulty of finding a common object model that is suitable for all of these objects.

In multi-target tracking the relationship between objects of interests can be modeled,

[2] Wang and Nevatia (2013) refer to this as *category-free* tracking.

for instance as proposed by L. Zhang and Maaten (2013). Multi-target tracking leads to a potential improvement of tracking results and we do note that the topic of multi-target tracking is a fruitful one. However, we restrict the discussion to *single-target* tracking to keep the scope of this work within reasonable bounds. In single-target tracking exactly one object of interest must be designated for tracking. This does include the cases when multiple other, possibly distracting objects are present in the video sequence. It is interesting to note that in principle every single-target tracker can be converted to a primitive multi-target tracker by instantiating the tracking algorithm independently for each object of interest. In this case however, all interaction between object of interests is ignored.

The third aspect where we apply a restriction to our discussion refers to the question *when* the algorithmic output is performed. An offline tracking algorithm is allowed to first process all images $I_1, \ldots, I_T$ from a video sequence and only then output of the object location for each individual frame. This allows for instance for the use of optimization algorithms (Andriyenko et al., 2012) where certain constraints such as trajectory smoothness and visual similarity are enforced simultaneously. While there are certainly many fruitful applications for offline tracking algorithms, we are interested in the more general class of tracking algorithms that output the tracking result immediately after being presented with an image. In this work, we refer to this class of trackers as *online*[3] trackers. To appreciate the difference between offline and online tracking algorithms, imagine a scenario where an unmanned aerial vehicle (UAV) should follow an object of interest by means of visual information. Clearly, the UAV needs to react immediately to an updated position of the object of interest, making the application of an online tracking algorithms necessary. One can generalize from the extreme cases of offline and online tracking to algorithms that delay the output for a number of frames. In an offline tracking algorithm the output is delayed by $T - t$ frames, where T is the number of frames in a video sequence and t is the number of the current frame. In online tracking algorithms the output is delayed by exactly 1 frame. While this class of semi-online trackers is worth investigating, we do not pay any attention to them in this work.

Based on the above discussion, we restrict the broad object tracking definition of Maggio and Cavallaro to one-shot single-target online tracking with bounding boxes:

> *Given an image sequence $I_1, \ldots, I_T$ and an initial bounding box b_1 in frame I_1 containing an object of interest, the aim is to find in each frame I_t the bounding box b_t that maximizes overlap with the object of interest while minimizing overlap with all other image areas.*

In the remainder of this work, we will use the term one-shot (object) tracking as a shorthand for this definition. The algorithmic loop that all one-shot tracking algorithms follow in principle is shown in Algorithm 1.

[3] This property has also been referred to as causal tracking (Kristan et al., 2016).

Algorithm 1 One-shot Tracking Loop

Input: Images $I_1, \ldots, I_T$, bounding box b_1

```
initialize(I_1, b_1)
for t = 2, ..., T do
    b_t ← track(I_t)
    print(b_t)
end for
```

One question that creators of one-shot tracking algorithms like to push aside (Cannons, 2008) is how to obtain the initial bounding box b_1. Typically it is assumed that the initialization is performed by some external mechanism, for instance by a manual selection or by a different algorithmic component, such as an object detector. We note that it is desirable to investigate more principled ways of initializing one-shot tracking algorithms and exclude this topic from our discussion as well.

1.2 Applications

An immediate question that arises from the discussion so far is what one-shot tracking algorithms are actually good for. Clearly, a general solution for the problem of one-shot tracking from today's viewpoint is unrealistic. The existence of such a solution would imply that one-shot object tracking algorithms could work in each and every setting, using whatever cheap image sensor is available, even in cases when the object of interest becomes extremely small. Still, there are a number of cases where one-shot object tracking algorithms work *sufficiently* well. These cases have in common that the object of interest is a more or less unique object in the scene and that no extreme viewpoint variations occur, such as abruptly changing from a frontal view of the object to a top view of the object. In this section, we present a selection of these cases mixed with interesting inquiries for applications that the author of this work has received over a timespan of multiple years. We will present these cases in a subjectively sorted order by decreasing realizability. Examples illustrating these use-cases are shown in Figure 1.2.

One desire that arises frequently whenever video data is recorded is to keep an object of interest in the center of the video. This task, usually carried out by human camera operators, is well suited for being tackled by one-shot tracking algorithms. For instance, by equipping a camera with one-shot tracking software and appropriate motors, a speaker wandering around on a stage might be kept centered by adjusting the camera orientation based on information from a tracking algorithm. Such a scenario is well-suited for one-shot tracking algorithms, as there is a clearly defined, unique object of interest and the scene remains roughly constant. Similarly, when people engage in conference calls, they are mandated to remain in front of the camera so that they can be seen by participants. Tracking their faces allows for adjusting the camera accordingly. Also note how important the online property of one-shot tracking algorithms is in this case.

Automatic camera adjustment | UAV navigation | Automated surveillance

Figure 1.2: Example applications of one-shot object tracking. Images are from Graether and Mueller (2012) and Ferryman and Ellis (2010).

Huge efforts are currently undertaken to make self-driving cars feasible. A self-driving car is expected to handle dangerous situations on the road at least on the same level of awareness as a human driver. These situations include for instance unexpected crossings of pedestrians, other cars that are behaving abnormally or animals on the road. In all of these cases it is necessary to track the objects that might cause potential harm. Furthermore, it is especially important to be able to track objects that have not been seen during an offline training phase, as the number of objects that can appear on a road is extremely high and any tracking failure might have fatal consequences.

A recurrent task in movie production is the addition of special effects into a recorded scene. One-shot object tracking can alleviate the tedious procedure of manually annotating objects of interests that need to be enhanced. Typically, these effects are meant to be applied between two cuts, keeping the difficulties arising from different camera viewpoints at bay. These techniques might also be implemented in popular video hosting platforms, allowing for performing these actions in a browser. An advantage in these scenarios is that the user can interact with the tracking software and is able to correct errors manually in a semi-automated fashion. It has to be noted however that this task might require a more fine-grained estimation of the object position than a bounding box, such as a segmentation.

While researchers have worked for a long time on making robots autonomous, this topic has gained increasing attention recently due to the advent of affordable unmanned aerial vehicles (UAVs). These UAVs can be equipped with cameras, allowing for exploiting the video data in a multitude of ways. One way of doing so is to perform video-based flight stabilization. In this application, a non-moving target on the ground is defined that the UAV can use as a reference for maintaining a stable position. In more sophisticated scenarios, the target can be allowed to move, making it possible for UAVs to follow objects of interests and to record them or interact with them as necessary. The recording of moving objects of interest is especially interesting in the context of action videography, where athletes as well as hobbyists keep looking for new ways of recording themselves from unorthodox camera angles.

An ever re-occurring use case of object tracking lies in automated surveillance. Studies

have shown that supervisors of surveillance systems are unable to work effectively for an extended timespan such as hours (Smith, 2002). The outlook of improved surveillance performance and reduced costs has led to a considerable interest in automating this task. However, the biggest obstacle for making one-shot tracking algorithms work effectively in this scenario is arguably the abrupt transitions and appearance changes when the object of interest moves from one camera to another. For example, despite excellent camera coverage, the terrorists responsible for the Boston Marathon bombing in 2013 were identified exclusively by inspecting the video footage manually. While motivated differently, researchers in biology have begun to use tracking techniques to analyze the behaviour of animals such as mammals or birds in their natural habitats. This scenario suffers however from similar difficulties.

This small selection of use cases gives an insight into the vast possibilities that one-shot tracking can offer. Clearly, there are many more interesting areas of applications, such as augmented reality, games and medical imaging, but we stop the discussion at this point and turn to the question what the common challenges to all of these problems are.

1.3 Challenges

The goal of designing a one-shot object tracking algorithm is to come up with an object model that captures the essence of the object of interest and allows for localizing it. In the next chapter, we will give an overview about different object models that have been proposed in the scientific literature. Before that, we explore in this section common challenges encountered when devising models for one-shot object tracking. As a start, we present an introductory quote to this topic (Box, 1979):

> *All models are wrong, but some are useful.*

While this statement emerged from the discussion of statistical models, it can equally well be applied to the design of object models for one-shot tracking, where the guiding principle should be to come up with a model that works instead of a model that is correct. This quote can also be interpreted as a hint to find the right balance in the level of *generality* of the object model. We will explore now certain aspects of what can happen if the generality is out of balance.

Fundamental problems arise in one-shot object tracking if the model that is used does not represent the actual object adequately. The first reason for this might be an object model that is too general. An object model that is too general will easily fit to other objects in the scene. Imagine for instance the task of tracking a red ball. It is straightforward to design an object model that exclusively responds to red colors. Such a simple object model will fit the red ball perfectly well, however it will also fit to any other red object in the scene, for instance the red shirt of a child playing with the ball. In the object tracking community, the challenge when properties expected of the object of interest are exhibited by other objects in the scene has been referred to as *clutter*.

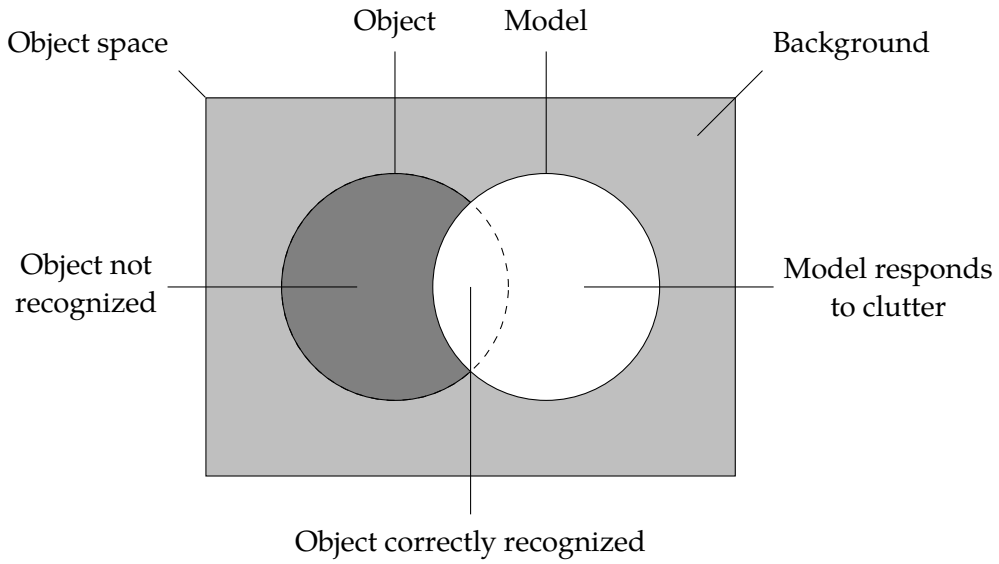


Figure 1.3: Relationship between the object of interest, the object model and other objects in the scene (clutter).

To stop the model from fitting the child's shirt, but still keep fitting the red ball, one might have the idea to incorporate the shape of the ball into the object model, which from now on responds only to objects that are both red and round. By doing so, we have decreased the model generality and made it more specific. But what if the child decides to squeeze the ball? What if the child occludes a part of the ball? What if another red ball enters the scene? In these cases, the model is no longer general enough to capture the essence of the object of interest.

This situation is illustrated schematically in Figure 1.3, where the light gray rectangle refers to the appearance of all imaginable objects in videos. The dark gray blob refers to all possible appearances of the object of interest (e.g. the ball) and the white blob refers to the appearances of all objects that the model responds to. The overlapping area of the two blobs hence are the appearances of the object that the model can recognize and left and right of it are missed object appearances and clutter, respectively. A more general model therefore means that the white area gets larger, while a more specific model means that the white area gets smaller. At the same time, objects that occupy a large space in the diagram exhibit a lot of appearance variations, while objects occupying a small area can be considered more stable. While this figure oversimplifies things slightly, it illustrates that one-shot object tracking is a very difficult problem, as no clue is available beforehand about the complexity of the object of interest. Aside from this general consideration, we will now turn to specific circumstances that make objects more difficult to track.

The main visual challenge in one-shot object tracking are *appearance variations*. These appearance variations come about by a multitude of factors, which we will broadly group

into two categories. In the first category we find *extrinsic* appearance changes. These changes refer to differences in the relationship between the object of interest and other objects in the scenes, of which the camera itself is probably the most important one, as it is always present. As soon as either the object or the camera is moved, the object appears in the image in a *transformed* way. Some of these transformations are relatively easy to handle. Translation, scaling and in-plane rotations[4] merely change the geometry of the projected image. As soon as out-of-plane rotations can occur, unseen visual object data can become available. At the same time, some object parts might disappear due to self-occlusion. Even a rigid object might appear in a *deformed* way due to the variety of possible transformations. Another challenge in this category are partial occlusions, which occur whenever parts of the object are not visible due to other objects in the scene. Other extrinsic appearance changes can be caused by different global or local illumination. The former can typically be handled by employing suitable features that compensate for different levels of brightness. The latter, however, can have much more severe effects and it is currently not clear how to handle these changes in a principled manner.

The second category contains *intrinsic* appearance changes. These changes are caused by the object itself, typically by non-rigid movement. Many object classes can exhibit this kind of movement, such as humans or animals when moving arms and legs. Also inanimate objects can show articulated motion, for instance vehicles with movable parts such as doors or wheels. Similar to the extrinsic deformation of the object, the intrinsic deformation causes changes in the appearance of the object that are very hard to predict and present major obstacles for creating robust one-shot tracking algorithms. In summary, we go so far as to say that the deformation of objects is the major source of errors in one-shot object tracking. In addition to the challenges presented here, there are numerous other challenges that make object tracking a difficult task, such as artifacts caused by the recording process. For a comprehensive enumeration of these challenges we refer the reader to the surveys mentioned in Chapter 2.

1.4 Contribution

This work aims at making several contributions to the field of one-shot object tracking that we will present in the subsequent chapters. The principal contribution of this work addresses the main challenge presented in the previous section, namely the deformation of objects. There are a number of approaches in one-shot tracking that have attempted to deal with this challenge by breaking down the object model into multiple parts. We will discuss these part-based methods in Chapter 3, where we distinguish between basic part-based models, constellation models and star-shaped models. For each model, we

[4] In a 3D-coordinate system with the camera pointing down the z axis, an in-plane rotation refers to a rotation of the object around the z axis, while the rotation around the x or y axis are referred to as out-of-plane rotations.

will discuss strengths and weaknesses.

Chapter 4 then introduces our contribution, an extension to the star-shaped object model that allows for dealing with deformations within the model in a more principled manner than what was previously possible. We refer to this contribution as the Deformable Part Model for One-Shot Object Tracking (DPMOST). While the DPMOST rests on simple assumptions, a rich set of properties arises from them. We will discuss these properties, show how the DPMOST can be made invariant to scaling and rotation and show its deep connection to agglomerative clustering techniques. As the DPMOST is independent on the actual method that is used for establishing part correspondences, we discuss this topic at length in Chapter 5. Here, we take the view that existing correspondence methods either lead to static correspondences or adaptive correspondences. We argue that these two types are fundamentally different from each other. Static correspondences are based on certain information, such as descriptors found in the initial object bounding box. This information is likely to be correct and should not be updated with new information. On the other hand, adaptive correspondences are computed from frame to frame, thus delivering an up-to-date view of the object of interest at the cost of a high uncertainty. While static correspondences are able to provide robustness, adaptive correspondences are able to provide accuracy. We propose a way of maintaining the advantages of both of these paradigms in the form of static-adaptive correspondences. Additionally, we show how static correspondences can benefit from the disambiguation of reference descriptors. This technique is another contribution of this work and refers to the question how an initial estimate of correspondences can be used to obtain a more accurate solution. To evaluate our proposed concepts in practice, we formulate the one-shot tracking approach CMT with the DPMOST being at its core.

To compare a tracking algorithm to the state of the art in a fair manner, a common evaluation framework is needed. in Chapter 6 we first discuss relevant evaluation meaures and methodologies and shed a new light on the overlap measure that is heavily used in computer vision in general and propose a potential alternative. Recently, authors of one-shot tracking algorithms began to employ a method for visualizing tracking results that we call success plots. We show an interesting connection of success plots to empirical cumulative distribution functions (ECDFs), leading to a new interpretation of the commonly performed operation of computing the area under the curve of a success plot. We provide a rigorous experimental evaluation to assess the performance of CMT and thus the plausibility of our contributions. This experimental evaluation contains a detailed analyis of the internals of the DPMOST, quantitative results on a dataset as large as 77 sequences, as well as qualitative results. Lastly, Chapter 8 provides a summary of our work and discusses potential future research directions. We close with an assessment of the current state of one-shot object tracking and propel ideas how to advance the field in general.

Before we present the individual contributions in detail, we give a broad overview about the field of one-shot tracking literature in the upcoming Chapter 2. This chapter is

meant to provide an introduction to the unfamiliar reader with common concepts used by one-shot tracking researchers to deal with the various challenges encountered during the exciting task of tackling one of the most exciting problems science has to offer - how to make a computer see like a human.

Chapter 2

Overview about One-Shot Tracking

While tracking is a relatively young field of study, the tracking literature is massive. Researchers have undertaken attempts to provide overviews about object tracking approaches in three qualitative surveys by Yilmaz et al. (2006), Cannons (2008) and X. Li et al. (2013), of which Cannons arguably provides the most comprehensive and at the same time the most underrated one. While all of these surveys aim at providing a broader overview about the topic of visual object tracking, many, if not most of the presented concepts are applicable to one-shot object tracking as well. In addition to these qualitative surveys, large-scale experiments have been undertaken recently by Wu et al. (2013), Kristan et al. (2013) and Smeulders et al. (2014) to assess the performance of different one-shot tracking algorithms quantitatively. Last, but not least, there exists a book dedicated exclusively to object tracking (Maggio and Cavallaro, 2011).

In general, designers of one-shot tracking algorithms focus on four central algorithmic aspects, as shown in Figure 2.1, which the following four sections are devoted to. Section 2.1 discusses the *prediction* step that provides an estimate of the object position based on previous information. In Section 2.2 the *feature extraction* step is covered that identifies and retrieves the relevant information from the image. Section 2.3 deals with the *localization* step, in which the final position of the object is determined. Finally, Section 2.4 explores the *update* step, stating how new information can be incorporated into the object model to adapt it to new appearances of the object. The concatenation of these four steps is referred to as the *object model*.

2.1 Prediction

Even before the actual image content is analyzed, it is possible to tell something about the position of the object of interest by making certain assumptions about the object *motion*. For instance, if the object of interest was observed to move at constant speed into a single direction, it might be reasonable to assume that in the next frame this motion is continued, making it possible to *predict* the object position. We will now briefly describe prediction techniques that have been used in the tracking context.

The Kalman Filter (Kalman, 1960) provides under some very strong assumptions an optimal way of estimating the hidden state of a system (for instance the position of an object) when only noisy measurements are available (such as a localization performed on

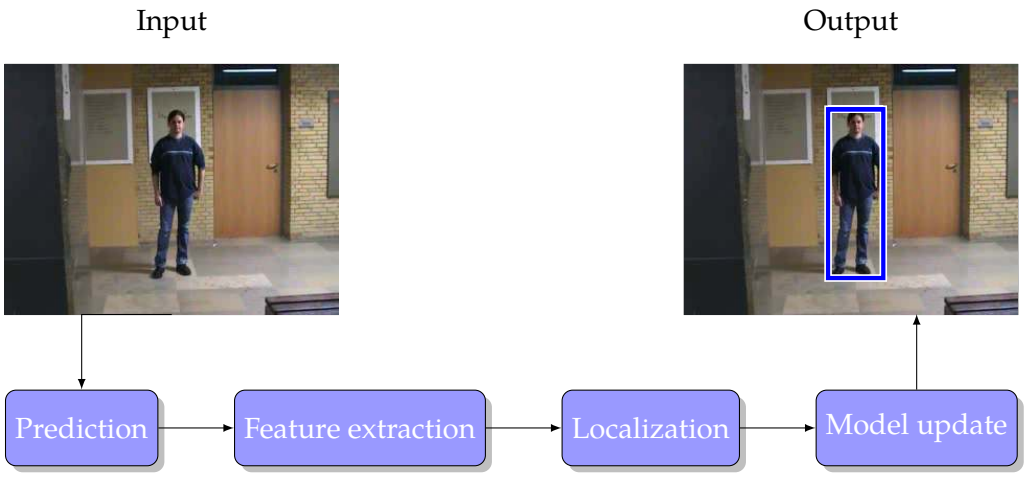


Figure 2.1: The tracking pipeline that is found in virtually every single one-shot tracking algorithm. The concatenation of these steps is referred to as the object model.

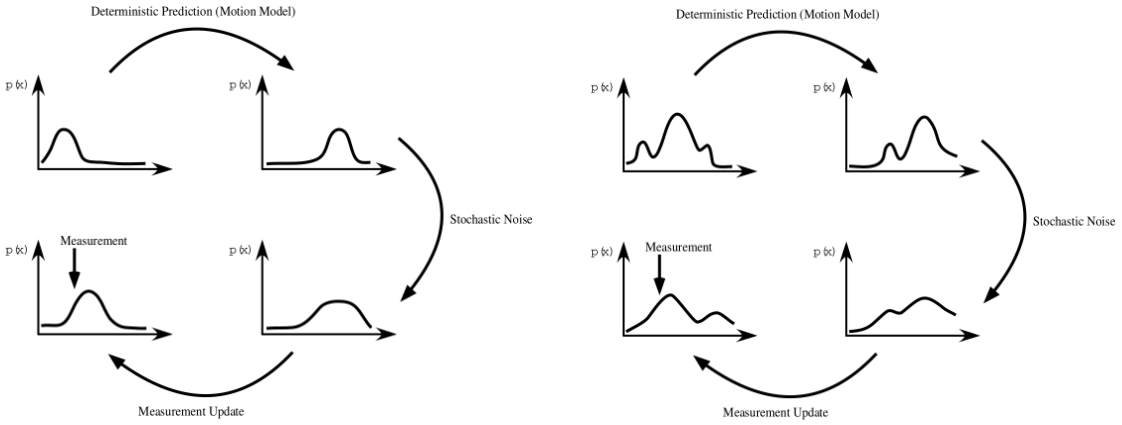
image data). The main steps of the Kalman Filter are shown in Figure 2.2a, of which we focus here only on the prediction step, where the new state of the system is predicted by using a motion model of the target. In the Kalman Filter, a general assumption is that all variables stem from Gaussian distributions. The estimate for the current state is defined as

$$x_t = Ax_{t-1} + Bu_{t-1}, \tag{2.1}$$

where x_{t-1} is the state from the previous iteration and A is the deterministic model that relates the previous state to the current state. B and u allow for modeling control input, which is usually ignored in computer vision applications. The nature of A as a matrix restricts the prediction to a linear model, which is one of the main drawbacks of the original Kalman Filter. This issue has been addressed by the Extended Kalman Filter (EKF), allowing instead for differentiable functions to be used[5]. The Kalman Filter is useful when the statistical properties of a system are well-known and can be can be modeled using Gaussian distributions. The latter condition does however not apply to object tracking, as clutter in the scene typically causes the measurement function to form several modes instead of one, thus conflicting with the assumption of Gaussian distributions. This particular drawback has been adressed by Particle Filters. This type of filtering has a similar general setup as the Kalman Filter (Figure 2.2b), but removes the requirement to model the distributions analytically. Instead, samples of possibly highly complex distributions are propagated, thus allowing for multiple modes in the distributions. This property allowed the Particle Filter to be used in tracking of objects through cluttered scenes, which was prominently done in the CONDENSATION algorithm (Isard and Blake, 1998).

In summary, prediction is useful when suitable assumptions about object motion can

[5] For details refer to Welch and Bishop (1995).



(a) Kalman Filter (b) Particle Filter

Figure 2.2: Filtering approaches used for predicting the object position. (a) In the Kalman Filter all variables are modeled as Gaussians, leading to unimodal distributions. (b) The Particle Filter allows for multimodal distributions, allowing for tracking through clutter. Image is from Cannons (2008).

be made, which is feasible in restricted scenarios such as tracking in radar data (Wax, 1955). In one-shot tracking scenarios similar assumptions can hold if the movement of the object is not overly abrupt. However, in the case of shaking cameras or any other kind of abrupt motion change, these assumptions fail. Another problem that occurs when relying too much on the results of previous frames is the problem of error accumulation (Lepetit and Fua, 2005), eventually leading to tracking failure. Often, researchers employ simple motion models where the object is allowed to move within a certain radius to reduce the search space of an object detector (Grabner and Bischof, 2006; Babenko et al., 2009).

2.2 Feature Extraction

One of the most important aspects of every computer vision algorithm is the question of what image information should be used, ususally referred to as *feature extraction*. Essentially, a feature is a function from image space to a feature space. The basic idea of using features is to transform the image into a different representation, in which objects or parts form invariant manifolds and thus can be better separated from each other than in the image space.

Traditionally, hand-crafted features have been used in the tracking literature, meaning that the feature extraction process is designed after some guiding principle or best practice. One exception to this rule are boosting-based techniques that combine many simple features into a more powerful one in an online manner (Grabner and Bischof, 2006). In the area of object recognition there recently has been a paradigm shift regarding

feature extraction sparked[6] by the work of Krizhevsky et al. (2012), who demonstrate that learning feature hierarchies from large labeled datasets is not only possible, but leads to tremendous performance improvements. The one-shot tracking community has been relatively unaffected by this novel feature extraction process so far (with the recent exception of H. Li et al. (2014), who learn a hierarchy of features online), presumably due to the absence of training data in one-shot scenarios. In this overview, we therefore focus on hand-crafted features only.

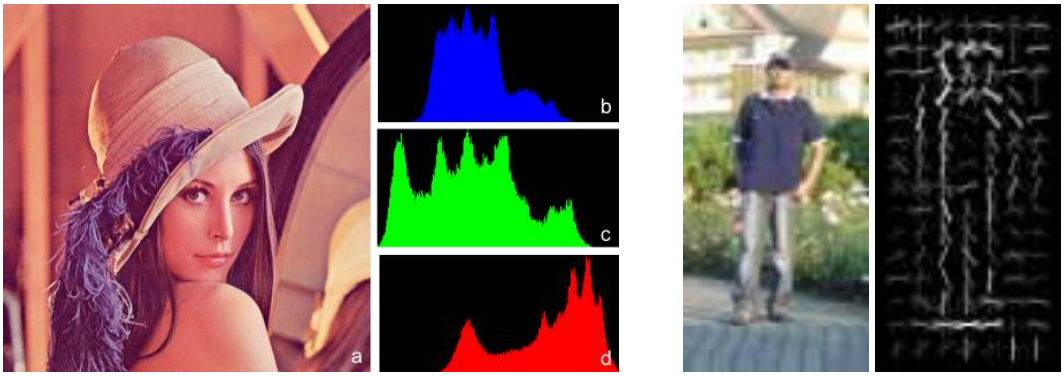
The general aim is to find features that are both discriminative and invariant to various transformations. A feature is discriminative if it can be expected only on the object of interest and does not occur on other objects or spuriously in the background. At the same time, feature invariance is also important, meaning that the outcome of a feature computation should not change even if the object of interest undergoes certain transformations, such as scaling, rotation or even affine motion. It has been argued that these two properties should be traded off, depending on the application (Varma and Ray, 2007).

One can in principle distinguish between global and local features. A global feature is designed to represent the information of a whole image region. In contrast, a local feature is designed to be evaluated at *interesting* image positions such as corners. While it is certainly possible to compute global features at interesting image positions or local features over image subregions, this is not necessarily a good idea, as the balance between invariance and discriminative power might not be given. We will describe selected global features that have been used in the tracking literature in Section 2.2.1. While there exists an abundance of different local features, such as edgels or small image patches (Tuytelaars and Mikolajczyk, 2008), we restrict the discussion of local features to detectors of interest points in Section 2.2.2 and to the computation of their descriptors in Section 2.2.3. For a comprehensive overview about feature extraction in object tracking, the reader is referred to X. Li et al. (2013). Please note that the available literature to the topics in the upcoming sections is massive. The goal of these sections therefore is to provide an overview about the different underlying paradigms as well as to introduce some techniques that will be referred to in later chapters. A profound discussion of the topic of feature extraction is out of the scope of this work.

2.2.1 Global Features

The canonical global feature of an image region is the image region itself, often referred to as a *template*. It is prominently used in the method of Lucas and Kanade (1981) to compute the sparse optical flow between two images. However, also higher-level tracking algorithms still employ templates as a representation (Ross et al., 2008; Kalal et al., 2012). In spite of their simplicity, templates are a very discriminative feature. However, already

[6] In fact, the technique used by Krizhevsky et al. (2012) was already developed in the Eighties, but was made practical only by recent advances in hardware and large amounts of training data.



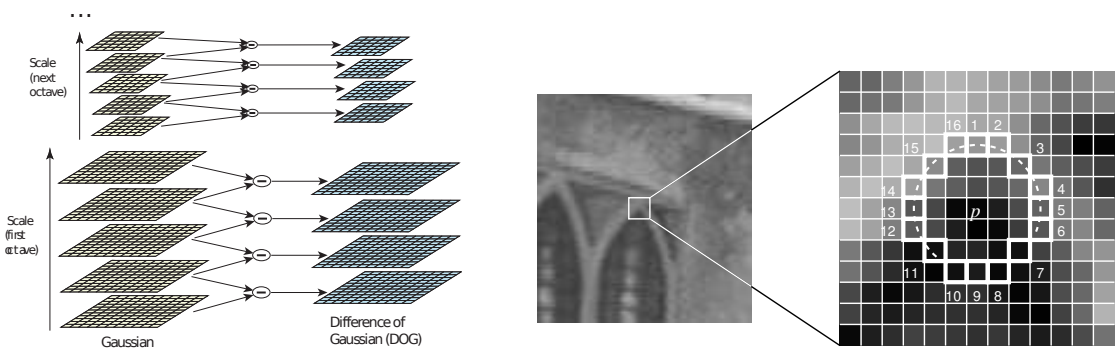
(a) Color histogram (b) HOG descriptor

Figure 2.3: Visualization of different global features: (a) color histogram, (b) Histogram of Oriented Gradients (Dalal and Triggs, 2005).

small deviations in the object appearance might make it impossible to establish the association between the target template and the object of interest.

One of the most widely used features in object tracking is the color histogram, shown in Figure 2.3a. It has notably been used in the Mean-Shift tracking algorithm by Comaniciu et al. (2000), but keeps occurring in more recent tracking approaches as well (Zhao et al., 2010). A color histogram assigns each pixel a bin depending on its color value. Typically, the bins are spaced such that the color space is divided into equal intervals. Spatial information in a color histogram is disregarded, leading to both the desired effect that different poses of the object of interest are accounted for (the feature is invariant to different poses) and the rather undesired effect that objects with similar color compositions produce similar histograms (the feature is not discriminative). Additionally, colors are heavily influenced by changes in global and local illumination as well as different camera devices, which is especially harmful when objects should be tracked over multiple cameras (Rinner et al., 2015). However, when the object of interest consists of unique colors, the color histogram still performs remarkably well in single-camera scenarios.

In order to avoid the negative of aspects of using color information, researchers have focused on feature extraction from gray-scale images. For less expressive objects, the Histogram of Oriented Gradients (HOG) has proven to be a remarkably good image descriptor (Dalal and Triggs, 2005). As shown in Figure 2.3b, the descriptor focuses on gradients on the object of interest. Unlike the classical color histogram mentioned above, in the HOG descriptor local cells are used when summing up the gradients. HOG descriptors are known to work very well as the input to linear classifiers, such as support vector machines (Cortes and V. Vapnik, 1995). Initially proposed for the detection of humans, the HOG descriptor has become the de-facto standard hand-crafted descriptor in object recognition and has also found its use in one-shot object tracking for instance by Yu et al. (2008) and Danelljan et al. (2014).



(a) SIFT (b) FAST

Figure 2.4: Interest point detection. (a) Scale-Invariant Feature Transform (Lowe, 2004). (b) Features from Accelerated Segment Test (Rosten et al., 2010).

2.2.2 Interest Point Detectors

The principle idea of employing an interest point detector is to find image locations that differ from their immediate environment. According to Tuytelaars and Mikolajczyk (2008), the most important property of an interest point detector is its repeatability, meaning that given two images of the same scene, a high number of corresponding interest points should be detected.

A class of interest point detectors that was studied early is based on measuring self-similarity. Moravec (1980) proposes to use those pixels as interest points where in an 8-neighborhood one of the sum of square differences between a candidate patch and the neighboring patch is high. The main problem with this approach is that this operator is not isotropic, meaning that under the presence of an edge that is not exactly horizontal, vertical or diagonal, a point on this edge will be recognized as an interest point, which is not desired. To remedy this circumstance, Harris and Stephens (1988) propose a modification to the Moravec operator by analyzing the partial derivatives of the sum of square differences in x and y direction. Large eigenvalues of the resulting matrix then determine a corner. Shi and Tomasi (1994) propose a slight modification of the Harris corner detector by directly using the minimum of the two eigenvectors as an indicator for corner strength. In practice, this change leads to a detection of more suitable corners.

The previously mentioned methods provide only the position of the interest point, but do not consider different scales the corners might appear in. In his seminal work, Lowe (2004) proposes a method called SIFT (Scale-Invariant Feature Transform) that detects interest points in a scale-invariant manner. Here, minima and maxima of the difference of Gaussians operator applied in scale space define interesting image regions, as shown in Figure 2.4a. Additionally, to make feature extraction on these interest points invariant to rotation, the orientation of the interest point is computed by extracting the gradient magnitude and direction in a window around it.

Only recently, researchers have studied how to detect and describe local features more

efficiently. Rosten and Drummond (2005)[7] propose an interest point detector called FAST (Features from Accelerated Segment Test), the idea of which is depicted in Figure 2.4b. Here, a circle of pixels is considered around the corner candidate. If there exists a set of contiguous pixels in the circle which are all brighter than the candidate pixel plus a threshold, or all darker minus a threshold, then the candidate is added to the list of corners. Rosten et al. devised a mechanism that evaluates those pixels first that are most likely to result in an information gain. This way, many candidates can be rejected very quickly.

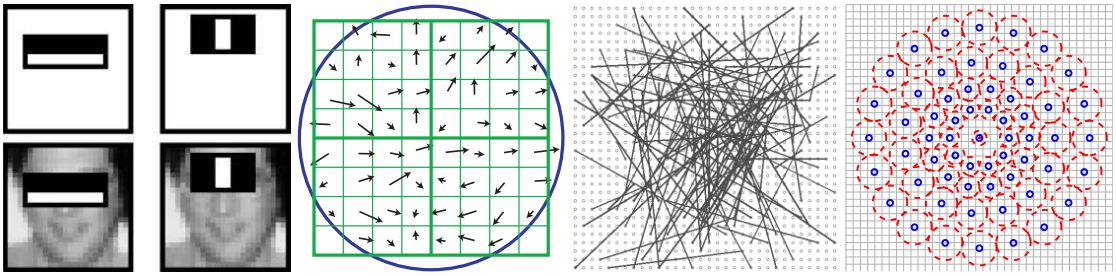
2.2.3 Local Descriptors

Haar-like features (Figure 2.5a) were popularized in their seminal work by Viola and Jones (2001) for face detection. The output of a Haar feature is defined by the subtraction of the sum of all pixels from two or more disjoint image regions. In spite of their apparent primitivity, the combination of many of these features leads to a powerful descriptor. As in Viola and Jones (2001), suitable positions for Haar features are often learned in a training phase. An appealing property of Haar features is that they can be evaluated in constant time when integral images are used. Haar features are known to work very well on faces, as they capture their brightness distribution effectively. Haar features have been used in a tracking context by Grabner and Bischof (2006) for online selection of features as well as in Kalal, Matas, et al. (2010).

In SIFT (Lowe, 2004), a descriptor similar to HOG is computed around interest points, as shown in Figure 2.5b, capturing local gradient directions around the interest point. While the SIFT descriptor has initially been rejected by the computer vision community as not being principled enough, it was demonstrated in practice that it outperformed basically every other feature descriptor that existed to this date. One reason why SIFT has not been used extensively in the tracking literature (with the recent exception of Pernici and Del Bimbo, 2014) is its relative expensive computation. To remedy this circumstance, (Bay et al., 2006) propose SURF (speeded-up robust features), decreasing the computing time considerably.

Lepetit et al. (2005) propose an interesting perspective on the simultaneous description and matching of interest points. In their work, a classifier is trained during a learning phase that can then distinguish different interest points on the object. Based on a single example of the object of interest artificial training data is synthesized by warping the initial patch with affine transformations. The classifier itself consists of multiple Random Ferns (Özuysal et al., 2007) which are similar in spirit to Random Forests (Breiman, 2001). In Random Ferns, very high classification speed can be obtained by performing fast binary tests on random pixel pairs on each layer of the random ferns instead of computing a full image descriptor. These tests merely measure which of the two pixels has a higher brightness value. It has to be noted that the pixel comparisons are generated only once,

[7] Also see Rosten and Drummond (2006) and Rosten et al. (2010).



(a) Haar-like (b) SIFT (c) BRIEF (d) BRISK

Figure 2.5: Different local descriptors. (a) Haar-like features (Viola and Jones, 2001), (b) Scale-Invariant Feature Transform (Lowe, 2004), (c) Binary Robust Elementary Independent Features (Calonder et al., 2010), (d) Binary Robust Invariant Scalable Keypoints (Leutenegger et al., 2011).

so that future comparisons yield meaningful results.

Based on the idea of Random Ferns, Calonder et al. (2010)[8] propose an extremely fast keypoint descriptor called BRIEF (Binary Robust Elementary Independent Features), as shown in Figure 2.5c. The pinciple idea is to compute a feature composed of binary elements only. Similar to Random Ferns, each element of the vector is determined by a simple brightness comparison between two randomly determined pixel positions within a predefined area around the keypoint. These binary descriptors can be compared efficiently by employing the Hamming distance as a similarity measure. This is equivalent to a bitwise XOR operation, which can be performed efficiently on modern computing architectures. Rublee et al. (2011) extend this concept in their work called ORB (Oriented FAST and Rotated BRIEF) by introducing invariance to rotation by incorporating corner orientations.

Recently, authors have investigated how to improve upon the random arrangement of the BRIEF pattern. As shown in Figure 2.5d, in BRISK (Binary Robust Invariant Scalable Keypoints) Leutenegger et al. (2011) apply Gaussian smoothing with different kernel sizes to individual patches around test positions, as denoted by the blue dots and the red circles, respectively. Additionally, the test positions are arranged on concentric circles, allowing for invariance to rotation. Alahi et al. (2012) employ a similar, but more biologically inspired idea. In their approach called FREAK (Fast Retina Keypoint) they propose to employ a higher density of test positions at the center of the keypoint, similar to how the human eye has evolved. Additionally, the evaluation of the descriptor is performed only gradually to save processing time, as many non-informative keypoints can be discarded by looking at very few test positions.

[8] Also see Calonder et al. (2012).

2.3 Localization

The localization step in one-shot object tracking refers to the question how the extracted features are used to infer the position of the object of interest. These models can be broadly divided into two categories. Approaches that employ error surfaces or similarity functions are often used for a *local search* (Section 2.3.1). On the other hand, approaches that employ a classifier to distinguish between the object and background in feature space are referred to as *tracking-by-detection* methods (Section 2.3.2).

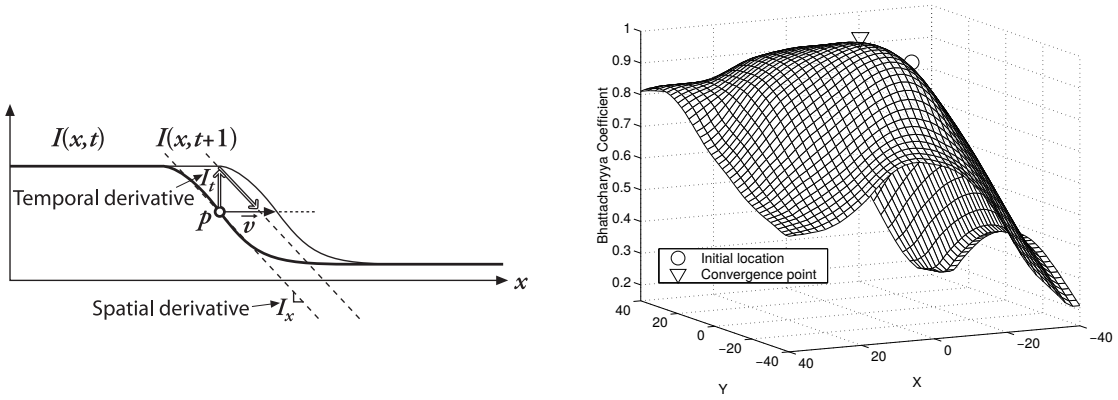
2.3.1 Local Search

As discussed in Section 2.1, the predicted motion of the target in one-shot tracking is not a helpful cue, if strong temporal continuity cannot be guaranteed. However, as noted by Cannons (2008) the information from previous frames is still valuable and can be used to serve as a starting point for a local search. The principle assumption behind employing a local search in object tracking approaches is that the differences between adjacent video frames should be small.

One approach that exploits this assumption directly is the method of Lucas and Kanade (1981). This method, initially proposed for image registration, is typically used to compute the sparse optic flow between video frames, which is defined as the motion between two images at a specific image position x. It is important to note that in contrast to the motion models discussed in Section 2.1 the motion model employed by Lucas and Kanade does not depend on the motion observed in previous frames. Without going into mathematical details, the principal idea of Lucas and Kanade is to perform an iterative local search based on Newton's method for estimating the displacement v of the point x, as shown in a simplified manner in Figure 2.6a for the one-dimensional case.

Another example for a local search in tracking was proposed in the Mean-Shift tracker of Comaniciu et al. (2000), who model the object of interest as a color histogram. The basic idea is to perform a local search in the current frame beginning from the last known object position in the previous frame. By defining a similarity function between the object template and the image content, a surface similar to Figure 2.6b is created, where the circle denotes the starting point of the local search. The idea is then to iteratively reach the maximum of this compatibility surface (the triangle in Figure 2.6b), which is carried out in this work by the Mean-Shift algorithm. This simple algorithm states that the new position should be moved towards the center of mass of the compatibility surface. Typically, only few interations are necessary to reach this goal.

While both of these methods are more complex than what is described here, some important conclusions can be drawn from them. For a local search to work, a more or less smooth error function or compatibility function has to be defined, that can be used to find interesting positions on these functions, such as minima, maxima or zeros. These error surfaces offer a very fast way of obtaining an accurate localization. Compared to tracking-by-detection methods, which we will discuss in the next section, object models



(a) Optical flow (b) Mean-Shift

Figure 2.6: Localization by local search. (a) Optical flow in one dimension as in the method of Lucas and Kanade (1981). The displacement $\vec{v}$ is iteratively estimated by means of the the spatial derivative I_x and the temporal derivative I_t. Image is from Bradski and Kaehler (2008). (b) Mean-Shift tracking (Comaniciu et al., 2000). Object localization is performed by iteratively climbing to the maximum of a compatibility surface, in this case the Bhattacharyya coefficient.

for local searches are kept relatively simple, leading to a certain lack of robustness. This is clear by appreciating that local searches can always get stuck in local extrema before the global maximum is reached. However, in certain circumstances this property can even be advantageous, for instance when a similar object appears that by chance achieves a higher similarity score. In this case, it is advantageous to look for the closest local maximum instead of the global maximum. Additionally, a local search can be much faster than a global one.

2.3.2 Tracking-by-Detection

While relying on a purely local search is accurate over few frames, in the long run there is the chance that a local search is disrupted, which can for instance occur by abrupt camera motion. In recent years, there has been a trend in tracking research of freeing oneself from strong assumptions and thus ignoring positional information from previous frames altogether. This tracking paradigm has become known as *tracking-by-detection*. Tracking-by-detection is closely related to classical object detection in still images. The main difference however is that no offline training data is available for training classifiers. A common route in tracking research therefore has been to adapt offline learning techniques to the online domain (Grabner and Bischof, 2006; Babenko et al., 2009; Saffari et al., 2009). In contrast to local searches, object models in tracking-by-detection methods are often incrementally refined to capture the entire variability of the object's appearance.

Numerous tracking-by-detection approaches based on diverse object models have been proposed. These methods have in common the ability of globally searching the

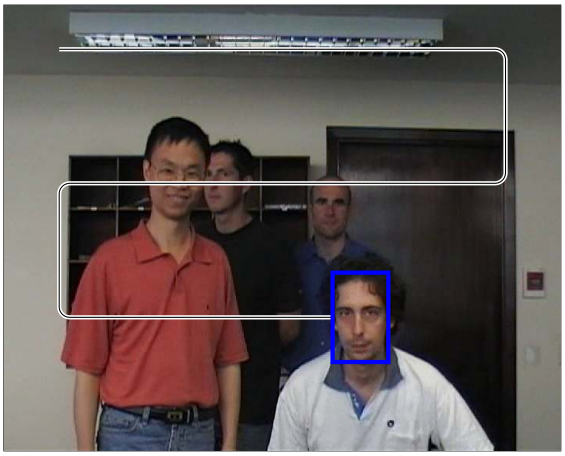


Figure 2.7: Sliding window classification is an essential part of tracking-by-detection approaches, independent of whether the search is performed for objects as a whole or individual object parts.

input image for the object of interest, even though several methods artificially restrict the classifier search space (such as Avidan, 2004) to enforce temporal constraints. In one way or the other, the global search is accomplished by evaluating a classifier in a sliding-window manner, as shown in Figure 2.7. On a first glance, methods that rely on keypoints (Maresca and Petrosino, 2013; Pernici and Del Bimbo, 2014) or other bottom-up techniques (Godec et al., 2011) seem to escape this paradigm. However, to detect the individual object parts a sliding-window classifier is still necessary (Lehmann et al., 2011). As an exhaustive search can be very expensive, it is important to employ an object model that can be evaluated quickly. It can be argued that the sliding-window-based localization of tracking-by-detection methods is less accurate than performing a local search, since for reasons of efficiency not every possible subwindow is evaluated, especially when the search is performed in multiple scales.

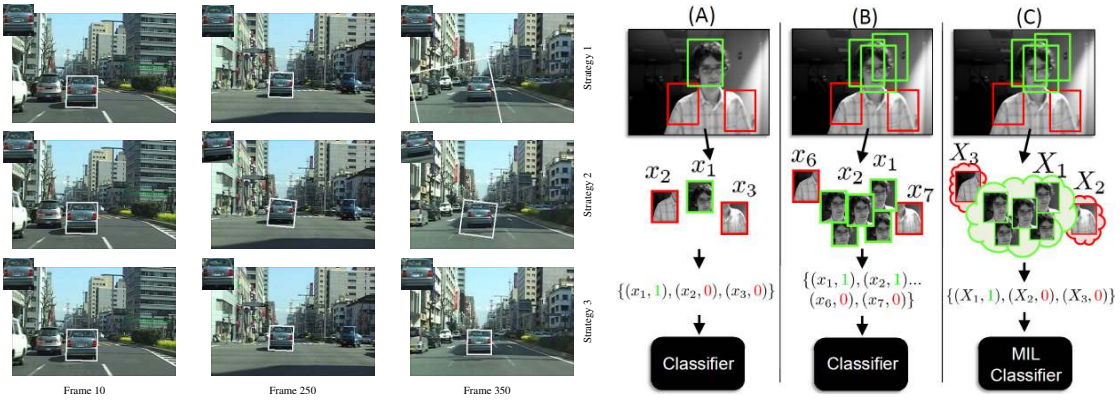
The classifiers that underly every sliding window detector can be broadly categorized into *generative* and *discriminative* classifiers (X. Li et al., 2013). Generative classifiers aim at answering the question what probability distribution might have generated the data. Unfortunately, the probability distributions in computer vision however are typically very complex and non-Gaussian, making this a daunting task. Also, it has been argued that one should never solve a problem that is more general than one actually needs to solve (V. N. Vapnik, 1995). Instead, discriminative classifiers focus on finding a decision boundary between the object and background in feature space, thus allowing more freedom in the design of the classifier. As discriminative classifiers also tend to be faster than generative ones, they are typically favored in tracking-by-detection methods.

2.4 Model Update

A perennial source of failure in object tracking concerns the question of how and when to update an object model so that it remains a good representation of the object of interest. This is important as the sum of internal and external appearance variations let the object inevitably appear in a different way compared to its initial appearance. In this section, we will discuss different paradigms that have been used in the literature to address this aspect. The question of how to update an object model without erasing relevant information is closely related to the stability-plasticity dilemma, which relates to the trade-off between the stability required to retain information and the plasticity required for learning something new (Grossberg, 1987) when the capacity of the model is limited. A complete stable model will retain all information in it but is unable to incorporate any new information. On the other hand, a completely plastic model will immediately adapt to new information, but any information stored in the model to this date will be lost.

Matthews et al. (2004) perform a study on this subject based on templates, but much of this work is applicable to other object representations as well. Matthews et al. suggest that the simplest possible strategy is not to update the model at all, as shown in the top row of Figure 2.8a. While this strategy prevents any erroneous updates, it also is impossible to add any new information to the object model. In case of severe appearance changes this strategy is prone to failure. Another naïve method is to perform a complete update, meaning that in every frame the original object model is replaced with a representation computed in the current frame. This strategy is often used in algorithms performing a local search, such as in Lucas and Kanade (1981). The main problem with this strategy is that it encourages *drift*, as shown in the center row of Figure 2.8a. Drift refers to the situation when the object model gradually adapts to a different object, which is typically the background. The reason for this situation is that output of a tracking algorithm is ususally never aligned perfectly well to the object, where the ubiquitous use of bounding boxes certainly adds its share to this effect. When a mis-aligned bounding box is used to extract new features, there is a chance that instead of the original object, parts of other objects "leak" into the object model. This way, with every update, small errors accumulate until after some time the tracking algorithm adapts to a background object. Matthews et al. (2004) suggest a simple, but very powerful idea for *drift-correction* during model update. Essentially, the idea put forward by Matthews et al. is to perform an update using the newly extracted features only if the localization using the initial model is similar to the localization performed using the latest model, depending on a threshold. While this strategy can be considered to be primitive as well and is in no way guaranteed to yield good results, it can serve as the basic insight that not all model updates are equally plausible. It is safe to say that the constraints that should be employed for updating object models robustly are currently unknown. It is a very interesting research question whether such constraints exists at all. Nevertheless, researchers have come up with heuristics to tackle this problem.

Instead of blindly updating the complete model in every frame, Collins et al. (2005)



(a) Drift correction (b) Multiple Instance Learning

Figure 2.8: Different strategies for updating the model . (a) Drift correction according to Matthews et al. (2004). Top row: No update is performed. Center row: Update is performed in every frame. Bottom row: Update with drift-correction is performed. (b) In MIL tracking (Babenko et al., 2009), training examples are presented in bags, allowing the learner to decide which examples are suitable for updating the classifier.

and Grabner and Bischof (2006) aim at finding the most discriminative features using an online learning mechanism. This way, background features are excluded from the object model. It can however be argued that discriminative features might also appear in the background, for instance due to a mis-alignment of the bounding box. A more principled way of dealing with this topic has been proposed by Grabner et al. (2008), who see object tracking as a semi-supervised learning problem. By imposing a prior on the first image patch, the incoming image data is treated as unlabeled data. Semi-supervised learning algorithms can then be used to label this unlabeled data. However, if the prior is too strong, it is hard for the labeling algorithm to make the connection to the correct unlabeled data. If the prior is too weak, all unlabeled data suddenly becomes very similar to the initial appearance of the model.

Babenko et al. (2009) address the problem of erroneous model updates by employing Multiple Instance Learning, as shown in Figure 2.8b. In standard binary classification, each training sample is either labeled as positive or negative, as shown in columns A and B. In multiple instance learning, training examples are no longer labeled individually, but are presented as labeled "bags", as shown in column C. A positive bag is assumed to contain at least one positive training example, all other bags are negative. This way, the learner has more options in finding a decision boundary, which proves beneficial in object tracking. Another very successful way of leveraging the semi-supervised learning paradigm has been proposed by Kalal et al. (2012), which will be discussed at length in Section 3.1. Here, the optic flow is used as a guiding principle for selecting positive and negative training examples.

In summary, it seems that the best strategy for permanently updating the model is to

perform the updates as conservatively as the object of interest allows. However, certain objects require more aggressive model updates due to their fast-changing appearance. How the update rules come about is a matter of the concrete object model in use and no clear answer can be given in general.

2.5 Conclusion

This chapter has given the reader an overview about the main aspects that have to be dealt with when devising an object model for the problem of one-shot object tracking. It should have become clear by now that one-shot object tracking is not a trivial task and requires a well-working interplay between the different subproblems prediction, feature extraction, localization and update. In the next chapter we will disuss strengths and weaknesses of a special class of object models, where the object model is broken down into individual parts. Here, all of the steps presented in this chapter are not considered for the object as a whole, but rather on the level of individual object parts.

Chapter 3

Part-based Object Models

In the previous chapter, we have given a very broad overview about the field of one-shot object tracking and have discussed important aspects that object tracking researchers have focussed their attention to. A fundamental question in one-shot object tracking that we discuss in this chapter is whether to model the object as a holistic[9] entity or to break down the object model into parts. Part-based object models are employed in order to address some of the challenges discussed in Section 1.3. As an example for a one-shot algorithm that is based on a global object model, in Section 3.1 we discuss the recently proposed object tracking method TLD. To appreciate why part-based models can solve some of the challenges encountered when global object models are used, we then discuss general properties of part-based object models and contrast them to global object models in Section 3.2. Next, we present and analyze general robust part-based model-fitting techniques that have been used for computer vision in Section 3.3. We will then look closer at two types of part-based models that have been used in the one-shot tracking literature, namely constellation models in Section 3.4 and star-shaped models in Section 3.5. Finally, we discuss the part-based one-shot tracking approach HoughTrack in Section 3.6. Note that the content is this chapter is not meant to provide an exhaustive overview about the numerous part-based models that exist in computer vision, but rather to discuss interesting work that is relevant to the problem of one-shot tracking.

3.1 Global Object Model in TLD

In their work called TLD (Tracking-Learning-Detection), Kalal et al. interpret object tracking as a semi-supervised learning problem. In semi-supervised machine learning, a learner is presented only with a small number of labeled examples. Additionally, at its disposal is a ususally much larger number of unlabeled examples. As mentioned before, the aim of semi-supervised learning is to find a decision boundary between positive and negative examples that takes into account the unlabeled data, leading to a hopefully better result than relying on the labeled data alone. In classical machine learning, clustering methods are often used to find structure in the unlabeled data (Chapelle et al., 2006). Kalal et al. instead employ a relatively simple short-term tracker as a way of collecting

[9] We use the terms “holistic” and “global” interchangeably in the context of object models.

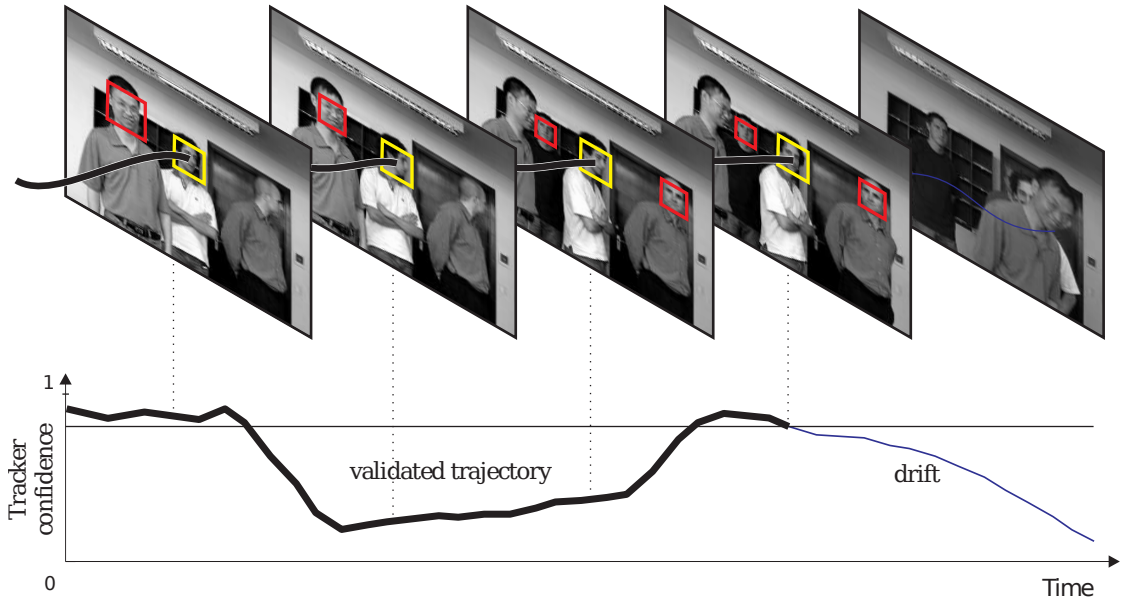


Figure 3.1: In Tracking-Learning-Detection (Kalal, Matas, et al., 2010), a short-term tracker (shown in black) is used to collect positive training data (shown in yellow) and negative training data (shown in red) for an object detector.

positive training examples, as shown in Figure 3.1. These positive training examples are used to train an object detector that is based on a global object model with the aim of being able to re-initialize the tracking process when the optic flow component loses the target. Importantly, false positive responses from the object detector are fed back to the training process.

Kalal et al. employ a cascaded classifier for object detection, as shown in Figure 3.2. Each of these stages contains an increasingly complex classifier. The first stage is a variance filter that rejects homogeneous image regions, the threshold of which is determined by the variance of the first image patch. The second stage consists of a Random Fern classifier (Özuysal et al., 2010), an ensemble classifier whose most important property is its extremely high classification speed. Random Ferns come with the cost of working with binary feature vectors only. Kalal et al. employ a feature descriptor that is reminiscent of BRIEF (Calonder et al., 2012). The final stage of the classifier cascade is a nearest neighbor classifier based on the normalized cross-correlation of resized image patches (15×15 pixels) as a distance measure. The classifier cascade is evaluated in a sliding-window manner over the whole image in every frame. If after a non-maximal supression step exactly one detection remains, the short-term tracker is re-initialized to this detection.

While the object detection component is relatively complex, the short-term tracker in TLD is considerably simpler. It is based on the estimation of optic flow using the method of Lucas and Kanade (1981). In frame I_{t-1}, points on the object of interest are sampled on a regular grid. The position of each of these points in frame I_t is then estimated by

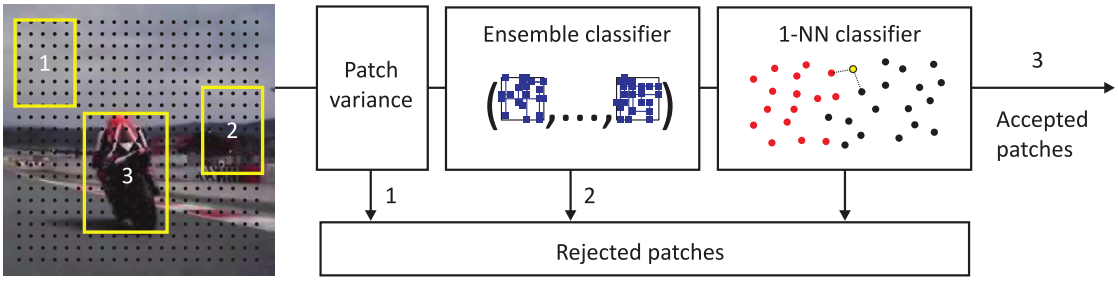


Figure 3.2: The object detection cascade in TLD (Kalal et al., 2012) consists of increasingly complex classifiers.

computing the optic flow and interpreting it as a displacement vector. An estimate for the bounding box is obtained by computing the median of the translational changes and changes in scale and transforming the bounding box of I_{t-1} accordingly. By using a mechanism called forward-backward tracking (Kalal et al., 2010), the robustness of this method is further increased, as erroneous optic flow estimates can be identified. This works by computing the optic flow in not only in forward direction (from I_{t-1} to I_t), but also in backward direction (from frame I_t to I_{t-1}). This way, a plausibility score for the correctness of a tracked point can be obtained. Kalal et al. then select the highest scoring points for continued processing and discard the other points.

In a learning step, Kalal et al. update the Random Fern classifer and the nearest neighbor classifier both with positive and negative training data. In case the object detector did not give a positive result, it is updated with positive training data stemming from the current result of the short-term tracking process, if available. In case the object detector identified several image regions as positive, the ones that do not agree with the short-term tracker are used as negative training examples to prevent future false positives. It has to be noted that the learning step depends on a number of carefully engineered rules about when exactly an update is performed.

The use of a global object model enables Kalal et al. to come up with an update rule that is remarkably robust in practice. Slight appearance changes are incorporated into the object model at ease due to their global similarity to the object model. Interestingly, while the object detector is based on a global object model, the short-term tracker is based on a simple part-based model, suggesting that a part-based representation is also feasible for the object detector. While TLD is able to achieve excellent tracking results, there are a number of downsides that can be attributed to the use of a global object model:

- TLD highly depends on an initialization of the object where little to no background is contained in the initial bounding box. If this consideration is not followed, all similarity measures yield low values in case the object moves to a different background, leading to the inability of detecting the object of interest.
- TLD is unable to to handle the case when a large part of the object interest becomes occluded abruptly, as all classification stages analyze the object holistically.

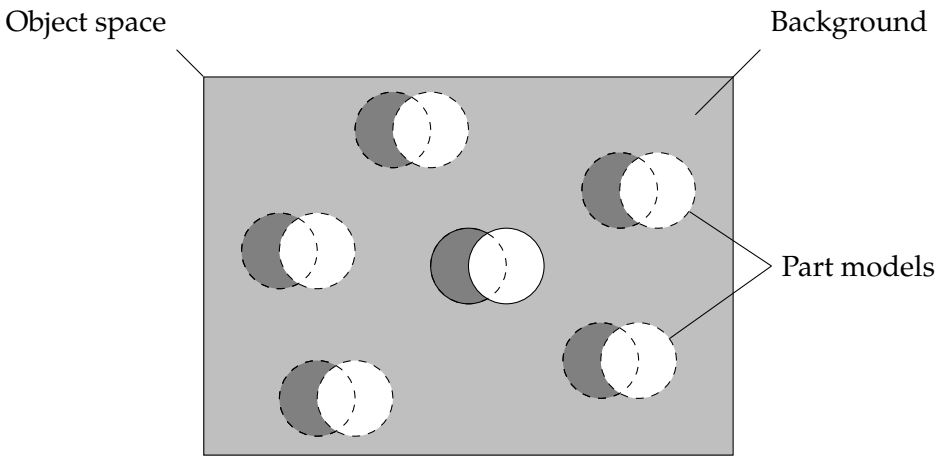


Figure 3.3: In contrast to holistic object models, part-based models provide a way of decomposing an object model into more specific submodels.

- When the object becomes occluded gradually, the occluding parts are incorporated into the object model instead of being recognized as outliers, which is not desired.

The problems presented here are not found exclusively in TLD, but represent fundamental problems that occur when global object models are used. In the next section, we will explore how part-based representations can remedy these issues, while opening up some new questions.

3.2 Promises and Challenges of Part-Based Object Models

To appreciate the principal motivation behind employing part-based object models, it is useful to recall Figure 1.3 and to contrast it with Figure 3.3, where a part-based object model and its relationship to the object space is presented. Here, the single entity of a holistic model has been replaced by multiple sub-models. It is important to understand that these individual part models can and should be much more specific according to our definition of generality from Section 1.3, essentially leading to a smaller occupied area in the object space[10]. It is also clear that these more specific individual part models still suffer from similar challenges as a holistic object model. However, by breaking down the object model into individual, possibly independent parts, the *redundancy* in the overall model is increased. This means that if a number of parts fail to be recognized correctly or provide inaccurate localization information it is still feasible to correctly track the overall object.

So far we have discussed considerations about the appearance of part-based models. Another fundamental question is how the individual parts in a part-based model relate to

[10] This does not necessarily imply that the parts themselves have to be small in the sense that they occupy less image space.

each other with respect to their spatial structure. It is not easy to answer in general what level of independence to allow individual parts. If all parts completely depend on each other this leads to a very rigid part layout that does not have any advantage over a holistic object model. On the other extreme, each part could be tracked completely independent of other parts. In this case the spatial structure between parts is not exploited at all, leading to the inability of recovering individual parts as soon as something goes wrong. A general guideline might be to give the parts as much flexibility as necessary, but not more. As we will see in the following sections, a common theme in part-based models is to distinguish *inlier* parts from *outlier* parts. This is the reason why part-based models can be much more robust against local challenges, such as partial occlusions, than holistic object models. In a similar fashion, local appearance variations can be handled effectively, either by treating changed parts as outliers or by updating the appropriate part model.

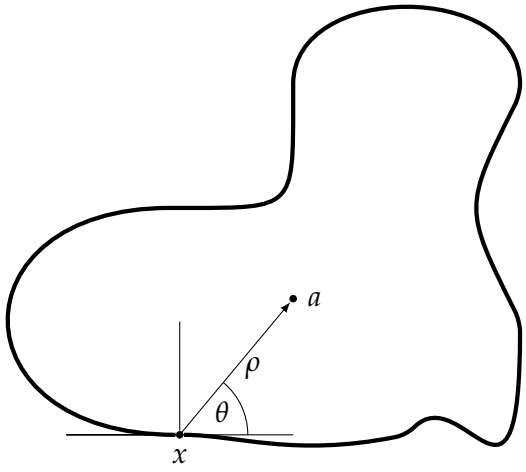
The question might arise why part-based models are not used ubiquitously in one-shot object tracking despite possessing these obvious advantages. The answer to this question lies in the difficulty of training part-based object models. This difficulty is not only present in object tracking, but also in object recognition, where for a long time holistic object models have outperformed part-based models (Felzenszwalb et al., 2010). Intuitively, the number of parameters (e.g. part location, appearance) in an object model increases with each additional model part. To properly estimate these parameters, a large amount of training data is needed to sample the parameter space exhaustively, which is not available in one-shot tracking. Still, some authors have undertaken the effort to explore part-based object models for one-shot object tracking, which we will discuss in the next sections together with prominent part-based models proposed in image recognition. The common denominator of these models is that they are composed of a number of individual parts which are represented in a common coordinate system. In the remainder of this chapter, we discuss object models that fit the following definition of a part-based model:

> *A part-based object model describes the spatial structure and interaction of individual model parts $x_1, \ldots, x_n$, where each part x refers to a 2D coordinate.*

Note that individual model parts might well carry much more information than just their location. For instance, each model part might be associated with information about scale or orientation. These aspects are however more dependent on the concrete part models that are in use and are not central to the discussion in this chapter.

3.3 Basic Part Models

In this section we discuss examples of basic part models that have been widely used in many different areas of computer vision. These methods address the problem of finding the parameters of a model given a noisy set of part correspondences. Noisy in this case refers both to small variations in the part locations itself, but also to gross outliers in the



(a) GHT

Input: Data X
Output: Model parameters W

$W = \emptyset$.
$\eta = \infty$.
for $i = 1 \ldots k$ **do**
  $X' =$ random subset of X
  $W' = F(X')$
  $X^+ = \{x_j \in X : L(W', x_j) < \theta\}$
  if $|X^+| > d$ **then**
    $W' = F(X^+)$
    $\eta' = L(W', X^+)$
    if $\eta' < \eta$ **then**
      $W = W'$
      $\eta = \eta'$
    end if
  end if
end for

(b) RANSAC

Figure 3.4: Methods for estimating model parameters. (a) The Generalized Hough Transform (Ballard, 1981). The curvature at the boundary of objects, the angle θ and the distance to the center ρ are stored in an R-table to enable voting into a parameter space. (b) In RANSAC (Fischler and Bolles, 1981), random subsets of the data are repeatedly used to find model parameters.

part correspondences. The basic models that are described in this section have received an extraordinary amount of attention by computer vision researchers, making them ideal candidates for pondering advantages and disadvantages of part-based models.

A classical approach in this category is the Hough Transform as proposed by Hough (1962). The principle idea of the Hough Transform is to perform a mapping of the image space into a parameter space, where evidence for model parameters is accumulated. In its simplest form, as described by Duda and Hart (1972), this principle is applied to line detection. Here, the input image is assumed to contain binary values indicating the presence or absence of edge pixels. These values typically are obtained by applying an edge detector on the input image. For each edge pixel, voting is then performed in the parameter space for all lines that could possibly go through this pixel. It is important to note that the parameter space is not continuous, but rather discrete. It is therefore necessary to manually set the size of the bins in which the votes are accumulated. While each pixel contributes many votes to parameter values that do not correspond to actual lines in the image, the accumulation of votes allows for retrieving the correct parameter values by searching for maxima in the accumulated votes. On a first glance, the Hough Transform does not fit the definition of a part-based model presented in the previous section as the spatial structure of individual model parts is not modeled directly. However,

there is in fact an analytical description of how the parts should be laid out, which is a much more reliable source of information than what can be learned from an example. However, the necessity to provide an analytical description restricts the Hough Transform to relatively simple shapes, such as lines or circles.

In an important paper by Ballard (1981), the Hough Transform is extended to the Generalized Hough Transform (GHT). In contrast to the Hough Transform the GHT allows for detecting objects of arbitrary shape where similar to one-shot tracking a template of the object is used. The principle idea is depicted in Figure 3.4a. In the GHT, a so-called R-table is created that relates the information from the edge pixels (the orientation) and a reference point on the object (typically the center). In the detection phase, whenever an edge pixel is encountered, the R-table is used to look up the voting direction. The evidence accumulation and the localization of the object works then similarly as in the Hough Transform. From today's point of view the GHT appears relatively primitive, as edge information is certainly not the most reliable feature for natural images. Still, the GHT exhibits all important properties of a part-based model.

One classical approach that has been widely used in computer vision in general when it comes to robustly estimating the parameters of a model is Random Sample Consensus (RANSAC). This technique, first described by Fischler and Bolles (1981), aims at estimating the model parameters W given noisy data X, where "model" in this case refers to an arbitrary statistical model. A detailed algorithmic outline of RANSAC is shown in Figure 3.4b. The main idea of RANSAC is to compute a minimal solution W' for the model using a random subset of the data X'. To this end, a fitting function F is used that might for example solve a system of equations to obtain the model parameters. As the data is assumed to contain outliers, this solution might or might not be a good overall estimate for the model parameters, depending on whether it was computed on inliers only. A criterion whether a solution is good or bad is based on the consensus of all samples including those that so far have not been used the model fitting. To this end, all samples in X are tested against a model-specific loss function L, yielding for each sample an error. The number of samples that *agree* is then an indicator for the general applicability of the model. This overall process is repeated k times and the best solution so far as measured by the number of inliers that support it is returned as a result.

RANSAC has been used extensively in the computer vision literature to compute transformations between noisy point correspondences both in 2D image space as well as in 3D space. These point correspondences are obtained using local features as presented in Section 2.2.3. Typically, the model parameters W are used to contain the transformation parameters in matrix form, where depending on the desired transformation certain parameters can be restricted to for instance allow for similarity transformations[11] only. The data X contains the point pairs and their respective image locations and a classical choice for the loss L is the squared reprojection error. Transformation estimation using

[11] A similarity transformation is restricted to translation, scaling and rotation and does not consider general affine motion, which includes shearing.

RANSAC is known to work very effectively. It is however based on the very strong assumption that objects have to be *rigid*. As soon as this assumption is violated, a direct application of this concept is no longer possible.

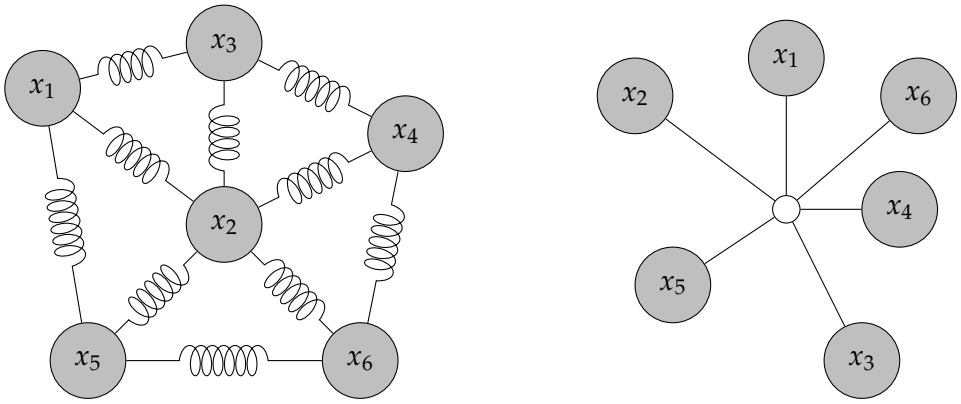
While it is worthwile to consider basic models such as the GHT or RANSAC as a source for inspiration for one-shot tracking algorithms, these models fail quickly as soon as the object of interest exhibits some form of deformation. In the case of the GHT, deformations lead to severe problems, as the edge information at the boundary of objects is no longer reliable as soon as the object for instance rotates slightly out of plane. On the other hand, the strong rigidity assumptions in RANSAC render this approach inapplicable for non-rigid objects. Numerous extensions have been proposed to these basic methods, of which we will discuss some in the following sections.

3.4 Constellation models

Constellation models treat individual parts jointly. Usually, this is done in an optimization framework, where deformation of individual parts is traded off with an appearance-based similarity measure. The idea of constellation models goes back to Fischler and Elschlager (1973), who presented it as a framework for pictorial structures. A typical layout of a constellation model is shown in Figure 3.5a where the springs between the model parts denote a certain flexibility. The principle idea behind constellation models is that each part places constraints on the location of other other parts. For instance, the torso of a person is usually in-between the head and the legs and any deviation from the expected position will incur a cost to this specific hypothesis, as the springs in the constellation have to be expanded or contracted.

A probabilistic constellation model is proposed by Fergus et al. (2003), who model appearance, scale, shape and occlusion of the object parts as Gaussian distributions. Parts on each image are identified using an interest point detector. The key idea is to learn the parameters of the distributions using Expectation-Maximization, thus establishing the probabilistic position of the parts and their appearance. For classification, a likelihood-ratio test is performed against the hypothesis that the object class is not present in the image. As the shape is modeled using a full covariance matrix, the number of parameters in the model increases quickly as more parts are added, effectively limiting the number of parts to about six.

As an example of a constellation model used in object tracking, we consider the work of Schindler and Dellaert (2007), who investigate the problem of bee tracking in a hive. Schindler and Dellaert employ a part-based approach, where each bee is modeled by three parts in a particle filtering framework. Each model part has a distinct appearance model and there is one global shape model representing the configuration of parts. Even in this relatively simple setup, it is necessary to *learn* plausible part constellations in a training phase. One might argue that this is only neccessary because in the problem investigated by Schindler and Dellaert, many almost identical objects and parts are



(a) Constellation model (b) Star-shaped model

Figure 3.5: Different part layouts. (a) In Constellation models, pairwise spatial constraints between individual parts $x_1, \ldots, x_n$ are modeled, here denoted as springs. (b) In a star-shaped model, all parts are positioned with respect to an anchor point, shown in white.

present so that the parts constellation plays a crucial role in distinguishing individual objects. However, also in one-shot scenarios multiple parts with similar appearance might occur as well. Authors have investigated to learn the constellation of parts in an online manner, as in Kwon and Lee (2009) and Čehovin et al. (2011). As the dimension of the learning problem increases with each model part, many training examples are necessary to exhaustively sample the space of possible configurations. Furthermore, as already mentioned multiple times, there are no hard class labels available, making the training process fragile in the first place.

Instead of learning the part constellation, L. Zhang and Maaten (2013) make use of an extension to the pictorial structures framework by Felzenszwalb and Huttenlocher (2005) for the tracking of multiple objects, but also perform experiments on multiple parts of a single object. Instead of a plausibility scores for the whole part constellation, for each pair of parts a constraint of the form

$$c_{i,j} = \|(x_i^t - x_j^t) - (x_i^1 - x_j^1)\|^2 \tag{3.1}$$

is added to a global optimization problem. The approach of L. Zhang and Maaten (2013) is especially effective if many similar objects are present. In this case, the spatial location between objects might be even more discriminative than the features that are extracted from individual parts. In case of severe deformations or dislocations the springs however can be more of a hindrance than a benefit, especially when parts are discriminative.

In summary, constellation models aim at placing certain constraints on how individual parts can be arranged. The information of what constellations are plausible has to be either learned in a training phase or set explicitly by some very general assumptions. Both of these methods are unsuited for one-shot object tracking, as no reliable training data is available and overly strong constraints placed on parts configurations might make

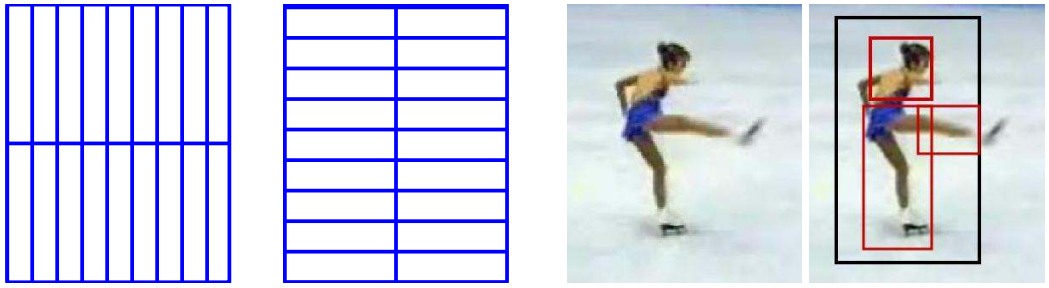
results even worse. Additionally, the number of constraints increase exponentially in the number of parts, if all pairwise constraints are considered, making inference difficult and expensive. Thus, the number of parts in constellation models is typically limited to few. As a consequence, star-shaped models have proven an interesting an alternative to constellations models, which we will discuss in the next section.

3.5 Star-Shaped Part Models

At a closer look, the GHT can be considered a special case of a more general family of models that has become known as *star-shaped* object models. Star-shaped models follow the setup shown in Figure 3.5b, where the position of each part is modeled with respect to one anchor point on the object of interest. The canonical choice for this point is the center of the object. The main advantage of star-shaped models over constellation models is that the inference is simple, as constraints exist only between the parts and anchor points, but not between individual parts. For example, Fergus et al. (2005) are able to use twice as many parts in their star-shaped model compared to their fully-connected constellation model. Star-shaped models have appeared in different forms, of which we will discuss some examples here.

In what can be considered a very basic form of a part-based model for object tracking, Adam et al. (2006) employ a star-shaped object model with a fixed configuration of parts. Adam et al. propose this model to address the loss of spatial information in color-based tracking, such as Comaniciu et al. (2000). To this end, parts are laid out in a rectangular grid, as shown on the left of Figure 3.6. This part-based model is employed in a sliding-window approach. Instead of computing color histograms over the whole subwindow, they are computed separately for each model part. Each part then contributes a certain score to the current subwindow as measured by a similarity function that compares the part in the current frame to the initial part histogram. It is interesting to note that in this work neither the part locations nor the templates are updated. Nejhum et al. (2008) expand on the idea of Adam et al. and suggest to employ histograms of adapted size and location to better capture the object's appearance, as shown in Figure 3.6 on the right. Essentially, the part locations and dimensions are determined in every frame by solving an optimization problem to cover as much of the object as possible using a fixed number of boxes. As additional information, a segmentation of the object is performed to separate it from background. Both Adam et al. and Nejhum et al. address the problem of finding part correspondences by evaluating the individual parts jointly in a *top-down* manner. In Adam et al. (2006) it is assumed implicitly that the part configuration does not change at all, disallowing any kind of deformations. In Nejhum et al. (2008) there is some form of adaption to deformation by adjusting the part positions in every frame. However, this concept is cumbersome as deformations are assumed to take place at a slow pace, which is not always the case in practice.

In contrast to top-down approaches, *bottom-up* techniques aim at providing each part



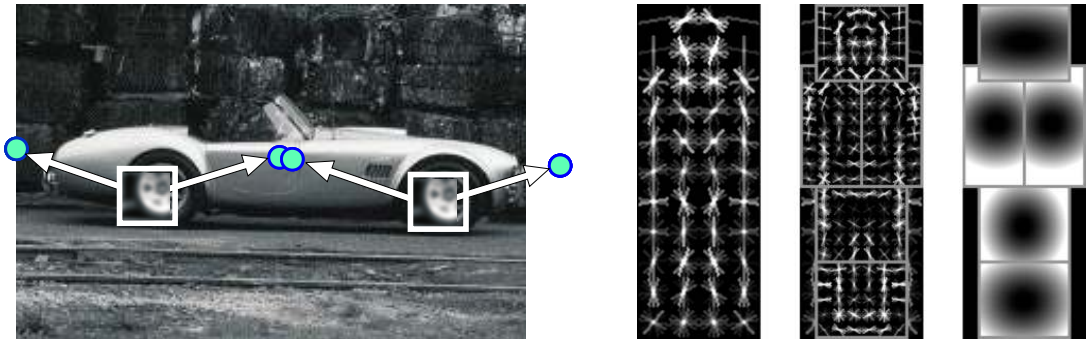
(a) FragTrack (b) Nejhum et al. (2008)

Figure 3.6: Star-shaped models used in top-down approaches, where in the localization step the part layout is fixed. (a) Adam et al. (2006) employ a non-changing layout. (b) Nejhum et al. (2008) adapt the part locations in the update step.

more flexibility. In the following, we discuss two approaches that are not directly related to object tracking, but made an impact in the object recognition community strong enough to make them worth discussing here. Leibe et al. (2008) propose a probabilistic extension to the GHT, in which evidence for object classes is accumulated. To this end, first a codebook of appearances is created using a clustering scheme based on information from interesting image regions in a training set. Using this codebook the Implicit Shape Model (ISM) is learned, that relates the locations of invididual parts in a probabilistic manner. For example, as shown in Figure 3.7a, the two tires of a car refer to the same entry in the codebook, but consist of two different entries in the ISM. In the inference step, evidence about object classes is accumulated by the votes of each part for the object center. In Figure 3.7a, plausible votes will form clusters at true object location, while implausible votes are cast spuriously and are discarded after a thresholding step. Gall et al. (2011) build on this approach by directly learning votes for image parts using what they call Hough Forests. Hough Forests are similar in spirit to Random Forests (Breiman, 2001), but instead of a class label they output multiple voting vectors targeting the expected center of the object.

Felzenszwalb et al. (2010) expand on the idea of using a global HOG description of objects by incorporating fine-grained information about parts into the object model. The global HOG descriptor acts as a root filter, whereas the parts are connected in a star-shaped model, as shown in Figure 3.7b. Part descriptions are computed on twice the resolution as the root filter. The score at a particular image location then consists of the score of the root filter, individual part scores minus a deformation cost for parts that deviate from their expected location. The learning framework consists of a latent Support Vector Machine to determine appropriate filter weights, the locations of the part and the deformation costs. The DPM has achieved for a long time excellent results on the PASCAL VOC challenge (Everingham et al., 2010) until the advent of end-to-end learning techniques such as Krizhevsky et al. (2012).

In summary, top-down star-shaped models are too rigid for one-shot tracking methods,



(a) ISM (b) DPM

Figure 3.7: (a) Implicit Shape Model (Leibe et al., 2008). (b) Deformable Part Model (Felzenszwalb et al., 2010).

as the part location is fixed during inference. Bottom-up star-shaped models have the clear advantages over constellation models that they are easier to train and also can handle part inference better, as outliers are implicitly accounted for. Clearly, the presented bottom-up approaches are not directly applicable to one-shot object tracking, as they rely on a large amount of labeled training data to be available. Next, we will discuss in detail a recent one-shot tracking approach based on a bottom-up star-shaped models in the framework of Hough Forests, where learning of part-based models takes place in an online manner.

3.6 HoughTrack

In their work called HoughTrack, Godec et al. (2011) propose a part-based representation where in each frame votes are cast for the object center by part correspondences. The outline of HoughTrack is shown in Figure 3.8. On the input image (top left) voting is performed in a sliding-window manner (top center) in order to localize the object of interest. The voting mechanism in HoughTrack is based on Hough Forests (Gall et al., 2011), where voting vectors for the object of interest are stored in the leaf nodes of the grown decision trees. This mechanism works well even in case of partial occlusions, as missing object parts do not interfere with the principle of looking for the cell with the highest number of votes. A more severe problem occurs however when the object of interest deforms. In this case, votes of individual parts no longer target the object center and the position estimation gets more and more diffuse up to a point where the object can no longer be identified. To deal with this case, Godec et al. propose an update mechanism that accounts for the deformation of the object of interest. To this end, after the voting step they back-project the votes in order to identify the parts supporting the hypothesis for the object center (top right image in Figure 3.8). These parts are then used to guide a GrabCut-based segmentation (Rother et al., 2004) in order to determine foreground and background parts (bottom right). The foreground parts as well as background parts are

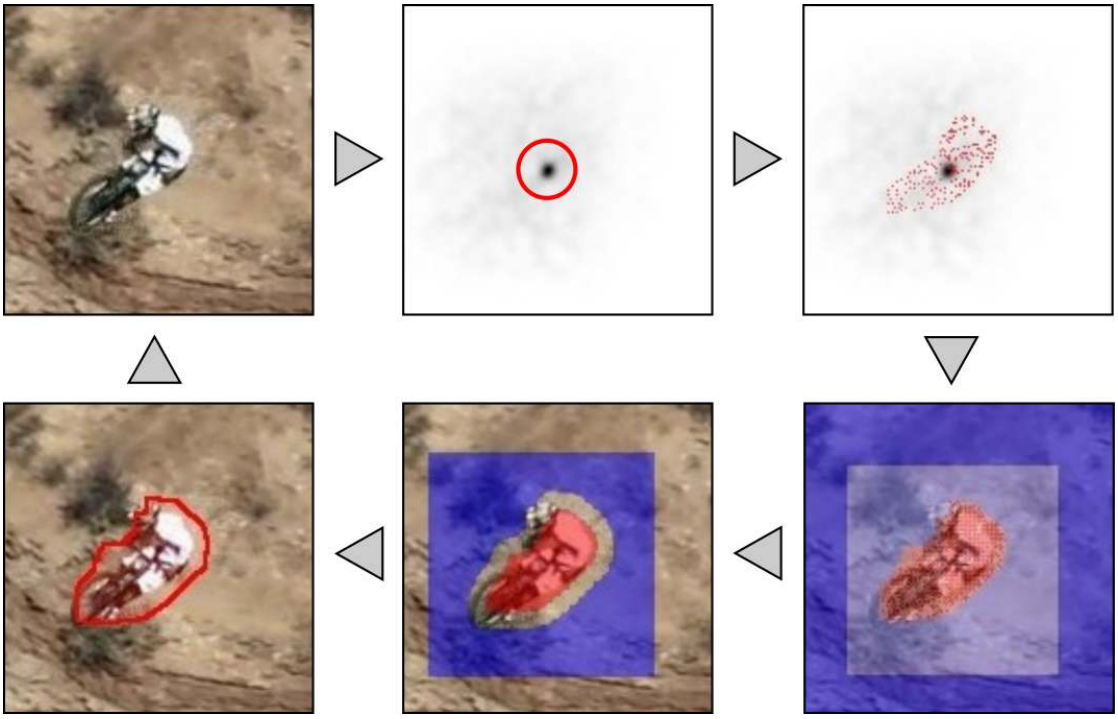


Figure 3.8: Outline of HoughTrack (Godec et al., 2011). Clockwise from top left: input image, localization by voting, support, segmentation, update, output.

then used to update the Hough Forest with new training examples (bottom center), where a small margin is added between positive and negative examples to avoid ambiguities. The segmentation is additionally used as algorithmic output (bottom left image), giving highly accurate results.

A vote for an image patch is obtained by computing the features of the patch and passing down the feature vector through the decision trees until a leaf node is reached. Godec et al. employ Lab-color space, first and second derivatives in x and y direction and a 9-bin histogram of gradients as features. A detailed illustration of the voting process is shown in Figure 3.9. As in the GHT, the votes (left) are discretized (center) into voting bins and accumulated. From the accumulated bins, the strongest votes are selected in order to reduce the number of irrelevant votes and cast onto the voting map (right). The cell containing the highest number of votes is then assumed to mark the object center. It is important to note that the bin sizes in Hough-Forest-based approaches are typically very small, as many votes are cast. This is possible due to the much richer feature extraction process compared to the GHT, where only edge information is used and bin sizes are typically broader. Due to the dense computation of votes, HoughTrack is relatively slow, a drawback which has been adressed by Duffner and Garcia (2013), who skip the feature computation step and instead use the raw image data as input for the Hough Forest.

The combination of Hough voting and segmentation in HoughTrack can be considered

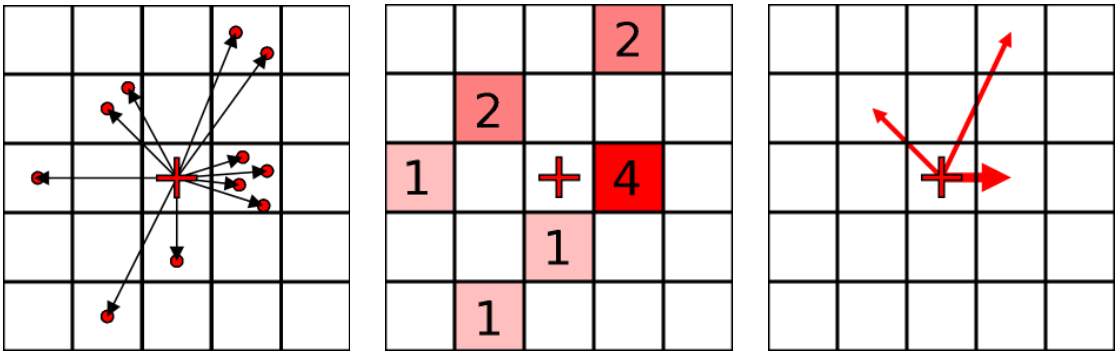


Figure 3.9: Bottom-up voting of Hough Forests in HoughTrack (Godec et al., 2011). Weak votes (left) are combined into strong votes before being cast. Votes are collected in an accumulator space.

a clear novelty in one-shot object tracking. It is a successful application of an on-line learning approach for a bottom-up star-shaped model. Essentially, the entire handling of deformation in HoughTrack takes place in the update process. This update process is however arguably a very fragile part of one-shot tracking algorithms. In the case of HoughTrack, the segmentation algorithm is not guaranteed to always yield the best possible segmentation of the object of interest. In fact, in case of clutter or partial occlusions a frequent failure of HoughTrack consists in incorporating unwanted objects into the object model, eventually leading to drift. This observation motivates the idea to move the handling of deformations from the update step to the localization step in the one-shot tracking pipeline, which is one of the main motivations of this thesis.

3.7 Conclusion

Part-based object models allow for dealing with object tracking challenges in a principled manner. This comes however with a more difficult learning scenario, as the parameters in the overall model increases. The discussion in this chapter motivates the need for a more principled approach to handling deformations in one-shot object tracking that should ideally have the following properties:

- As in one-shot tracking no reliable training data is available, the deformation of the object should be handled in the localization step instead of relying on possibly erroneous updates.
- A star-shaped model should clearly be preferred over a constellation model, due to their lower number of parameters and their ease in inference.
- The approach should not make any assumptions about the number and the nature of the object parts and should be applicable to a wide variety of part models.

- Inference should be done in a sparse way instead of having to deal with a possibly large accumulator space.

In the next chapter we will introduce the main contribution of this work, the Deformable Part Model for One-Shot Object Tracking, that aims to satisfy the above properties.

Chapter 4

Deformable Part Model for One-Shot Object Tracking

In this chapter, we introduce the Deformable Part Model for One-Shot Object Tracking (DPMOST). In a nutshell, the essence of the DPMOST can be expressed the following way:

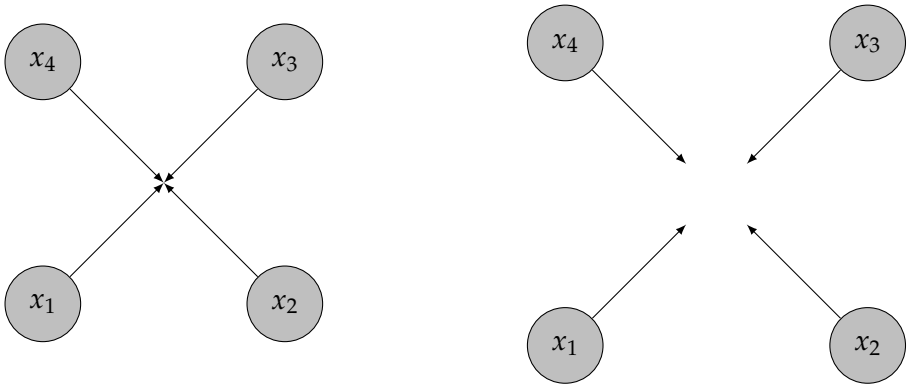
> *The DPMOST is a star-shaped part model extended by a threshold δ steering the allowed deformation of the object of interest.*

We first give some motivating examples in Section 4.1 that will help developing an intuition as to why the DPMOST can work. In Section 4.2 we formally describe the DPMOST model and discuss its properties. In Section 4.3 we provide another perspective on the DPMOST from the viewpoint of agglomerative clustering.

4.1 Motivation

To motivate the idea behind our proposed deformable part model, consider the star-shaped model composed of four parts in the left of Figure 4.1. Here, the star-shaped connections to the center here are represented as votes, indicated by an arrow. The parts are in a non-deformed state, leading to the convergence of all votes in the anchor point. On the right, the situation is depicted when a deformation is applied to the individual parts. Obviously, the votes no longer target the anchor point, but rather are spread around it. Note that the actual nature of the deformation itself does not matter and could be anything from articulated motion to perspective transformations. Depending on the amount of deformation, classical voting approaches such as the GHT can handle this deformation as long as it is smaller than the bin sizes in which the votes are accumulated. If the deformation gets larger, votes end up in different bins, making it very difficult to localize the object, as no sigificant maxima in the accumulator space might occur.

Instead of attempting to increase or shrink the bin sizes, a different idea is depicted in Figure 4.2, where each vote is associated with a *flexibility* as indicated by the gray balls of radius δ around the end of each vote. It is helpful to keep in mind that the level of flexibility itself does not arise from the part constellations itself, but rather is an independent fixed quantity. It is now crucial to understand that this flexibility does



(a) Object not deformed (b) Object deformed

Figure 4.1: (a) A star-shaped model composed of four parts. (b) The same model with a deformation applied to its individual parts.

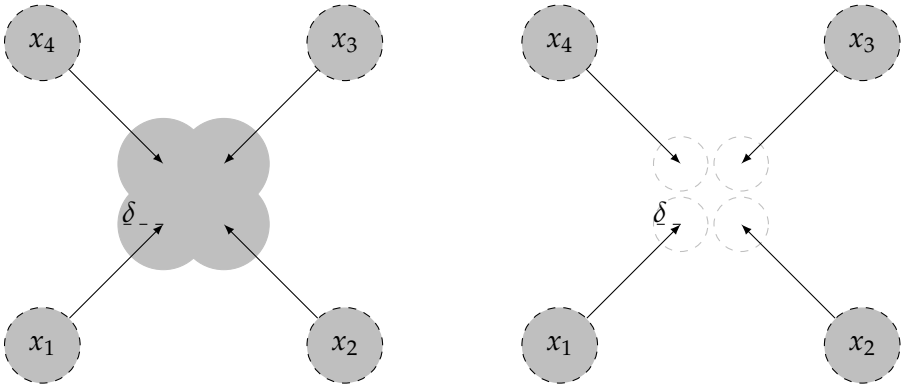
have an effect on other votes that are cast at the same time. The four parts depicted in Figure 4.2 on the left can be interpreted as *agreeing* on the object location within the limits of a given flexibility δ. On the right-hand side, exactly the same parts and votes are shown as on the left, but with a smaller flexibility. In this case, one would rather say that the parts *disagree* on the object location.

Let us consider two more examples. In Figure 4.3 on the left, again a deformation of the individual parts is depicted. This time however, the deformation is more extreme in a sense that some subsets of parts do no longer agree on the object location. For example, the parts x_4 and x_3 are widely separated with respect to the level of flexibility and do not agree on the object location. However, it is important to realize that in this case there are other *mediating* parts in-between that "connect" the parts x_4 and x_3. This property is central to our approach.

Recall from the discussion in the previous chapters that the association of individual parts is a difficult and error-prone task. It might well happen that the position of individual parts is estimated incorrectly. This situtation is depicted in Figure 4.3 on the right, where the part x_3 no longer connects to the rest of the parts. It could however well be that the position of the part was actually estimated correctly, but due to a deformation the part no longer connects. The radius of this flexibility therefore steers a trade-off between identifying deformed parts and rejecting wrong part correspondences.

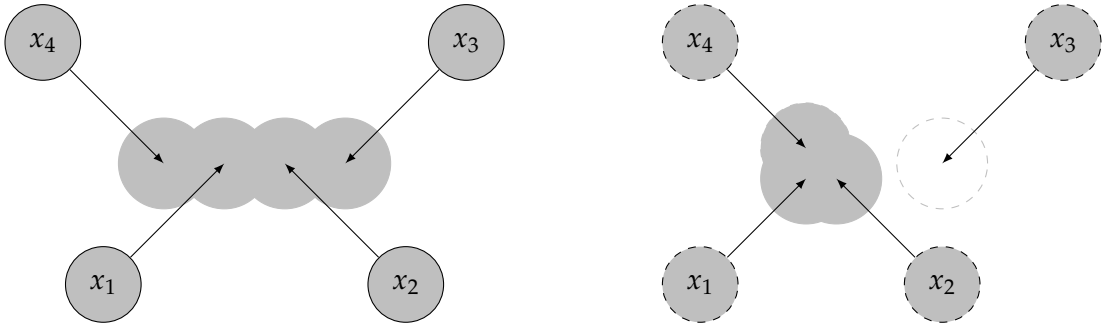
4.2 Definition and Properties

In this section we will formally introduce the DPMOST and discuss its properties. We begin with our proposed extension to the classical star-shaped model in Section 4.2.1 and continue with the discussion of the concept of transitive consensus that arises from this extension in Section 4.2.2. In Section 4.2.3 we show that the DPMOST can be made



(a) Parts in agreement (b) Parts in disagreement

Figure 4.2: By introducing a certain amount flexibility for each vote, parts can be thought of agreeing (a) or disagreeing (b) on the object location, depending on the radius of the allowed flexibility.



(a) Mediating parts (b) Outlier part

Figure 4.3: (a) Even during extreme deformations, remote parts (x_4 and x_3) might be connected by mediating parts (x_1 and x_2). (b) The part x_3 does not agree with the rest of the parts about the object location and is considered an outlier.

invariant to changes in scale and in-plane rotation. Finally, we show in Section 2.3 how the DPMOST can be used for localizing the object of interest.

4.2.1 Deformation in a Star-Shaped Model

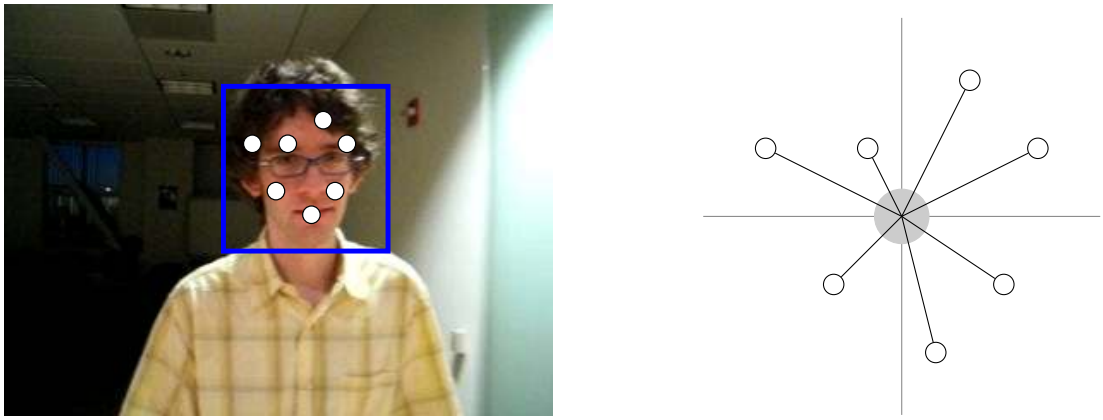
The DPMOST is based on a star-shaped model which in turn is composed of parts $\{x_1^1, \ldots, x_N^1\}$. Let us denote this reference configuration of parts as Z. Without loss of generality, the reference parts Z can be represented in a common mean-normalized coordinate system as shown in Figure 4.4 on the right, where the center of the object serves as the anchor point. This configuration represents a *known* configuration of parts that is assumed to be *correct*. In the context of one-shot object tracking, we define correct to mean based on certain information. Where Z comes from is not directly relevant to the discussion in this chapter, but the canonical way of establishing it is to look for interesting parts in the initial bounding box b_1 in the first frame of the video sequence, as depicted in Figure 4.4 on the left.

In the DPMOST, the connections from the individual parts to the anchor point can be interpreted as votes, as it is common in traditional star-shaped models. In addition, in the DPMOST a radius δ is associated with each vote that models the flexibility of the part. We refer to this radius hereafter as the *deformation threshold*. This deformation threshold is the essential ingredient to our proposed model and distinguishes it from other star-shaped models used in the tracking literature. In the DPMOST the deformation threshold is assumed to be the same for each object part. This is clearly in contrast to the intuition that some object parts are more deformable than others. For instance, arms and leg of a person exhibit large deformations, while its torso often remains more stable. However, maintining separate deformation thresholds for individual parts opens up the problem of determining the individual threshold values. As we aimed at avoiding learning as much as possible in this work, we did not investigate this topic further. It might however be a fruitful ground for future research.

The DPMOST can be used to make statements about part correspondences in the current frame. To this end, we will define formally what constitutes a part correspondence. A part correspondence m_i is a tuple (x_i^1, x_i^t), with x_i^t denoting the absolute image position of the part $x_i^1 \in Z$ in the current frame t. Hereafter, we will denote the set of all part correspondences in one single frame by

$$\mathcal{L} = \{m_1, \ldots, m_M\}\,. \tag{4.1}$$

Note that the number of correspondences M does not necessarily have to be equal to the number of reference parts N. It is often the case that some correspondences can not be established, leading to M being less than N. At the same time it can happen that there are two correspondences for the same part, for instance due to background clutter. Though rare, M can therefore in principle be greater than N. As the DPMOST is independent of the actual technique for establishing part correspondences, we defer the discussion of what correspondence techniques should be used to the subsequent chapter.



(a) Initial part candidates (b) Part coordinate system

Figure 4.4: (a): The configuration of individual parts is typically learned in a one-shot manner during initialization by an initial set of part candidates. (b) Reference parts are represented in a common mean-normalized coordinate system.

A part correspondence can cast a vote h, which is interpretable as a *hypothesis* for the center of the object. In this work, we will explore different ways of casting these votes. For keeping the discussion simple, we begin with a purely translational voting mechanism

$$h(m_i) = x_i^t - x_i^1 \tag{4.2}$$

that does not account for more complex transformations of the object. In Section 4.2.3 we will generalize this simple mechanism to account for the scale and the rotation of the object of interest. Note that we employ the *identity* of the part for selecting the correct voting vector instead of constructing R-tables as in the GHT or Hough Forests in HoughTrack. For now, we leave open how exactly this identity is established but return to this question in detail in Chapter 5.

4.2.2 Transitive Consensus

We now introduce the concept of transitive consensus that arises from the combination of votes and the deformation threshold δ. The deformation δ allows for stating whether two correspondences m_j and m_k in $\mathcal{L}$ are in *consensus* about the location of the object. We capture this event in the predicate ω and define it to be

$$\omega(m_j, m_k) = \begin{cases} \text{true} & \text{if } \|h(m_j) - h(m_k)\| < 2\delta \\ \text{false} & \text{Otherwise} \end{cases}. \tag{4.3}$$

This means that two correspondences are in consensus if the Euclidean distance between their votes is less than 2δ. The general assumption behind this consideration is that if many correspondences are in consensus about the object location, this is a very strong

indication that the correspondences are correct. This suggests that the predicate ω is in fact a transitive relationship

$$\omega(m_i, m_j) \wedge \omega(m_j, m_k) \implies \omega(m_i, m_k). \tag{4.4}$$

When this transitive property is applied, *mediating* parts are able to "bridge" the gap between remote parts. It is important to consider the role of the deformation threshold δ in this situation. This "bridging" is enabled by the deformation threshold in the first place. By increasing the value of δ, remote correspondences can be incorporated into the consensus of other correspondences. At the same time however it is not guaranteed that the remote correspondence is not in fact incorrect. An incorrect correspondence means that the identities of the parts x_i^1 and x_i^t are *not* the same. We refer to these incorrect correspondences collectively as *outliers*. Note that the vote of an outlier typically does not target the center of the object, but rather random image locations. When δ is increased, therefore not only correct remote correspondences are incorporated, but also outliers. For this reason, δ should not be set to an arbitrarily high value.

As in every transitive relation, the elements of $\mathcal{L}$ are partitioned into disjoint subsets $\mathcal{L}_1, \ldots, \mathcal{L}_K$ by the predicate ω. Depending on the image content that is being analyzed, the number of elements in these subsets can vary significantly:

- When the object appears in a deformed way within the limits of δ and few distractors are present, there will be one subset $\mathcal{L}^\omega$ with a large number of elements and potentially some subsets containing few outliers each.
- When the object deforms in a way that the deformation threshold between mediating parts is exceeded, $\mathcal{L}$ will be fragmented into subsets of medium size.
- When another object appears that shares the appearance of the object of interest, there will be two large subsets.
- In case of partial occlusions, the size of $\mathcal{L}^\omega$ will gradually shrink until only outlier subsets are left.

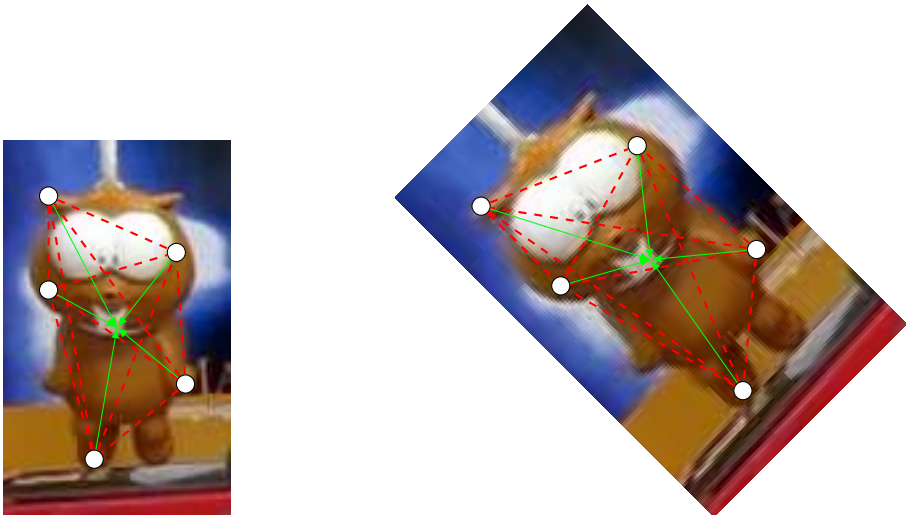
Based on these considerations, it is reasonable to define another predicate ϕ that expresses whether the correspondences in the largest subset $\mathcal{L}^\omega$ (which we will refer to as the consensus set) are reliable. While more complex predicates are imaginable, we define this predicate ϕ to be

$$\phi(\mathcal{L}^\omega) = \begin{cases} \text{true} & \text{if } |\mathcal{L}^\omega| \geq \theta_\phi \\ \text{false} & \text{Otherwise} \end{cases}, \tag{4.5}$$

where the threshold θ_ϕ is a parameter setting.

4.2.3 Invariance to Scaling and Rotation

As mentioned in Section 4.2.1, Equation 4.2 assumes implicitly that the object as a whole undergoes a translation only with respect to the reference part constellation Z. While



(a) Original Object (b) Transformed Object

Figure 4.5: To account for changes in scale and in-plane rotation, the votes (green) can be transformed accordingly by exploiting pairwise geometric properties (red).

slight changes in scale or in-plane rotation of the object can be handled by this simple variant of the DPMOST, it is desirable to account explicitly for these transformations, as they occur frequently and severely distort the spatial structure of the parts. Z can be thought of as representing the object of interest at scale $s_1 = 1$ and in-plane rotation $\alpha_1 = 0$. The goal therefore is to make the DPMOST invariant to the current scale s and in-plane rotation α. By doing so, Equation 4.2 becomes

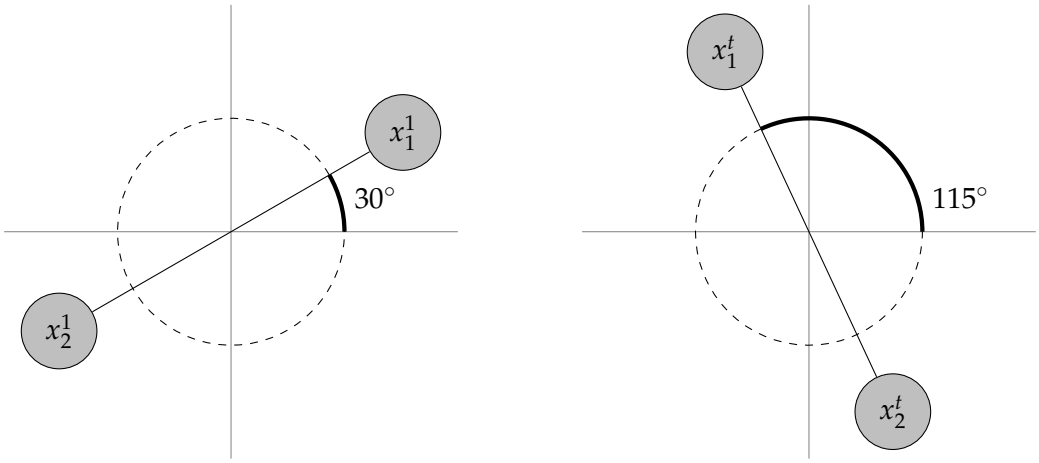
$$h(m_i) = x_i^t - s \cdot R x_i^1, \tag{4.6}$$

where R is the 2D rotation matrix

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}. \tag{4.7}$$

This change essentially rotates and scales the votes accordingly before they are being cast. In order to estimate the parameters s and α, one possibility to do so would be to cast multiple votes for each part correspondence and all possible transformation parameters, as it is done for instance in the GHT. This approach is however computationally extremely expensive. A different route can be taken by estimating the transformation parameters directly fom the part correspondences itself using robust statistical methods to account for possible outliers. For estimating s and α, pairwise geometric properties correspondences can be exploited as shown in Figure 4.5. An estimate for the scale s as proposed by Kalal et al. (2010) is

$$s = \text{median}\left(\left\{\frac{\|x_i^t - x_j^t\|}{\|x_i^1 - x_j^1\|}, i \neq j\right\}\right), \tag{4.8}$$



(a) Reference parts (b) Rotated part correspondences

Figure 4.6: The pairwise angular change between parts can be used to estimate the object rotation.

computing the pairwise difference in scale. The median is a robust statistic that selects from a set of numbers the middle number after sorting them. This way, extreme values on either end of the input distribution are ignored. As for the rotation, an estimate for α is obtained in a similar way by

$$\text{median}\left(\left\{\text{atan2}(x_i^1 - x_j^1) - \text{atan2}(x_i^t - x_j^t), i \neq j\right\}\right), \tag{4.9}$$

where atan2 computes the angle in the appropriate quadrant by means of the arctangent. An example for this is shown in Figure 4.6, where on the left the location of two parts in the reference configuration is depicted. The angle between a thought line between these parts and the x axis in this case is 30°. On the right, two parts are shown that correspond to the parts on the left. However, their geometry is different, leading to an angle of 115°. The change in rotation for this pair of parts therefore is $115° - 30° = 85°$.

An interesting question to ask is under what circumstances these heuristics can work. While the average is influenced heavily by outliers (in a statistical sense), the median is guaranteed to yield correct results as long as the number of inliers is greater than 50%. The same applies to the heuristics in Equation 4.8 and 4.9. It is also important to note that the correspondences themselves must be obtained in a manner that is invariant to scaling and rotation for the proposed heuristics to work.

4.2.4 Object Localization

A natural way of localizing the object of interest using the DPMOST is to compute the object center μ using the votes from inlier correspondences

$$\mu = \frac{1}{|\mathcal{L}^\omega|} \sum_{(x_i^1, x_i^t) \in \mathcal{L}^\omega} \left(x_i^t - sRx_i^1\right). \tag{4.10}$$

The mean is a reasonable choice here as the consensus set $\mathcal{L}^\omega$ is supposed to be free of outliers. Under the assumption that the object did not change its scale or in-plane rotation, a bounding box estimate b_t for the current frame can be obtained by placing the initial bounding b_1 over μ. When changes in scale or rotation occur, the combination of the object center μ, the estimated scale s and the in-plane rotation α can be interpreted as the parameters of a similarity transform

$$H = \begin{pmatrix} s\cos\alpha & -\sin\alpha & t_x \\ \sin\alpha & s\cos\alpha & t_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.11}$$

where t_x and t_y are the x and y coordinates of μ. This similarity transform can then be used to yield a rotated bounding box by first translating the initial bounding box so that its center is in the origin and then multiplying it to the right of H using homogenized coordinates[12].

Even though we stated in Chapter 1 to restrict the discussion to bounding boxes only, the DPMOST offers interesting ways of providing more fine-grained output. To go beyond bounding boxes, a couple of routes are imaginable. One option is to compute the convex hull of the correspondences in $\mathcal{L}^\omega$, which can interpreted as putting a "rubber band" around them. This way, some areas that would be recognized as foreground in a bounding box representation might instead remain background. However, if the object shape itself is highly non-convex, this approach will not improve the bounding box representation significantly. In this case, the output can instead be represented as small patches around part correspondences. However, depending on the technique that is used for establishing part correspondences, some areas might not be covered this way, such as homogeneous object regions.

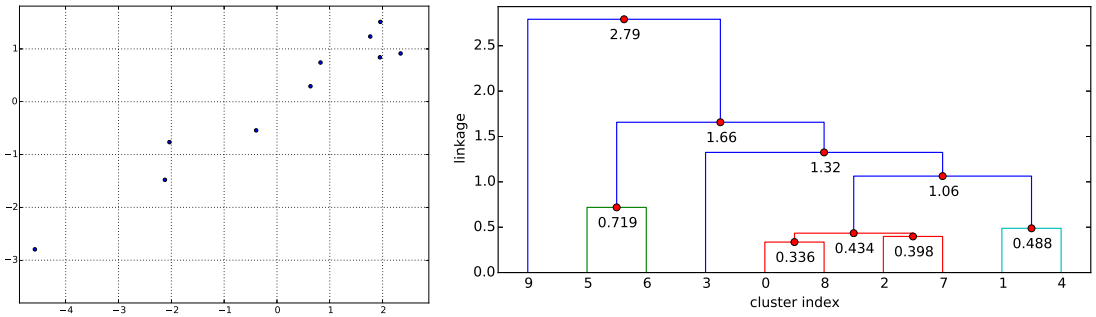
4.3 A Clustering Perspective

To understand from a more theoretical point of view why the DPMOST and the concept of transitive consensus can work, it is useful to view the process of finding connected correspondences as one of *clustering*. We will first give a brief introduction to clustering in Section 4.3.1. Next, we will discuss the approach of Cho et al. (2009) in Section 4.3.2, who employ clustering in the context of object recognition of deformable objects. In Section 4.3.3 we show how the DPMOST can be formulated in the framework of agglomerative clustering, and discuss the computational complexity of the DPMOST.

4.3.1 Overview about Clustering

Xu and Wunsch (2005) provide an exhaustive survey of clustering algorithms, on which we base our discussion in this section. In general, the aim of clustering is to partition

[12] The homogenized version of the vector $\begin{pmatrix} x & y \end{pmatrix}^\mathsf{T}$ is $\begin{pmatrix} x & y & 1 \end{pmatrix}^\mathsf{T}$.



(a) Data points (b) Dendrogram

Figure 4.7: Eample of hierarchical clustering. (a) Scatter plot of 10 data points stemming from a multivariate Gaussian distribution. (b) Dendrogram of the data. The linkage between two clusters indicates their distance. Here, single-linkage is used, meaning that the distance between two clusters is computed by the two closest datapoints of each cluster.

data into disjoint subsets. How this partitioning comes about and which parameters should be used depends on the exact goal of the application, the nature of the data and the subjective experience of the practicioner. Clustering is an unsupervised technique, meaning that no class labels for individual datapoints are necessary. Instead, clustering approaches aim at finding intrinsic structure in the data. This property already hints at the applicability in one-shot tracking scenarios, where no training data is available.

At the core of each clustering algorithm is the idea that two data points x_1 and x_2 can be compared using a *(dis-)similarity measure*. If x_1 and x_2 are similar, then they should end up in the same cluster and if they are dissimilar, they should be assigned different clusters. A very broad categorization of clustering algorithms can be performed by distinguishing between partitional clustering and hierarchical clustering. In partitional clustering, the main aim is to find an optimal partitioning of the data into K clusters, which can be seen as a combinatorial problem for which heuristics have to be used to maintain computability. On the other hand, hierarchical clustering algorithms aim at organizing the data in a tree structure, where the root node represents the whole data and the leaf nodes are the individual data points. We focus on this latter type of clustering in this section, an example of which is given in Figure 4.7, where a number of two-dimensional datapoints are represented in a dendrogram. Here, each datapoint starts out in an individual cluster numbered from 1 to 10. By going up the y axis (the linkage), clusters that are close-by are being merged. This way, the hierarchy between individual clusters is established.

This hierarchy can now be used to form flat clusters at will by employing a so-called cut-off threshold. In Figure 4.7, the cut-off threshold can be interpreted as drawing a horizontal line at a given amount of linkage. This line "cuts" the connection between clusters. A cut-off threshold of 0 implies that all datapoints end up in their own cluster. Vice-versa, a cut-off threshold of ∞ means that there will be exactly one cluster. This

type of clustering is also referred to as *agglomerative* clustering.

For the dissimilarity measure often distance metrics are being used. Popular choices are the Euclidean distance or the Manhattan distance. However, any function $D(x_1, x_2)$ can be used that satisfies the properties symmetry and positivity. The linkage between two clusters can then be computed in different ways:

- In *single-linkage*, the distance is determined by the two closest datapoints of two clusters.
- In *complete-linkage*, the distance between the two datapoint that exhibit the farthest distance is used.
- In *average-linkage*, the average pairwise distance between all points from each cluster is used.

More complex linkage variants are possible, as discussed in detail in Xu and Wunsch (2005). In summary, a hierarchical clustering algorithm is defined by its dissimilarity measure and its way of computing the linkage between clusters.

4.3.2 Recognition of Deformable Objects

Cho et al. (2009) propose an interesting approach for the recognition of deformable objects based on agglomerative clustering. In their approach, depicted in Figure 4.8, correspondences between objects are clustered based on two criteria:

- The dissimilarity between training descriptors and candidate descriptors.
- The geometric dissimilarity between two correspondences.

While the first criterion is a standard matching technique for SIFT descriptors, the geometric dissimilarity measure is more interesting. Using the notation from the previous section, it is defined as

$$D_{geo}(m_j, m_k) = \|x'_j - H_k x_j\| + \|x_j - H_k^{-1} x'_j\| + \tag{4.12}$$

$$\|x'_k - H_j x_k\| + \|x_k - H_j^{-1} x'_k\|, \tag{4.13}$$

where x'_i refers to the corresponding keypoint of x_i and H_i is a transformation matrix determined by the parameters of the keypoint detectors, which depending on the technique that is used can be estimated up to an affine transformation. The key idea here is that if there are two correspondences on an object, their keypoint parameters should undergo a similar transformation. Should the object itself undergo some deformation, the clustering approach is hoped to provide enough flexibility to recognize deformed correspondences as inliers while still identifying wrong correspondences as outliers. It has to be noted that the approach of Cho et al. relies heavily on the quality of the estimated parameters of the keypoint detectors. If they do not produce reliable results, one might be better off relying on appearance information exclusively.

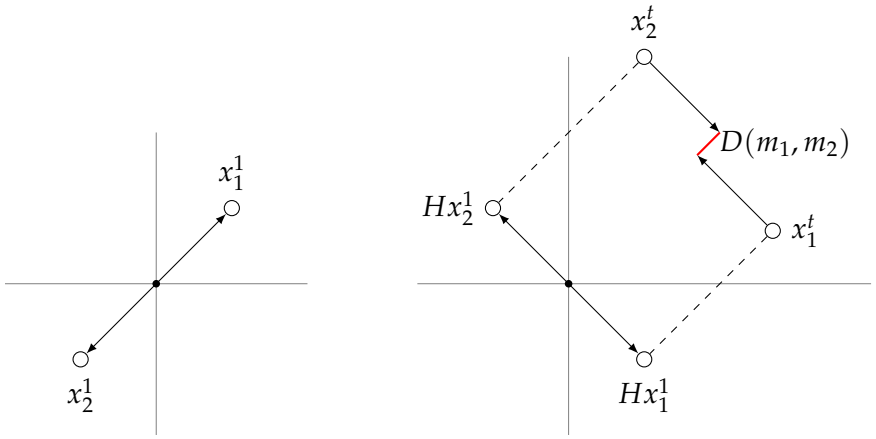
Figure 4.8: Cho et al. (2009) employ agglomerative clustering for the recognition of deformable objects. In their approach, the geometric deformation of objects is factored into the dissimilarity measure.

4.3.3 DPMOST and Agglomerative Clustering

As we will see in this section, the DPMOST can be formulated in an agglomerative clustering framework similar to the work of Cho et al. To appreciate this, consider Figure 4.9, where two part correspondences are depicted. On the left, their configuration in the star-shaped model is shown. The correspondences, which are shown on the right, underwent a clockwise rotation of 90° with respect to this configuration and were translated by slightly different amounts to the top and to the right. In this figure, the votes for each part correspondence are also depicted, which are defined by the estimated transformation matrix H that summarizes the transformations from Equation 4.6. Recall now that Equation 4.3 states that the Euclidean distance between two votes is required to be smaller than twice the deformation threshold. By ignoring the deformation threshold for a moment, we can interpret the distance between votes as a dissimilarity measure. In its written-out form, this is

$$D(m_j, m_k) = \left\| (x_j^t - Hx_j^1) - (x_k^t - Hx_k^1) \right\|. \tag{4.14}$$

In contrast to Cho et al. we compute the transformation matrix using the heuristics presented earlier, thus removing the dependence on parameter estimation by means of the part detectors. This is advantageous, as most of the local feature detectors presented



(a) Reference parts (b) Dissimilarity between correspondences

Figure 4.9: The DPMOST can be represented in a clustering framework by modeling the dissimilarity measure D between part correspondences as their distance between votes.

in Section 2.2.3 provide only a very rough information about the scale of the local feature. It is now crucial to understand that the deformation threshold δ presented earlier acts as the cut-off threshold in the framework of agglomerative clustering. This means that Equation 4.3 can be effectively computed by methods for hierarchical clustering.

By recognizing this, existing considerations about runtime complexity can be employed for the DPMOST. As noted by Xu and Wunsch (2005), most hierarchical clustering algorithms have an algorithmic complexity of at least $\mathcal{O}(M^2)$, where M is the number of data points to be clustered, which for the DPMOST are the number of part correspondences. This becomes apparent when considering that all pairwise distances between all part correspondences have to be computed to construct the hierarchy. The same argument applies to Equation 4.8 and Equation 4.9. It has to be noted however that in practical applications, the number of part correspondences M typically ranges in the order of 50-250, thus not posing severe computational problems. While it has not been investigated in this work, clustering algorithms targeted for large-scale databases might be used instead, such as BIRCH (T. Zhang et al., 1996). This clustering algorithm is able to partition large amounts of data in linear time.

4.4 Conclusion

In this chapter, the Deformable Part Model for One-Shot Object Tracking (DPMOST) has been introduced. It relies on the basic assumption that remote parts are connected by mediating parts and introduces a deformation threshold δ that specifies how large the deformation may become before parts are recognized as outliers. The most important property of the DPMOST is that it can handle deformations of the object without relying

on possibly erroneous training data. The only training information that is used to learn the initial parts layout Z stems from the only certain information in the problem formulation, namely of the initial bounding box. The DPMOST does not depend on a particular part representation and can handle an arbitrary number of parts. It can be made invariant to scaling and in-plane rotations by using robust heuristics based on the geometric layout of the part correspondences. As inference is performed in a sparse way, high processing speed can be obtained, even though the computation of the consensus is in $\mathcal{O}(M^2)$, where M is the number of part correspondences. The most distinguishing property from traditional star-shaped models, such as Fergus et al. (2005), is the transitive property of the part votes, effectively allowing for connecting remote parts by mediating parts. Also, there are much less parameters in the model, making it possible to estimate them in a one-shot manner. In the next chapter, we deal with the question what part representation should be used and how to establish part correspondences.

Chapter 5

Part Correspondences

In the previous chapter we have seen how the DPMOST can be used to deal with part correspondences on deformed objects in a principled way. The DPMOST is independent from the actual technique of establishing these correspondences. Nevertheless, it is clear that a particular choice of this technique has a strong impact on overall tracking performance. In this chapter, we will therefore discuss the topic of how to establish part correspondences at length. Note that an exhaustive comparison about each and every possible part model is out of the scope of this work. The discussion in this chapter is therefore limited to parts that can be matched using a descriptor of fixed size. In Section 5.1 we will discuss how candidate parts can be compared to reference parts by defining distance measures between their descriptors. Next, in Section 5.2 we discuss existing methods for how these distances can be used to associate the parts with each other by means of different matching strategies. Additionally, we introduce a novel strategy for disambiguating part correspondences based on the DPMOST. In Section 5.3 we then discuss the method of Lucas and Kanade as an example for optic flow estimation, which we view as another very different way of establishing correspondences. We introduce the idea of static-adaptive correspondences in Section 5.4 that aims at combining the advantages of robust, but inaccurate correspondences and accurate, but fragile correspondences. Finally, we will formulate in Section 5.5 the object tracking algorithm CMT that is based on the concepts DPMOST, static-adaptive correspondences and descriptor disambiguation. In this chapter, we re-use the notation for part correspondences from Equation 4.1. We stress the fact that this definition is applicable to all methods for establishing part correspondences that estimate the location x^t of a part x^1 in the current frame. Naturally, these methods will use much more information internally, such as information about the scale and the rotation of the part. For the DPMOST, the only output that is required is however the estimated location of the part.

5.1 Descriptors and Distance Measures

We will focus in this section on comparing the visual appearance of parts. A classical way of doing so is to compute the *descriptors* of reference and candidate parts. An overview about suitable features that can be used for this purpose was given in Section 2.2.3. Essentially, all these methods compute a d-dimensional descriptor c, thus mapping

image content into a feature space

$$c : I \rightarrow \mathcal{R}^d. \tag{5.1}$$

These descriptors have a couple of properties that are beneficial for employing them in a comparison. First, the dimension of the feature space is fixed, allowing for straightforward definitions of comparison functions on the feature space. In contrast, images of different size (i.e. different dimensions) must first be resized to a common size before a comparison can be performed. Second, tremendous efforts have been undertaken for devising descriptors that extract the *relevant* image information. There is no universal definition of relevance for descriptors, but obviously a descriptor that achieves good performance extracted more relevant information than a descriptor that performs poorly. It is however also clear that performance is dependent on the image content and some descriptors might perform better in certain situations than others. Third, descriptor extraction can be performed very fast, as already alluded to in Section 2.2.3.

With these considerations in mind, let us now ponder how two descriptors c^a and c^b can be compared. The most natural way of comparing two points in a space is to compute the Euclidean distance

$$L_2(c^a, c^b) = \|c^a - c^b\| = \sqrt{\sum_{i=1}^{d} \left|c_i^a - c_i^b\right|^2}, \tag{5.2}$$

also referred to as the L_2 norm of $c^a - c^b$. The L_2 norm was used in the original SIFT paper (Lowe, 2004) for computing distances between descriptors. However, depending on the assumptions that were made during the development of the descriptor, different distance measures might be more suitable than the L_2 norm.

It has to be noted that computing the L_2 norm is a relatively expensive operation, as the descriptors must be represented as floating point numbers according to Equation 5.1. When Equation 5.2 is computed on many elements, this might cause a considerable computational overhead even on modern computing hardware. Instead, clever researchers have developed so-called *binary* descriptors that are especially fast to compare in certain distance measures. These descriptors live in binary spaces

$$c : I \rightarrow (0, 1)^d. \tag{5.3}$$

This space can be intuitively thought of as a hypercube, where the elements of the space reside at the corners of the hypercube. In this space, the L_2 measure does not have a meaningful interpretation anymore. Instead, an immediately plausible distance measure is the number of edges that one has to wander from one corner on the cube to reach another corner of the cube. In the case of binary descriptors, this distance measure is the L_1 measure or equivalently, the Hamming distance

$$L_1(c^a, c^b) = \sum_{i=1}^{d} \left|c_i^a - c_i^b\right| = \sum_{i=1}^{d} XOR(c_i^a, c_i^b). \tag{5.4}$$

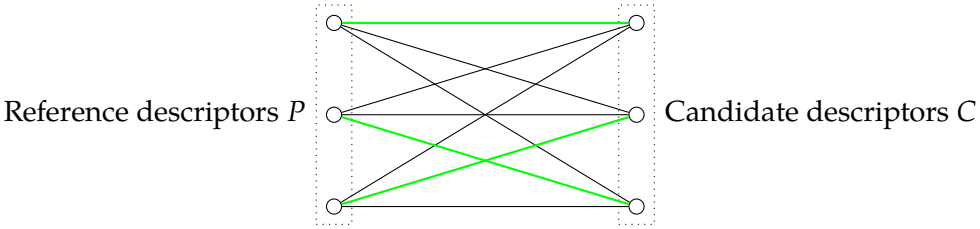


Figure 5.1: In the simple matching scenario depicted here, perfect matching results can be obtained by applying the nearest-neighbor rule.

It is now crucial to mention that modern CPU architectures provide dedicated support for computing the outcome of Equation 5.4. For instance, the SSE4.2 instruction set contains the *popcnt* instruction (Intel Corporation, 2007) that can be used for this purpose. Another advantage of employing binary descriptors lies in the fact that the L_1 distance between two binary descriptors is bounded by the dimension d. Without going into more details, the conclusion of this section is that the distance measure between descriptors should be selected with care. For the remainder of this chapter, we will abstract from the concrete distance measure that is used and refer to it as p.

5.2 Matching

The distance p between descriptors can now be used as the basis for establishing the association between descriptors from a reference set P and descriptors from a candidate set C. A very simple matching scenario is shown in Figure 5.1, where three reference descriptors are shown on the left and three candidate descriptors are shown on the right. The lines between those two sets represent distance comparisons, as discussed in the previous section. Let us for now assume that for each candidate descriptor a correct association is possible, meaning that for each candidate descriptor there is exactly one reference descriptor sharing the same identityLet us further assume that for the descriptors that belong together their distance is small and the distance to other descriptors from the opposite set is large[13]. In this case, an application of what is called the *nearest neighbor* (NN) rule leads to perfect matching results, as indicated by the green lines in Figure 5.1. In this rule, each candidate descriptor is associated with the reference descriptor to which the distance is shortest. This matching rule can be formalized by

$$NN(c) = \arg\min_{c_i \in P} p(c, c_i). \tag{5.5}$$

It should be clear by the number of assumptions required to achieve this perfect matching result that in practice the situation is much more complex. First of all, it is

[13] If this cannot be assumed, then the Hungarian algorithm (Kuhn and Yaw, 1955) can still provide a theoretically optimal solution.

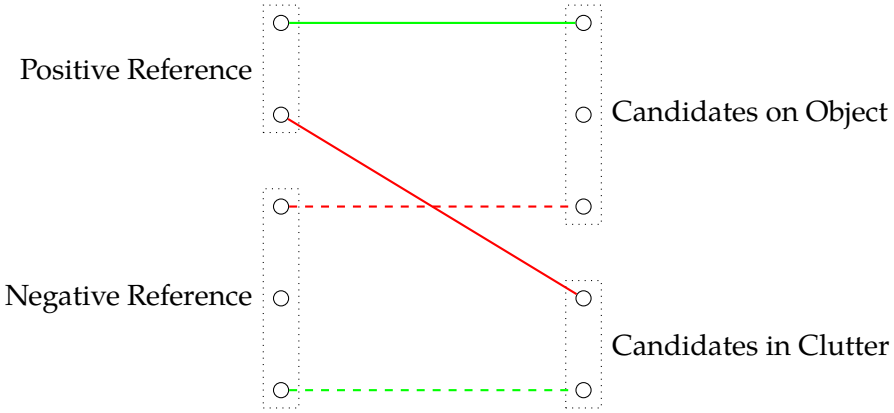


Figure 5.2: A more realistic matching scenario, where the reference descriptors have been divided into positives and negatives. Correct matches are shown in green, while incorrect matches are shown in red. The dashed lines denote matches to negative reference descriptors, which are discarded.

a rather unlikely situation that the number of reference descriptors and of candidate descriptors is the same. While in object recognition the number of reference descriptors in the database is much larger, in one-shot tracking one is often faced with the opposite situation that there is a small number of reference descriptors and a relatively large number of candidate descriptors in the current frame. From this consideration it is already clear that not all candidate descriptors can be matched to the reference set and a blind application of the NN rule will yield many wrong correspondences. It is therefore important to devise different matching rules that can overcome these situations. A simple measure to do so is to extend the NN-rule by a threshold θ_p.

$$NN^{\theta_p}(c) = \begin{cases} NN(c) & \text{if } p(c, NN(c)) < \theta_p \\ \emptyset & \text{otherwise} \end{cases} \tag{5.6}$$

This way, descriptors pairs that exceed a certain distance are not considered, leading to the situation that some candidate descriptors might not be matched at all.

When using the rule from Equation 5.6 wrong matches can still occur when by accident a "wrong" reference descriptor exhibits a shorter distance to a candidate descriptor than the actually correct reference descriptor or when a candidate descriptor from the background shares similar appearance with a foreground descriptor. To exclude these cases, the second nearest neighbor distance ratio (SNNDR) rule can be used that allows for a match only if the ratio between the nearest neighbour and the second nearest neighbor (NN_2) is smaller than a threshold γ.

$$SNNDR(c) = \begin{cases} NN(c) & \text{if } \frac{p(c,NN(c))}{p(c,NN_2(c))} < \gamma \\ \emptyset & \text{otherwise} \end{cases}. \tag{5.7}$$

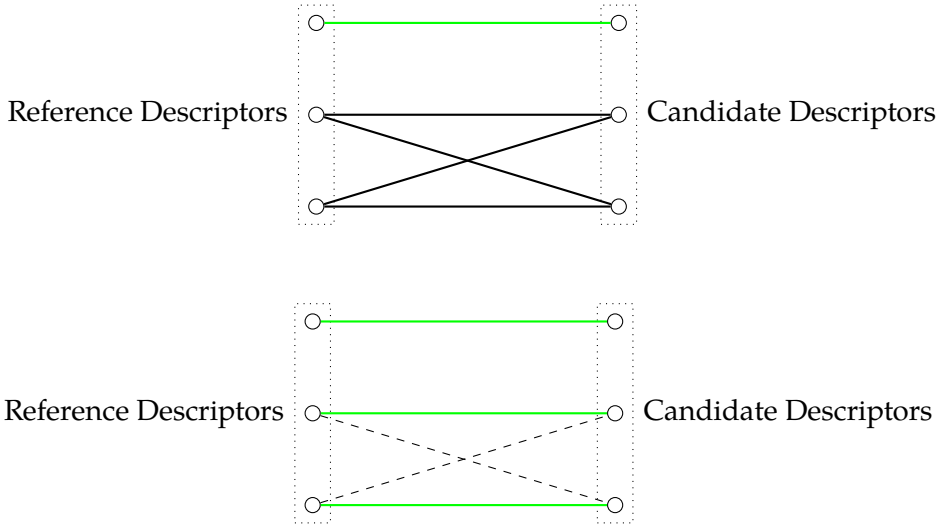


Figure 5.3: Top: The reference descriptors are ambiguous, leading to the inability of matching two of the three candidate descriptors. Bottom: By adding additional information, reference descriptors can be excluded, as indicated by the dashed lines. This leads to a disambiguation of the correspondences, improving matching results.

As an alternative to the SNNDR rule, backmatching can be performed. In this technique, matching is performed twice. In the second matching round, the roles of candidate and reference descriptors is swapped and only those correspondences are kept that yield the same association in both rounds.

It also makes sense to include *negative* descriptors into the reference set P and to discard all candidate descriptors that match to them. In the case of one-shot tracking, unambiguously negative descriptors can be found for instance outside of the initial bounding box. A more realistic matching situation is shown in Figure 5.2, depicting the positive reference set, the negative reference set as well as candidate descriptors on the object and clutter and the possible matching outcomes.

We now show how the DPMOST can be used to improve the matching result once the consensus set $\mathcal{L}^{\omega}$ is computed. To this end, consider the schematic matching illustration in Figure 5.3. In the top image, one correspondence could be established correctly. For the two other candidate descriptors however, two suitable reference descriptors exist according to their distance, as indicated by the black line. By following the SNNDR rule, none of these candidate descriptors can be matched, as another similar reference descriptor exists. The reference descriptors are *ambiguous*. However, if by some additional information one would be able to exclude one of the reference descriptors for each candidate, then the correspondences would be non-ambiguous. Recall that all discussion in this section has concerned only the *appearance* of the parts in the form of their descriptors. The DPMOST is able to provide additional information to the matching of descriptors in the form of the reference part configuration Z. The spatial information

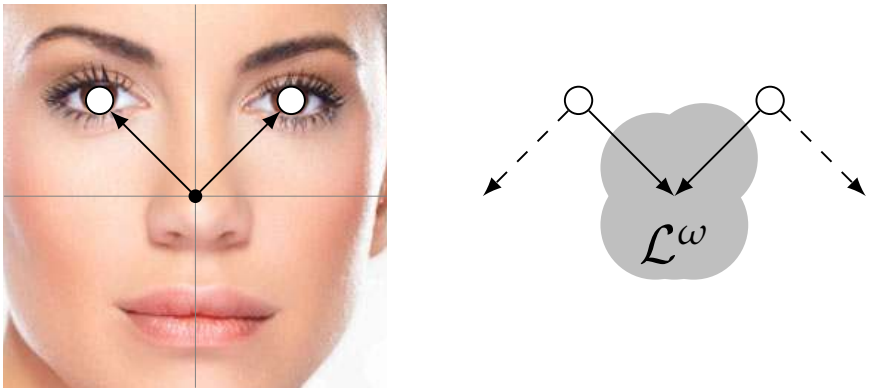


Figure 5.4: Left: Parts with similar descriptors are difficult to match based solely on their appearance. Right: We disambiguate these keypoints by excluding reference descriptors whose part correspondences can not be in consensus with $\mathcal{L}^{\omega}$.

in Z allows for *disambiguating* reference descriptors. As an example, consider the left image in Figure 5.4. Here, two ambiguous reference descriptors are shown in the form of the two eyes of a face. As these two parts of an object are virtually indistinguishable by their visual appearance, matching by means of the SNNDR rule will fail. On the right, two candidate descriptors are shown together with the consensus set $\mathcal{L}^{\omega}$, which has come about by other correspondences not shown in the image. Let us now assume that the parts of the candidate descriptors are located correctly and that the distance of the candidate descriptors to the reference descriptors of the eyes is reasonably low. By "simulating" a vote for each reference point (as indicated by the arrows), it becomes clear that one vote of each candidate descriptor is in consensus with $\mathcal{L}^{\omega}$, while the other one is not. Note that this way a potential deformation of the object of interest is respected. We suggest to follow exactly this procedure for all candidate keypoints that could not be matched during the SNNDR matching stage.

5.3 Optic Flow Estimation

A very different view on establishing part correspondences can be obtained when considering methods for estimating sparse optic flow. Here, the aim is not to find a proper assignment between reference and candidate descriptors, but rather to directly compute the *displacement* of individual parts from one frame to the next frame. As we will discuss in the next section, this approach is fundamentally different from descriptor matching. While there are different techniques for estimating optic flow, we will focus in an exemplary fashion here on the method proposed by Lucas and Kanade (1981) that we have already mentioned in Section 2.3.1.

The method of Lucas and Kanade, LK for short, rests on three assumptions. The first assumption referred to as *brightness constancy* states that a pixel at location x might

change its location in the second image I_t but retains its brightness value, formally

$$I_{t-1}(x) = I_t(x + v). \tag{5.8}$$

Here, v is the *displacement vector* of x. Clearly, this assumption alone is not particularly useful as there might be many pixels in I_t exhibiting the same brightness value as in $I_{t-1}(x)$. To alleviate this, a second assumption is made use of, typically referred to as *temporal persistence*. This assumption states that the displacement vector must be small. In the LK formulation, small means that $I_t(x)$ can be approximated by

$$I_t(x) \approx I_{t-1}(x) + I'_{t-1}(x)v, \tag{5.9}$$

where $I'_{t-1}(x)$ is the gradient of $I_{t-1}(x)$. Intuitively, this assumption establishes the connection between between the gradient, the image content and the displacement vector and is known as the *optic flow equation*. According to this equation, an estimate for v can be obtained by reformulating

$$v \approx \frac{I_t(x) - I_{t-1}(x)}{I'_{t-1}(x)}. \tag{5.10}$$

As images have two dimensions, Equation 5.10 is underdetermined for any pixel x and the solution space is a line instead of a point. To alleviate this, a third assumption called *spatial coherence* is introduced. It states that all pixels within a window W around a pixel should move coherently. The underdetermined single equation is thus turned into an overdetermined stack of equations which can be solved in a least-squared manner

$$\arg\min_v \sum_{x \in W} \left(I_t(x) - I_{t-1}(x) - I'_{t-1}(x)v\right)^2. \tag{5.11}$$

In Tomasi and Kanade (1991) a closed-form solution for this system is given. To further improve tracking results, some authors, e.g. Kalal et al. (2010), compute the optic flow not only in a "forward" direction, but also in a "backward" direction. If the backward estimate is close to $I(x)$ then the forward estimate is considered correct. A thorough dicussion about different methods for computing the optic flow is out of the scope of this work, for a survey see Barron et al. (1994).

5.4 Static-Adaptive Correspondences

In Sections 5.1 and 5.2 we have seen how correspondences between parts can be established by keeping a set of parts as a reference set and identifying potential part candidates in the current frame. An interesting question concerning the reference set is how to update this set so that it remains representative for the object of interest. Clearly, this consideration is subject to the same difficulties already presented in Section 2.4 and one can conclude that at best very conservative updates should be used to update the

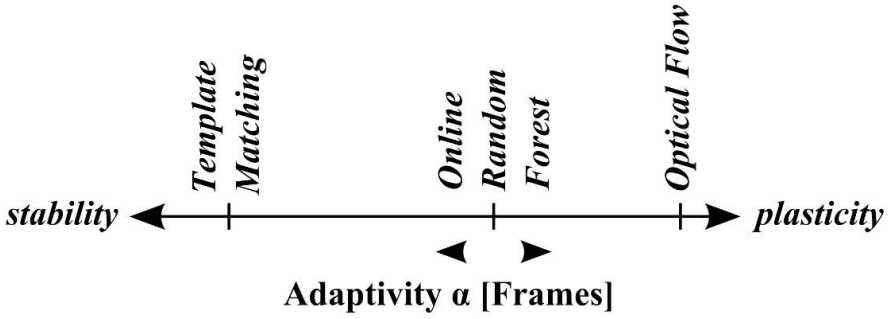


Figure 5.5: In PROST, multiple tracking algorithms from opposite ends of the adaptivity spectrum are used to achieve both robustness and accuracy.

reference set. It should also be noted that even though not immediately obvious the matching of descriptors fits exactly the tracking-by-detection paradigm discussed in Section 2.3.2, where the nearest-neighbor rule assumes the role of the discriminative classifier. Let us also keep in mind that apart from the descriptor disambiguation, the position of parts in previous frames does not play a role during descriptor matching. Due to the fact that a global search is performed for finding the best matches, a large number of frames between establishing the reference set and computing current descriptors does in principle not pose a problem, if the descriptor values remain similar. Descriptor matching can therefore be considered robust.

On the other hand, methods that employ a local search such as optic flow estimation techniques can be more accurate than a global search due to the restricted search space on which it is performed, as discussed in Section 5.3. This accuracy however comes at the cost of a loss in robustness. For instance, when the search space in the Lucas-Kanade method would be expanded to the whole image region and a global search would be performed, results would be rather poor as the error function that the Lucas-Kanade method aims to minimize would have many global optima. It is interesting to note that the disambiguation stage presented in the previous section is actually a local search with the aim of achieving more accurate results. Another crucial aspect when performing a local search is that the object model has to be updated more "quickly" than when searching globally, otherwise one might get stuck in the wrong local minimum.

The relationship between these two different kinds of establishing correspondences has to date been exploited by two one-shot tracking algorithms. In TLD (Kalal et al., 2012), which we have discussed at length in Section 3.1, a local search is performed to obtain training samples for the classifier performing a global search. In PROST (parallel robust online simple tracking), Santner et al. (2010) propose to employ multiple tracking techniques from opposite ends of the adaptivity spectrum, as shown in Figure 5.5. While there is no formal definition of what constitutes this spectrum, it is plausible that a template that is never changed is not adaptive, while an optic flow component whose internal template is updated in every frame is very adaptive. Santner et al. (2010) propose

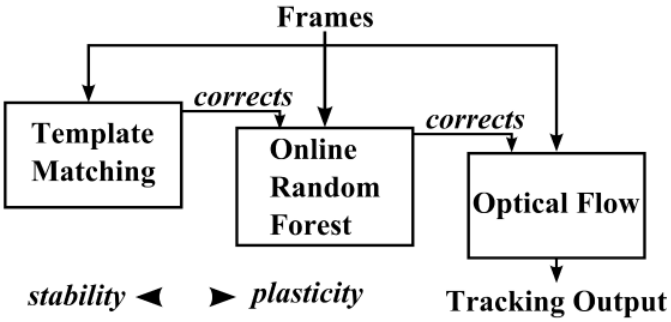


Figure 5.6: In PROST, more stable (i.e. less adaptive) tracking components overrule more plastic (i.e. more adaptive) tracking components.

to overrule the more adaptive components by the more robust ones, as shown in Figure 5.6. This is plausible as the adaptive components tend to have high short-term accuracy, but lack robustness.

With these considerations in mind, it makes perfect sense to transfer the concepts of Kalal et al. and Santner et al. that were developed for global object models to part-based object models. To this end, we suggest to treat the descriptors in the reference set as the non-adaptive and robust method of establishing correspondences. Similar to PROST, but different from TLD, we suggest to *never* update this reference set, as it was created with on the basis of very reliable information in the form of the initial bounding box. We refer to correspondences established by means of this reference set as *static correspondences* $\mathcal{L}^S$. On the other hand, we suggest to employ one *completely adaptive* component in the form of estimating the sparse optic flow, for example by using the method of Lucas and Kanade (1981). While optic-flow-based methods lack the required robustness for achieving good results in the long run, they can achieve much higher short-term accuracy than static correspondences. We refer to this type of correspondences as *adaptive correspondences* $\mathcal{L}^A$. As the optic flow is computed from frame I_{t-1} to I_t, one question is what image locations should be used as the starting point for the optic flow computation, apart from the trivial case of the first frame, where the starting points are identical to the reference set. Here, the DPMOST comes into play, as it provides a set of correspondences that is free of outliers, namely the consensus set $\mathcal{L}^\omega$. In the previous section we have introduced the idea of disambiguating reference descriptors to improve matching results with the outlook of achieving more correct correspondences. This idea becomes even more important when adaptive correspondences are used, as optic-flow-based correspondence methods require the location of the part in frame I_{t-1} as input. Another question that has to be answered is how to "fuse" the static and the adaptive correspondences, symbolically $\mathcal{L}^S \cup \mathcal{L}^A$, as each method provides one (or none in case of failure) hypothesis for the respective part location in frame I_t. For an answer to this question we follow the argumentation of Santner et al. and suggest to discard an adaptive correspondence when its static counterpart is available. This way, adaptive

Algorithm 2 CMT

Input: $I_1, \ldots, I_T, b_1$
Output: $b_2, \ldots, b_T$

$\mathcal{L}^1 \leftarrow$ detect_parts(I_1, b_1)
$Z \leftarrow$ normalize$(\mathcal{L}^1)$
$P \leftarrow$ compute_descriptors$(I_1, \mathcal{L}^1)$
$\mathcal{L}^- \leftarrow$ detect_parts$(I_1) \setminus \mathcal{L}^1$
$P \leftarrow P \cup$ compute_descriptors$(I_1, \mathcal{L}^-)$
for $t \leftarrow 2, \ldots, T$ **do**
  $C \leftarrow$ compute_descriptors$(I_t,$ detect_parts$(I_t))$
  $\mathcal{L}^S \leftarrow$ match(C, P)
  $\mathcal{L}^A \leftarrow$ optic_flow$(I_{t-1}, I_t, \mathcal{L}^{t-1})$
  $\mathcal{L}^* \leftarrow \mathcal{L}^S \cup \mathcal{L}^A$
  $s \leftarrow$ estimate_scale$(\mathcal{L}^*, P)$
  $\alpha \leftarrow$ estimate_rotation$(\mathcal{L}^*, P)$
  $\mathcal{L}^\omega \leftarrow$ transitive_consensus$(\mathcal{L}^*, Z, s, \alpha)$
  $\mathcal{L}^t \leftarrow$ disambiguate(C, P, Z)
  if $\phi(\mathcal{L}^\omega)$ **then**
    $\mu \leftarrow \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i^\omega$
    $H \leftarrow$ similarity_transform(μ, s, α)
    $b_t \leftarrow Hb_1$
  else
    $b_t \leftarrow \emptyset$.
  end if
end for

correspondences "survive" only as long as no suitable static correspondence exist.

5.5 Formulation of CMT

In this section we formulate the tracking algorithm CMT (Consensus-based Matching and Tracking). The main purpose of CMT is to evaluate the concepts introduced in the previous chapters in practice. We therefore do not employ any kind of image preprocessing that might skew the final results in any way in order to focus on the concepts that should be evaluated. During the formulation of CMT, we will frequently refer to Algorithm 2, where CMT is given in the form of pseudocode. A slightly simplified block diagram of this algorithm is shown in Figure 5.7. Additionally, Figure 5.8 depicts a visual outline of the individual processing steps in CMT on sample data.

To begin with, the first step in developing a tracking algorithm is to deal with the initialization phase. For setting up the DPMOST, this amounts to obtaining the reference configuration of parts Z. Many different ways of obtaining Z are imaginable, but arguably

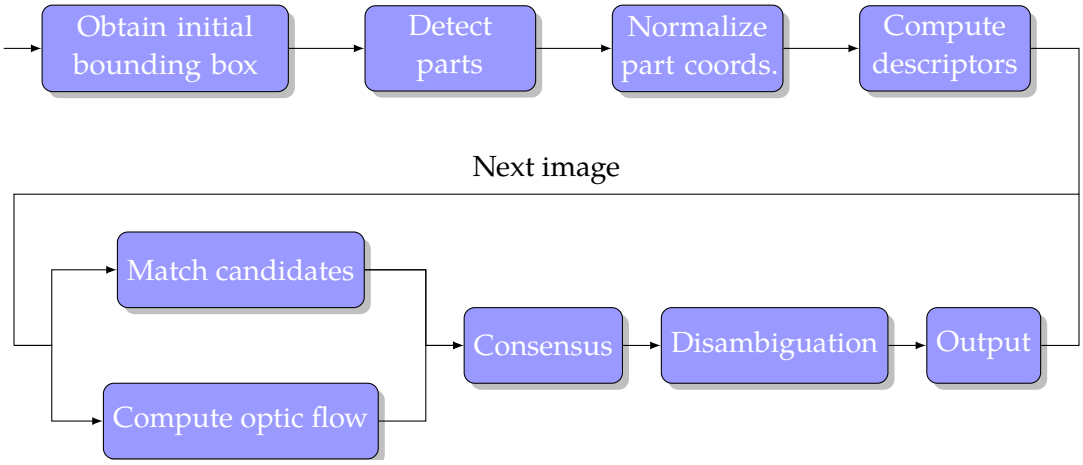


Figure 5.7: Block diagram of CMT.

the simplest one is to rely on existing part detectors. As many implementations for this task are available, we abstract from the concrete method used for this purpose and refer to it as an operation *detect_parts*, taking as input an image I and optionally a bounding box b, which can be used to specify a region of interest. For obtaining Z, we run *detect_parts* with the first image of the sequence I_1 and the inital bounding b_1 as arguments and consider the result of this operation as the first set of correspondences $\mathcal{L}^1$. By mean-normalizing $\mathcal{L}^1$ using the operation *normalize*, Z is obtained. $\mathcal{L}^1$ will also serve as the first basis for computing adaptive correspondences later. To complete initialization, we have yet to obtain the reference set of descriptors P that will be used to establish static correspondences throughout processing. To this end, we define another operation *compute_descriptors* that receives an image I and part locations as input. Again, this operation can be implemented by a multitude of different algorithms. We obtain P by running *compute_descriptors* on the input image and $\mathcal{L}^1$. In addition to the object parts, we also remember the background descriptors to improve matching results, as discussed in Section 5.2. To do so, we first retrieve all interesting parts outside of the bounding box b_1, compute their descriptors and add them to the reference descriptors P. The initialization phase is shown in Algorithm 2 in Lines 1-5.

For the rest of the sequence, we repeat the following procedure for each remaining image frame I_t. To establish static correspondences $\mathcal{L}^S$, we first obtain candidate descriptors C using the operations already described above. We then perform the association between candidate descriptors C and refererence descriptors P using the operation *match* that implements the SNNDR rule as discussed in Section 5.2. All statements for obtaining the static correspondences $\mathcal{L}^S$ can be found in Lines 7-8 of Algorithm 2. For computing the adaptive correspondences $\mathcal{L}^A$, we rely on the abstract operation *optic_flow* that computes the sparse optic flow between two images I_{t-1} and I_t for a set of point locations in the first image. Again, there are different methods that can implement this operation. We

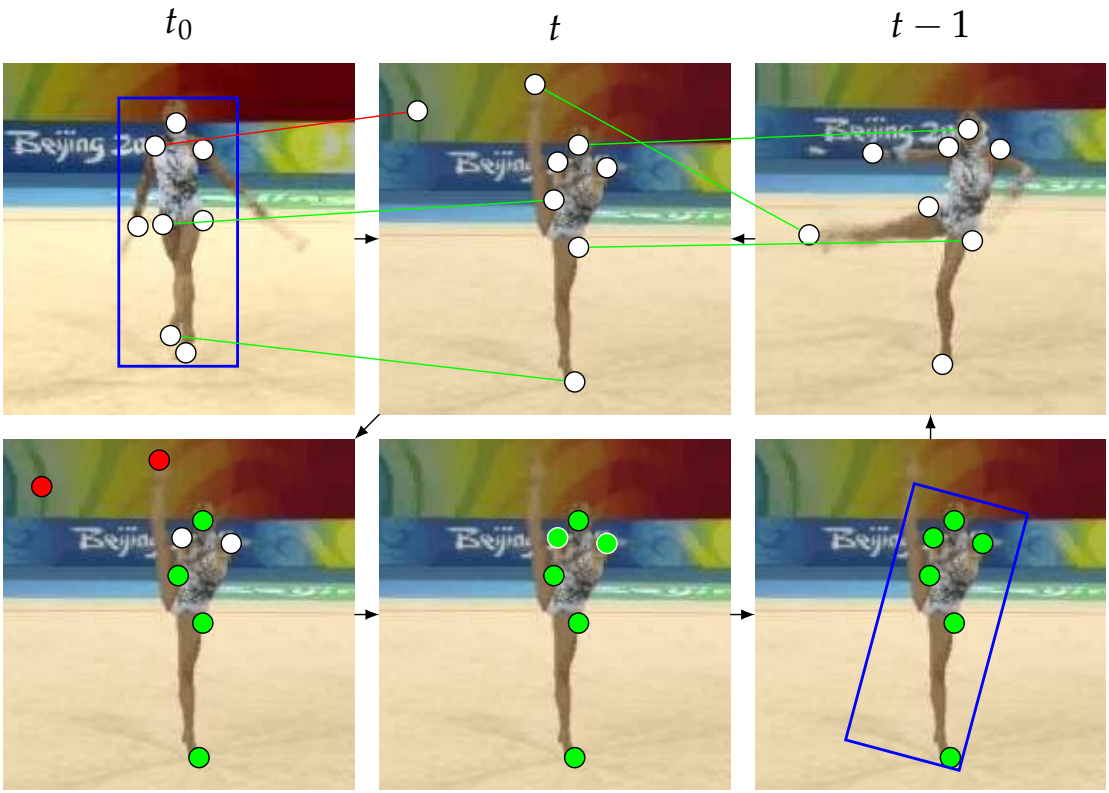


Figure 5.8: Outline of the CMT algorithm. Top row: From the initial bounding box in frame t_0 reference descriptors are extracted. In each frame t, these reference descriptors are matched to candidate descriptors to obtain static correspondences. Additionally, adaptive correspondences are obtained from frame $t-1$ by computing the optic flow. Bottom row: The DPMOST is used to obtain a partitioning of the correspondences into inliers and outliers (left). Next, ambiguous reference descriptors are disambiguated to strengthen the quality of the correspondences (center). Finally, a rotated bounding box is computed as the algorithmic output (right) and the tracking loop continues.

compute the optic flow on the part correspondences $\mathcal{L}_{t-1}$ from the previous frame. The computation of adaptive correspondences takes place in Line 9. In Line 10 we then combine the static and adaptive correspondences according to Section 5.4, giving preference to static ones, and denote this set as $\mathcal{L}^*$. To make use of the invariance of DPMOST to scale and rotation, we compute their estimates s and α in Line 11 and 12 using the operations *estimate_scale* and *estimate_rotation* according to the heuristics presented in Section 4.2.3. We then apply the concept of transitive consensus presented in Section 4.2.2 in Line 13 using the operation *transitive_consensus* to obtain the consensus set $\mathcal{L}^\omega$. To improve the subsequent estimation of adaptive correspondences for future frames, we disambiguate reference descriptors in Line 14 in a second matching round for all candidate descriptors

Operation	Library	Implementation
detect_parts()	OpenCV	FeatureDetector::detect()
compute_descriptors()	OpenCV	FeatureDescriptor::compute()
NN()	OpenCV	BFMatcher::knn()
optic_flow()	OpenCV	calcOpticalFlowPyrLK()
transitive_consensus()	fastcluster	MSP_linkage_core()

Table 5.9: Most of the operations in CMT can be implemented using the OpenCV library. For the *transitive_consensus* operation, the fastcluster library is used.

that could not be matched during the first matching operation.

Finally, to compute the algorithmic output of CMT, we employ the predicate ϕ in Line 15 to check whether the result is plausible according to the number of parts in $\mathcal{L}_t$. If ϕ is true, then we compute the center μ of the object by averaging the individual parts locations in $\mathcal{L}^\omega$ in Line 16 and compute the similarity transform H in Line 17 as discussed in Section 2.3. H is then used to obtain the output bounding box b_t in Line 18 by treating the coordinates of b_1 as homogenized column vectors. In case ϕ is false, we output an empty bounding box in Line 20.

For implementing the CMT algorithm, we make use of the OpenCV[14] library whenever possible. The OpenCV library provides interfaces both to low-level numerical computation operations, such as matrix multiplication as well as higher-level algorithms such different methods for computing descriptors. To this end, we implemented the operations *detect_parts*, *compute_descriptors*, *NN* and *optic_flow* using their corresponding function calls in OpenCV, as shown in Table 5.9. Additionally, we make use of a forward-backward check for the optic flow method, where we require that the backward result is within a radius δ of the original part. As discussed in Section 4.3.3, the *transitive_consensus* operation can be seen as the clustering of votes. For this purpose, we employ the fastcluster library (Müllner, 2013) that implements modern algorithms for computing agglomerative hierarchical clustering. We implemented our approach[15] in C++. Considering the free parameters that have to be set, CMT inherits the deformation threshold δ from the DPMOST and the distance threshold θ_p as well as the ratio threshold γ from the matching of descriptors for establishing static correspondences. Other parameters might be introduced depending on which implementations are used for the operations listed above, such as *compute_descriptors*. It has to be noted that the defomation threshold is measured in absolute pixel values, but can be easily set in a relative fashion for instance as a fraction of the diagonal of the input bounding box.

[14] Available at http://www.opencv.org.
[15] Available at http://www.gnebehay.com/cmt.

5.6 Conclusion

In this chapter, we have discussed what part representations are suitable for the DPMOST. On the one hand, we identify static part correspondences based on the matching of desriptors as one alternative. On the other hand, adaptive correspondences based on the estimation of the optic flow provide a second alternative. As these two alternatives stem from opposite ends of the adaptivity spectrum, we suggest to combine these two methods into static-adaptive correspondences. The DPMOST provides a suitable framework for handling these correspondences, as outliers are removed in each frame, making the computation of adaptive correspondences more reliable. Importantly, the static correspondences are never updated. Furthermore, by incorporating information from the reference part constellation Z, ambiguous descriptors can be disambiguated to improve correspondence results. This chapter concludes the algorithmic contributions of this work. We now turn to the question of how to properly compare the performance between different one-shot tracking methods in a fair and consistent manner.

Chapter 6

Evaluation

A perennial question in the object tracking literature is how to evaluate tracking approaches in a fair manner. In constrained tracking scenarios it is in principle possible to assess the performance of an object tracking algorithm in an automatic fashion where no manual work is required. This can be accomplished if the tracking scenario is constrained and the success or failure of the tracking process can be detected. In the case of one-shot tracking an automatic performance evaluation is however as daunting as creating a perfect tracking algorithm. Instead, by far the most common method to assess the performance of one-shot tracking algorithms is to compare the algorithmic output to *ground truth* data. We will discuss different ways of comparing algorithmic output to ground truth in Section 6.1, where the focus is on measures, and in Section 6.2, where the focus is on the overall evaluation protocol.

6.1 Measures

To compare different tracking algorithms, a dataset and appropriate measures are needed. As measures provide an indication of similarity between algorithmic output and ground truth, in Section 6.1.1 we will discuss how the ground truth data in datasets comes about. In Section 6.1.2 we will then investigate different ways of how to compare the ground truth data to algorithmic output on a per-frame basis. In Section 6.1.3 we will then explain how per-frame measures can be accumulated to provide concise information about tracking success. In Section 6.1.4 we will revisit a popular way of computing the overlap between groundtruth and algorithmic data. The traditional way of visualizing this resulting information typically has been to plot per-frame measures of individual sequences and to provide a tabular overview of accumulated measures. However, as tracking data sets become larger, authors have developed novel ways of providing an intuitive graphical representation of this information, referred to as success plots. These success plots will be discussed in Section 6.1.5.

6.1.1 Annotation Process

The ground truth data of a dataset should provide the ideal tracking output and is typically created by manually *annotating* individual video frames. It is not difficult to

imagine that this task is a tedious one. Again, while the perfect ground truth data consists of a pixel-wise segmentation of the object of interest, for reasons of practicability bounding boxes are used almost exclusively for creating ground truth data for one-shot object tracking sequences. The bounding box representation has two significant advantages over a pixel-wise segmentation.

- The number of parameters that define a bounding box is dramatically lower than a pixel-wise segmentation. A bounding box has four parameters (five in the case of rotated bounding boxes), while the number of parameters of a segmentation is equal to the number of pixels in an image.
- In the case of constant object motion, an estimate for the bounding boxes between two key frames I_i and I_j can be obtained by linear interpolation of the parameters of the two bounding boxes b_i and b_j, thus reducing the required effort for annotation.

However, the use of bounding boxes has the drawback that a canonical annotation for most objects does not exist. Consider for instance the example in Figure 6.1. As the object in this example is not rectangular, there is no bounding box that will exactly fit the shape of the object. There are in principle three different approaches of handling this situation.

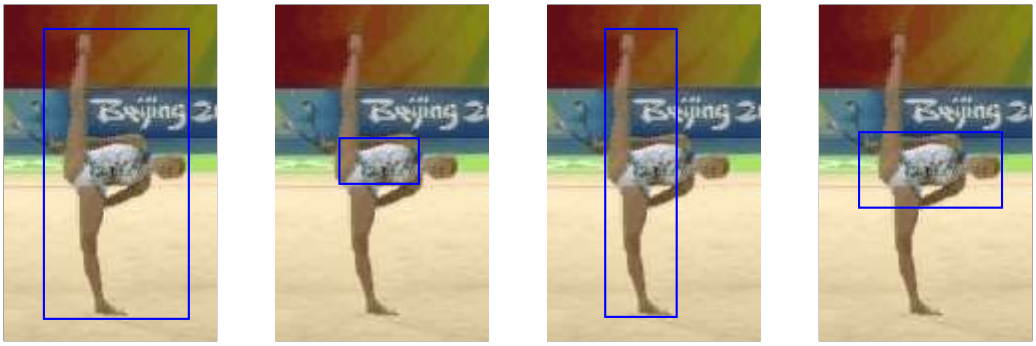
- The bounding box is chosen that completely contains the object of interest, as shown in Figure 6.1a.
- The bounding box is chosen that contains as much of the object as possible but no background, as shown in Figure 6.1b.
- A compromise between the two above is chosen, trading of the number of lost object pixels with the number of unwanted background pixels. Two examples for this are shown in Figures 6.1c. and 6.1d

In practice, the third way is usually chosen.

List et al. (2005) conduct an interesting study to assess the similarity of annotations performed by different annotators. List et al. conclude that most annotations are similar up to 80%. The above discussion demonstrates that bounding boxes offer an effective way of creating annotations but they should not be used for assessing the accuracy of tracking algorithms, but rather the robustness. To minimize bias in the evaluation protocol, it is important to consider the inherent ambiguity of ground truth data in the form of bounding boxes.

6.1.2 Per-Frame Measures

To compare the algorithmic output b_{ALG} to the ground data b_{GT} measures have to be defined that express the similarity between the two. As algorithmic output is performed



(a) Full (b) Minimal (c) Trade-off (d) Trade-off

Figure 6.1: Four different ways of annotating one and the same object. (a) The object is completely contained in the bounding box together with a high number of background pixels; (b) The bounding box covers only the object and no background at all. (c) The vertical axis of the object is seen as the principal axis for a trade-off. (d) The horizontal axis of the object is seen as the principal axis for a trade-off.

in every frame, these measures are referred to here as *per-frame* measures. Different measures capture different characteristics of the tracking algorithm.

A per-frame measure that has emerged very early in the tracking literature is the center error

$$center\_error = \|\mu(b_{GT}) - \mu(b_{GT})\|. \tag{6.1}$$

that captures the Euclidean distance between the centroid of the algorithmic output and the centroid of the ground truth. This measure makes sense if the actual distance between the estimated and the real location of the object is of great importance or if only the centroids are available as ground truth data. Traditionally, this measure has been applied for the tracking of points, for instance in radar data, where the amount of deviation matters significantly. The center error measure has also seen application in the evaluation of one-shot tracking methods (Babenko et al., 2009). However, this measure has considerable drawbacks:

- The dimensional information of the ground truth data is not used at all.
- The center error measure alone is difficult to interpret without additional information about the size of the object and the image, which might differ from sequence to sequence. For example, a center error measure that yields the value 50 might be a very good result if both the object and the image size are large. For small objects and images, this value might be a bad result.
- The center error measure is bounded only by the size of the image.

To overcome these shortcomings, instead measures have been developed that make use of the full dimensional information of b_{ALG} and b_{GT} (Hemery et al., 2007). The one

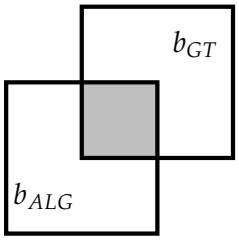


Figure 6.2: The Jaccard index as a measure for overlap between ground truth b_{GT} and algorithmic output b_{ALG}. The gray area corresponds to $b_{GT} \cap b_{ALG}$. The white area plus the gray area corresponds to $b_{GT} \cup b_{ALG}$.

single measure that is almost ubiquitously used for comparing two bounding boxes is the overlap measure

$$\psi = \frac{b_{GT} \cap b_{ALG}}{b_{GT} \cup b_{ALG}}, \tag{6.2}$$

that dates back to Jaccard (1912) and is also referred to in the remainder of this work as the *Jaccard index*. This measure has been prominently used in the PASCAL VOC Challenge (Everingham et al., 2010) for object detection and has seen widespread adoption in the computer vision as well as in the tracking literature. As shown in Figure 6.2, it measures the ratio between the correctly recognized part of the ground truth and the union between the ground truth and the algorithmic output. Compared to other overlap measures it has the following advantages:

- It is bounded between 0 and 1.
- It penalizes translation and scale changes in every direction roughly equally.

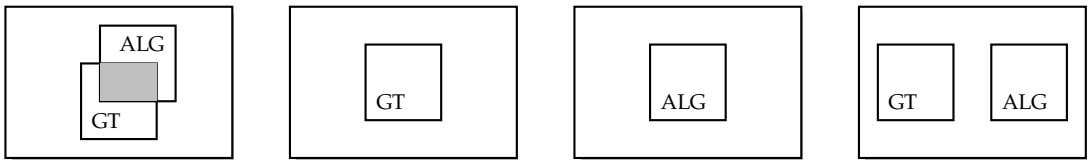
The Jaccard index is arguably popular both due to its obvious interpretability and its ease of implementation. While the Jaccard index has desirable theoretical properties, it cannot overcome the problem of ground truth ambiguity, as mentioned before. Many authors therefore compensate this ambiguity by introducing a threshold θ and thus convert the continuous overlap measure into a binary value. In the remainder of this section, we will discuss what new measures can be derived by imposing thresholds onto the overlap measure[16].

To this end we make use of the terms true positive (TP), false negative (FN), false positive (FP) and true negative (TN). These terms are borrowed from the terminology of binary classification, where the task is to distinguish an element as belonging either to the positive or negative class. In Table 6.3 the relationship between the class label and the predicted label is shown. In one-shot object tracking, the task is however slightly

[16] While this discussion deals solely with the overlap measure, it is immediately applicable to any other per-frame measure where high values indicate a desired result. It is also applicable to per-frame measures where low values indicate good results, such as the center error measure, by replacing all occurrences of $>$ with $\leq$ and vice-versa.

		Predicted Label	
		True	False
Class Label	True	TP	FN
	False	FP	TN

Table 6.3: Definition of true positives (TP), false negatives (FN), false positive (FP) and true negatives (FN) in binary classification.



(a) True positive (b) False negative (c) False positive (d) FN and FP

Figure 6.4: Conversion of per-frame measures into true positives, false positives and false negatives. The trivial case of true negatives is not shown here. Image adapted from Nebehay (2012).

different, as the aim is to predict the *location* of the positive class (the object). We will now see what adaptions have to be performed to these measures to adopt them for one-shot object tracking.

First, we derive the new measure TP (true positive). A true positive denotes the situation when the object of interest was correctly localized, as shown in Figure 6.4a. This is the case when the overlap measure is above the threshold θ. This situation can be expressed formally by

$$TP = \begin{cases} 1 & \text{if } \psi > \theta \\ 0 & \text{otherwise.} \end{cases} \tag{6.3}$$

Note here that the overlap measure ψ may not be defined if there is either no algorithmic output ($b_{ALG} = \emptyset$) or the object is not visible in the frame ($b_{GT} = \emptyset$).

If there is no algorithmic output and the object is visible (as in Figure 6.4b), or the measure is on the opposite side of the threshold (as in Figure 6.4d) then this is counted as a false negative

$$FN = \begin{cases} 1 & \text{if } (b_{GT} \neq \emptyset \wedge b_{ALG} = \emptyset) \vee \psi \leq \theta \\ 0 & \text{otherwise.} \end{cases} \tag{6.4}$$

The measures TP and FN are sufficient to analyze the ability of a tracking algorithm to correctly localize the object of interest. Still, it might be interesting from an application's point of view to analyze the behaviour of the tracking algorithm when the object is not visible. This is important when false alarms of an application are a problem. Imagine for instance an application where a costly action is performed as long as the object is visible.

$b_{GT} \neq \emptyset$	$b_{ALG} \neq \emptyset$	$\psi > \theta_\psi$	TP	FN	FP	TN
True	True	True	1	0	0	0
True	True	False	0	1	1	0
True	False	-	0	1	0	0
False	True	-	0	0	1	0
False	False	-	0	0	0	1

Table 6.5: Definition of true positives (TP), false negatives (FN), false positive (FP) and true negatives (FN) for one-shot object tracking.

If the algorithm localizes the object perfectly well, but still returns random results when the object disappears, the overall cost of the system increases. To this end, it makes sense to introduce two new measures FP (false positives) and TN (true negatives). A FP

$$FP = \begin{cases} 1 & \text{if } (b_{GT} = \emptyset \wedge b_{ALG} \neq \emptyset) \vee \psi \leq \theta \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

occurs if there is no ground truth data for the current frame ($b_{GT} = \emptyset$), but the algorithm still outputs a result, as shown in Figure 6.4c. It can also occur when the overlap between ground truth and algorithmic output is too small (Figure 6.4d). For completeness, we also provide the definition of true negatives TN, which occur when there is neither algorithmic output nor ground truth data for a frame, formally

$$TN = \begin{cases} 1 & \text{if } b_{GT} = \emptyset \wedge b_{ALG} = \emptyset \\ 0 & \text{otherwise.} \end{cases} \tag{6.6}$$

While this measure can be interesting, it is commonly not used for evaluating one-shot tracking approaches. A complete summary of these binary measures is provided in Table 6.5.

6.1.3 Accumulated Measures

In the previous section we have discussed methods of assessing the quality of the algorithmic output for a single frame. While it is certainly possible to visualize per-frame measures for comparing different tracking algorithms, it is often desirable to compute an *accumulated measure* on per-frame measures $f_1, \ldots, f_T$. In this section we will discuss accumulated measures that have been used in the literature.

The tracking length is one of the oldest accumulated measures, counting the number of frames up to the first failure. In our notation, a failure occurs when a false negative is detected. The tracking length measure can then formally be described as

$$tracking\_length = \min i, \text{ s.t. } FN_i = 1. \tag{6.7}$$

Whatever output comes after the first failure is ignored. One underlying idea behind this measure is that a manual re-initialization has to be performed after a tracking failure. The tracking length therefore measures the amount of time it takes until this failure occurs. However, in more complex scenarios one tracker might fail by chance in an early frame, distorting the results. In order to remedy this, the failure rate

$$failure\_rate = \sum_{i=1}^{n} FN(f_i) \tag{6.8}$$

has been used for one-shot tracking (Kristan et al., 2013), where a manual re-initialization is simulated after each tracking failure. It is very important to stress that both of these measures do not allow the tracker to correct its own errors in any way. In fact, employing these measures is useful for *short-term* trackers only in contrast to *long-term* trackers. While there is no immediately obvious conceptual difference between short-term trackers and long-term trackers, experience shows that in practice certain trackers are able to recover after a tracking failure, while others are not. This additional robustness comes however at the price of a possibly reduced short-term performance, as the tracker has to take care not to adapt too quickly to the object of interest. In summary, there is a certain trade-off between performance in short-term and long-term tracking. Both the tracking length and the failure rate are unsuitable for evaluating long-term tracking approaches because they ignore the intrinsic ability of long-term trackers to compensate errors and to re-detect the object of interest.

A natural way of formulating a long-term measure is to perform an average over all per-frame measures of a sequence

$$avg = \frac{1}{T} \sum_{i=1}^{T} f_i, \tag{6.9}$$

where care has to be taken that only those frames are considered where both the ground truth and the measure is defined. Again, this accumulated measure however creates the illusion of providing a very accurate result, while it is still based on imprecise annotations. To this end, authors (e.g. Kalal et al., 2012) have borrowed the notion of recall

$$recall = \frac{\sum TP_i}{\sum TP_i + \sum FN_i} \tag{6.10}$$

from the information retrieval literature. Recall measures tracking success in frames where the object is visible. It is interesting to note that if the object is visible in all frames of a sequence, the recall measure and the average of the TP measure are identical

$$recall^* = \frac{\sum TP_i}{\sum TP_i + \sum FN_i} = \frac{1}{T} \sum_{i=1}^{T} TP_i. \tag{6.11}$$

A complementary measure to recall is precision

$$precision = \frac{\sum TP_i}{\sum TP_i + \sum FP_i}, \tag{6.12}$$

measuring tracking success in frames where an algorithmic output exists. Recall and precision, originally stemming from information retrieval, can be interpreted using the notion of *relevance*. Precision measures how many elements from a retrieved set are relevant, while recall measures how many elements from the relevant population were retrieved. Clearly, there is a certain trade-off between these two measures that is steered by an internal threshold of the object tracking algorithm. Essentially, uncertain results can be suppressed using this threshold, resulting in less false positives, but usually also in a reduction in true positives. In contrast to binary classification, in one-shot tracking it is however not possible to increase the recall arbitrarily by lowering this threshold. This is due to the fact that the algorithmic output is a bounding box instead of a binary class label.

Researchers have looked at ways of combining recall and precision into a single measure. One way of doing so is to compute the *harmonic mean* of recall and precision

$$F = 2 \cdot \frac{recall \cdot precision}{recall + precision}. \tag{6.13}$$

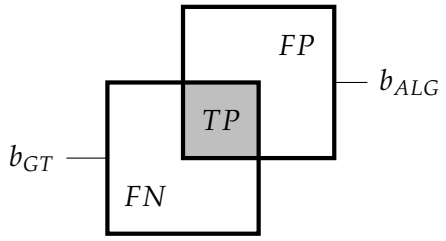
To appreciate why the harmonic mean is used for the F measure instead of the arithmetic mean, two intuitions are helpful. First, the idea of the F measure is that it should be high only when both recall and precision are high. For example if $recall = 0$ and $precision = 1$, then the arithmetic mean is 0.5, while the harmonic mean remains 0. The second intuition is that the arithmetic mean only makes sense if the numbers it is computed on have the same *scale*. For recall and precision, this is not the case, as their denominators are different. In fact, the harmonic mean is the reciprocal of the arithmetic mean of the reciprocals of the numbers it is computed on, eliminating these denominators. In summary, recall, precision and the F-measure are long-term tracking measures and should be used whenever the tracker is allowed to correct its own failures.

6.1.4 Overlap Measure Revisited

With the notation of accumulated measures defined from the previous section, let us again consider the overlap measure. A different way of interpreting the Jaccard index is to employ the binary measures from the previous section

$$\psi = \frac{TP}{TP + FP + FN} \tag{6.14}$$

as shown in Figure 6.6a. This time, the binary measures do not refer to per-frame measurements, but rather to individual pixels and are interpreted according to Table 6.3, where the class label of a pixel is true if it is contained within the b_{GT} and the predicted label of a pixel is true if it is contained within b_{ALG}. It is interesting to note that in the original paper, Jaccard (1912) employs the Jaccard index as a *similarity measure* between sample sets to describe the similarity of the flora in alpine regions. In this scenario, there is no notion of true or false. Transferred to the one-shot object tracking scenario, the



(a) Overlap measure revisited

TP	FN	FP	ψ	F
100	0	0	1.00	1.00
75	25	25	0.60	0.75
67	33	33	0.50	0.66
50	50	0	0.50	0.66
50	50	50	0.33	0.50
25	75	75	0.14	0.25
0	100	100	0.00	0.00

(b) ψ and F on sample values

Figure 6.6: (a) The overlap measure re-interpreted as a combination of true positive (TP), false negative (FN) and false positive pixels (FP). (b) Comparison of Jaccard index ψ and F measure for different values of TP, FP and FN, based on a hypothetical ground truth bounding box containing 100 pixels.

Jaccard index measures in the same way the similarity of the pixel distribution with respect to their classes (TP, FN and FP). There is however the striking difference that the two samples sets have very different origins. The ground truth is assumed to be true, while the algorithmic output is a form of prediction.

Let us now recall the formula for the F measure from Equation 6.13 and expand it using the notion of TP, FP and FN.

$$F = 2 \cdot \frac{recall \cdot precision}{recall + precision} = 2 \cdot \frac{\frac{TP}{TP+FN} \cdot \frac{TP}{TP+FP}}{\frac{TP}{TP+FN} + \frac{TP}{TP+FP}} = \tag{6.15}$$

$$= 2 \cdot \frac{\frac{TP \cdot TP}{(TP+FN)(TP+FP)}}{\frac{TP \cdot (TP+FP) + TP \cdot (TP+FN)}{(TP+FN)(TP+FP)}} = \tag{6.16}$$

$$= 2 \cdot \frac{TP \cdot TP}{(TP+FN)(TP+FP)} \cdot \frac{(TP+FN)(TP+FP)}{TP \cdot (TP+FP) + TP \cdot (TP+FN)} = \tag{6.17}$$

$$= 2 \cdot \frac{TP \cdot TP}{TP \cdot (TP+FP) + TP \cdot (TP+FN)} = \tag{6.18}$$

$$= \frac{2 \cdot TP}{2 \cdot TP + FN + FP} \tag{6.19}$$

The outcome of this calculation bears a striking similarity to Equation 6.14, the difference being that in the F measure the TP have more weight both in the numerator and in the denominator. It is easy to prove that the F measure always yields larger values than the

overlap measure ψ by substituting $a = TP$ and $b = FP + FN$.

$$\frac{2a}{2a+b} \geq \frac{a}{a+b} \tag{6.20}$$

$$\frac{2a(a+b)}{(2a+b)(a+b)} \geq \frac{a(2a+b)}{(2a+b)(a+b)} \tag{6.21}$$

$$2a(a+b) \geq a(2a+b) \tag{6.22}$$

$$2a^2 + 2ab \geq 2a^2 + ab \tag{6.23}$$

$$2ab \geq ab \tag{6.24}$$

$$2 \geq 1 \quad \square \tag{6.25}$$

Figure 6.6b visualizes in tabular form the difference between the F measure and ψ for a ground truth bounding box of 100 pixels. While it does not make sense to prove that one measure is more "correct" than the other, it can be argued that the F measure actually yields more intuitive results. Especially in the case when all TP, FN and FP are 50, it is difficult to argue why the result of the similarity measure should be below 0.5, as it is in the case of ψ. While for the comparison of different tracking algorithms it is arguably irrelevant whether the F measure or ψ is used, one should not forget that the overlap threshold θ_ψ has to be set manually. This threshold should have a plausible interpretation. Even though admittedly this discussion is somewhat subject to a personal understanding of aesthetics, we argue here that the F measure actually provides a more plausible interpretation than ψ. Nevertheless, to remain comparable to the experimental results of other authors, we employ the overlap measure ψ for conducting experiments.

6.1.5 Visualization of Measures

The classical way of visualizing and comparing tracking results is to plot per-frame measurements for individual sequences, as it is commonly done with the center error. When the number of sequences grows larger, a common route is to instead display accumulated measures over sequences in a tabular form. As tracking datasets tend to become larger, this approach is also no longer practicable. Recently, success plots have been used to overcome this problem[17]. An example for such a success plot is shown in Figure 6.7, where on the x axis a threshold θ is shown. On the y axis, the percentage (the success rate) is shown for which the measure in question f is within the threshold θ. Formalizing this[18], a success plot is a function[19]

$$S(\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{I}_{f_i > \theta}, \tag{6.26}$$

where $\mathcal{I}$ is the indicator function. A success plot can in principle be applied to any kind

[17] To avoid confusion, it is noted here that Babenko et al. (2011), Henriques et al. (2012), and Wu et al. (2013) refer to success plots of the center error measure as precision plots.

[18] Again, for measures where small values indicate a good result, the signs have to be reversed.

[19] In other scientific areas, this function is called survival function.

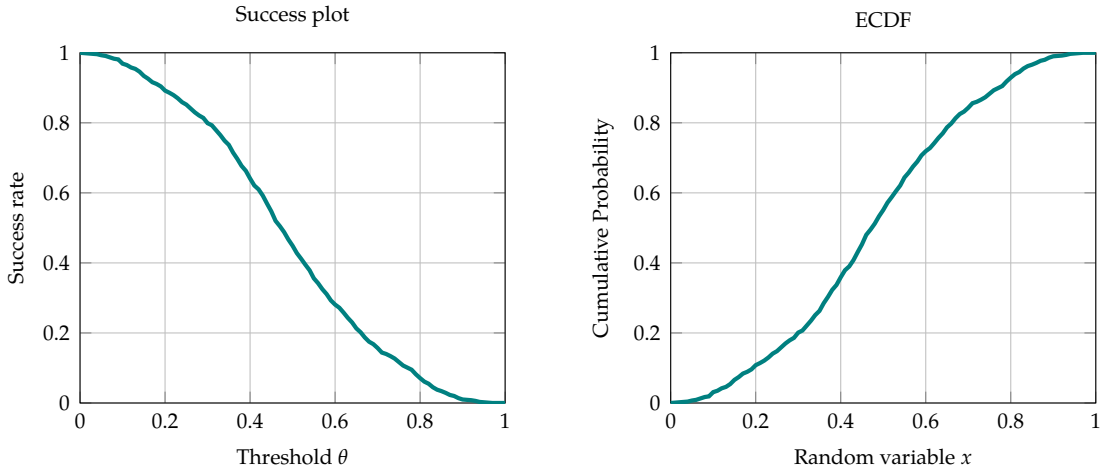


Figure 6.7: Equivalence of success plots and empirical cumulative distribution functions (ECDFs). Both plots were created from the same underlying data (N=1000), drawn from a triangular distribution with limits (0,1) and mode 0.5.

of measure, be it per-frame measures or accumulated measures. Still, care has to be taken not to hamper the interpretability of a success plot. For example, Wu et al. (2013) employ a success plot of the overlap measure ψ to visualize tracking performance on a dataset. Instead of providing one success plot per sequence, they concatenate the algorithmic output and the ground truth data of all sequences into a single giant sequence. This way however, it has to be considered that a long sequence has more impact onto the final result than a short sequence, which may not be desired. Furthermore, as the overlap measure is subject to ground truth ambiguity, the significance of the success plots with respect to overlap used by Wu et al. is questionable for a threshold > 0.8.

To understand how a success plot should be interpreted, a very helpful insight that we provide here is that success plots are related to empirical cumulative distribution functions (ECDF) by the identity

$$S(\theta) = 1 - ECDF(\theta), \tag{6.27}$$

where the ECDF of the random variables $x_1, \ldots, x_n$ is

$$ECDF(t) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{I}_{x_i \leq t}. \tag{6.28}$$

The equivalence is visualized in Figure 6.7, where the same data was used to create a success plot (left image) and an empirical cumulative distribution function (right image). By recognizing that a success plot is closely related to the ECDF, it also becomes clear that the *area under the curve* (AUC) of a success plot is nothing else than the mean of the individual measures.

$$AUC(S) = \frac{1}{n} \sum_{i}^{n} f_i = \int_0^\infty S(\theta)\, \mathrm{d}\theta. \tag{6.29}$$

6.2 Evaluation Protocols

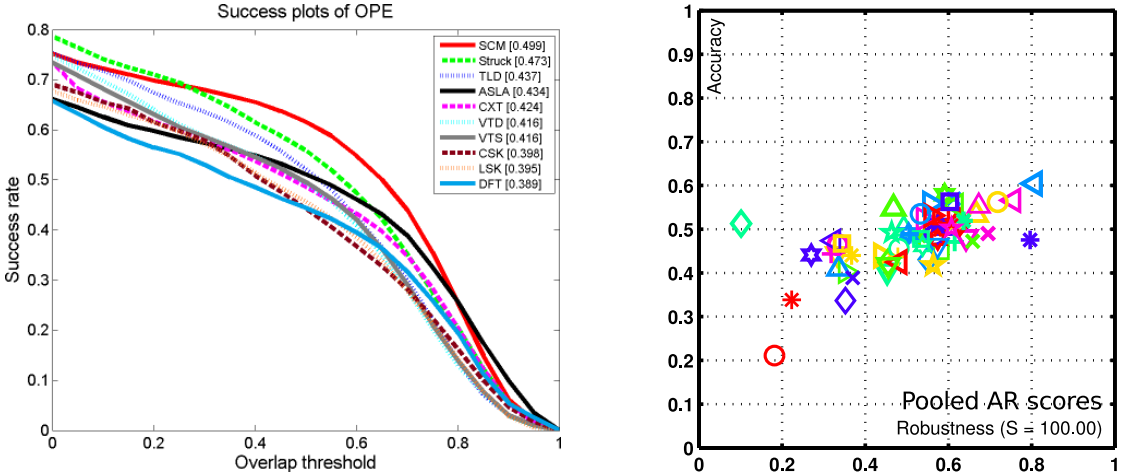
An evaluation methodology for one-shot object tracking consists of two essential things. The first ingredient are appropriate measures that assess the desired properties of tracking algorithms. While we have discussed properties of different measures at length already in the previous sections, we will discuss the choice of measures in existing evaluation methodologies for one-shot object tracking in Section 6.2.1. The second ingredient is a *dataset* that should be as diverse as possible to assess the behaviour of the compared trackers in different situations. Furthermore, it has to be equipped with ground truth annotations for all sequences. We will discuss this topic in Section 6.2.2. Finally, in Section 6.3 we describe the evaluation protocol that we employ for this work that aims at providing a fair and easily interpretable way of comparing long-term tracking algorithms.

6.2.1 OTB and VOT

When publishing research about tracking, authors have typically used best practices for evaluating and comparing their own approach to the state of the art. The Object Tracking Benchmark (Wu et al., 2013), OTB for short, was the first tracking methodology to achieve widespread acceptance in the tracking community. The authors of OTB proposed an evaluation methodology comprised of multiple experiments. For their main experiment called one-pass evaluation (OPE), each tracker is run using the first entry of the ground truth data as initialization. The resulting algorithmic output of all sequences is then concatenated and analyzed with respect to overlap ψ and the center error measure. Success plots are used to visualize the results, as shown in Figure 6.8a. The area under the curve is used to compute a final ranking and is visible in the top right corner of the plot. This value amounts to the respective average per-frame measure, as it was shown in Section 6.1.5. In two other experiments, called temporal robustness evaluation (TRE) and spatial robustness evaluation (SRE), Wu et al. test the robustness of trackers to perturbations in the initial bounding box.

There are three points of criticism that can be directed at OTB. One point is that by concatenating all sequences into a single sequence, longer sequence have more weight than shorter sequences. While one could argue that this is a desired effect, in our opinion one sequence should be interpreted as a single problem that a tracker has to solve. From this view, each sequence should contribute the same weight to the overall result. The second point is that a success plot of the overlap measure is not particularly revealing due to the inherent ground truth ambiguity, as discussed in Section 6.1.1. In fact, the most interesting part of the success plots in OTB is a slice of the success plot where the overlap threshold is 0.5, which is a reasonable requirement for the overlap. The third point of criticism is that the average overlap as a means of ranking trackers is prone to ground truth ambiguity.

Another methodology for object tracking has been developed by Čehovin et al. (2016), who analyze different measures with respect to their correlation and conclude that the two



(a) OTB (b) VOT

Figure 6.8: Evaluation methodologies. (a) The OTB tracking evaluation of Wu et al. (2013) employs a success plot of the overlap measure. (b) The VOT evaluation of Kristan et al. (2016) employs a combined measure of accuracy and robustness.

measures ψ and $failure\_rate$ correlate least. Based on this consideration, the visual object tracking (VOT) challenge (Kristan et al., 2016) was initiated, that aims at providing an annual challenge to measure progress in tracking research and at the time of writing has been held three times (Kristan et al., 2013; Kristan et al., 2014; Kristan et al., 2015). [20]. As in the VOT evaluation methodology trackers are automatically re-initialized after failure, these two measures can be interpreted as measuring the trade-off between accuracy (A) and robustness (R) and are visualized in an AR plot, as shown in Figure 6.8b, where each evaluated tracker occupies one coordinate in AR-space. Additionally, a ranking methodology is employed, aiming at capturing only statistically significant differences between trackers. Again, one problem with this methodology is that the average overlap is subject to ground truth ambiguity. To remedy this effect, starting with Kristan et al. (2014) rotated bounding boxes are used as ground truth data, that allows for a more accurate annotation. It has to be noted that the measures of the VOT challenge meant to evaluate short-term tracking algorithms only.

6.2.2 Dataset

Until recently, authors of tracking algorithms evaluated their approaches on a mere handful of sequences. An evaluation that is performed on few sequence however tells little about the capability of a tracking algorithm for achieving good results on sequences the algorithm was never tested on. While not directly applicable, this problem is closely related to the problem of *overfitting* in machine learning. For instance, a classical problem

[20] More details at http://www.votchallenge.net.

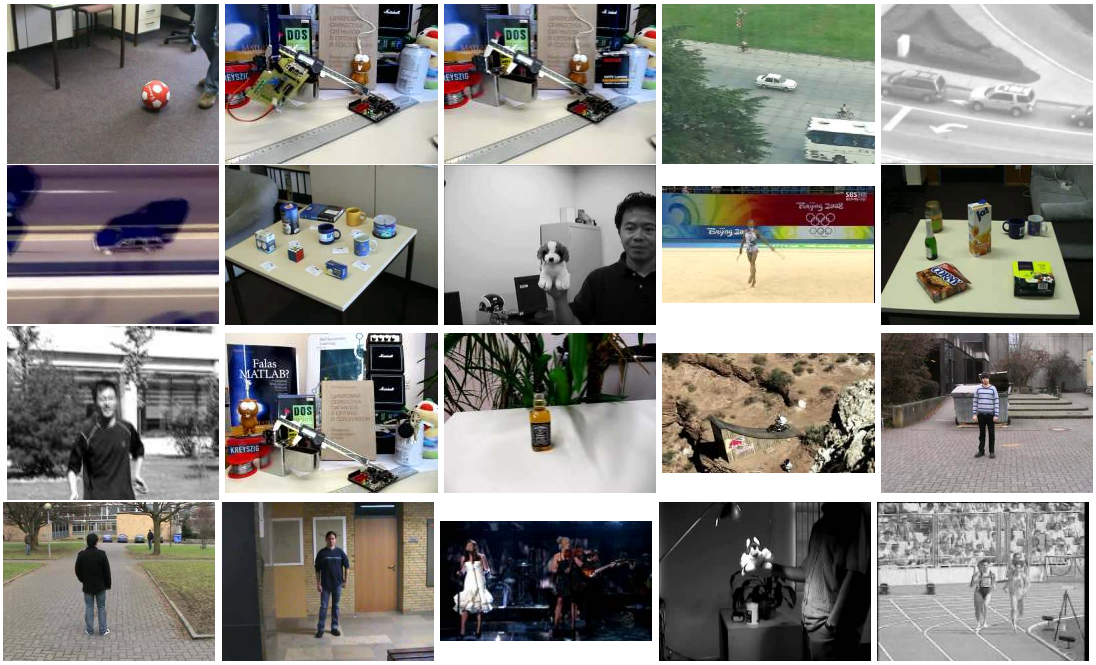


Figure 6.9: Example sequences from the Vojir dataset. From left to right, top to bottom: *ball*, *board*, *box*, *car*, *car 2*, *carchase*, *cup on table*, *dog1*, *gym*, *juice*, *jumping*, *lemming*, *liquor*, *mountain-bike*, *person*, *person crossing*, *person partially occluded*, *singer*, *sylvester*, *track running*.

in machine learning is classification, where the task is to correctly predict the class of an instance after being presented with a *training set* containing a number of labeled examples. It should come as no surprise that obtaining a low error rate on the training set itself is a trivial task, as this can be achieved by memorizing the whole training set. Such an approach will however yield a very high error rate on a dedicated *test set*, because the data will not be exactly the same as in the training set. Instead, the classifier should use the training set to find a decision boundary that generalizes well to unseen examples. As in one-shot object tracking there is no dedicated training data at all, one might be tempted to think that this problem does not exist. Extreme overfitting in one-shot tracking would mean that the tracking algorithm memorizes a direct mapping from image content to ground truth bounding boxes. While such a tracker would hardly make it through peer review, it should be made clear that virtually every tracking algorithm contains a number of parameters that can be tweaked one way or the other to improve the results in certain situations. When browsing the tracking literature it is next to impossible to find a paper that does not claim to outperform the state of the art. This is clearly possible as a tracker whose parameters were overfitted to a small custom set of sequences will perform poorly on another custom set of sequences.

With the publication of OTB and VOT, this situation has slightly improved, as they both provide a standardized set of sequences. This way, tracking algorithms at least

now overfit to the same dataset instead of overfitting to a custom selection of sequences, making the overall results more comparable. Furthermore, these standardized datasets tend to be much larger than what was previously used. OTB employs 50 sequences, while in the third edition of the VOT challenge 60 sequences were used. With a diverse dataset that large, it is much more difficult to artificially improve overall results by tweaking certain parameters. Even if the dataset itself is used to find optimal parameter settings, it is still plausible that the tracker will perform well on an unseen sequence.

Another interesting dataset was proposed by Vojir and Matas (2014), consisting of a compilation of 77 sequences, subsets of which were used by different authors for publishing one-shot tracking results. The sequences are extremely diverse with respect to the objects of interests, as shown exemplarily in Figure 6.9, but also differ considerably in sequence length.

6.3 Conclusion

Summarizing the previous discussion, a proper evaluation methodology for long-term tracking should account for sequence length, should not overly penalize single-frame errors and should allow for automatic re-initialization. The evaluation should be performed on a large dataset that should not have been compiled by the author of the tracking algorithm. With these considerations in mind, we propose the following evaluation protocol for a quantitative comparison of long-term trackers to other trackers that is similar to the one used in Kalal et al. (2012).

As discussed in Section 6.1.4, while it is not entirely clear that the overlap measure ψ provides completely intuitive results, we still choose it to remain comparable to other evaluation methodologies. To evade the problem of ground truth ambiguity, we convert each per-frame measure into the binary measures TP, FN and FP, as discussed in Section 6.1.2. Kalal et al. employ a threshold of $\theta_\psi = 0.25$, does not seem restrictive enough with Figure 6.6b in mind. Instead, we suggest to employ the more natural threshold of $\theta_\psi = 0.5$. To account for sequences of different lengths, we suggest to compute the recall as an accumulated measure for each sequence and to provide a success plot of all obtained recall values, providing a much better overview this way than what was possible with a tabular presentation. We will refer to this evaluation methodology in the next chapter as sequence-based OPE.

Chapter 7

Experiments

To avoid empty theorizing about the benefits of the concepts introduced in the previous chapters, it is necessary to evaluate them in practice. In Section 7.1 we evaluate CMT with respect to parameter settings, different part correspondence methods, the heuristics used in the DPMOST. We then provide quantitative results of CMT in comparison to baselines and state-of-the-art tracking algorithms as well as a speed comparison in Section 7.2. To exclude any bias from the evaluation that might be introduced from a personal selection of sequences, for our comparison we employ the Vojir dataset[21], that we referred to in the previous chapter. We use the sequence-based OPE evaluation methodology proposed in Section 6.3. Finally, in Section 7.3 we show qualitative results.

7.1 Analysis of CMT

As the central parameter in the DPMOST is the deformation threshold δ, it makes sense to evaluate this parameter exhaustively. To this end, we provide a plot of the effect of δ on the average recall on the Vojir dataset in Figure 7.1 on the left. Here, δ is varied from 0 to 100 pixels, where 0 requires complete rigidity in the model. The plot shows that a proper setting of δ is very important. When it is set too low, many correct correspondences are rejected. When it is set too high, many outliers are considered as inliers. In both of these cases, the localization of the object is hampered. There appears to be a "sweet spot" between $\delta = 15$ and $\delta = 40$. It is clear that the optimal parameter setting for δ depends on multiple factors. The main factor is arguably the deformation of the object, but also the size of the object and the image size play a role here. An interesting future research direction lies in adapting the parameter in an automatic fashion during runtime.

To measure the impact of the method used for establishing correspondences, we compare the effect of employing static correspondences, static-adaptive correspondences and disambiguated static-adaptive correspondences. The result is shown in Figure 7.1 on the right. As expected, the static correspondences perform much worse than the other two correspondence types. As soon as adaptive correspondences are added, the performance increases dramatically. While the addition of disambiguated static-adaptive correspondences does not lead to an increase this drastic, it nevertheless contributes a

[21] Available at http://www.gnebehay.com/cmt.

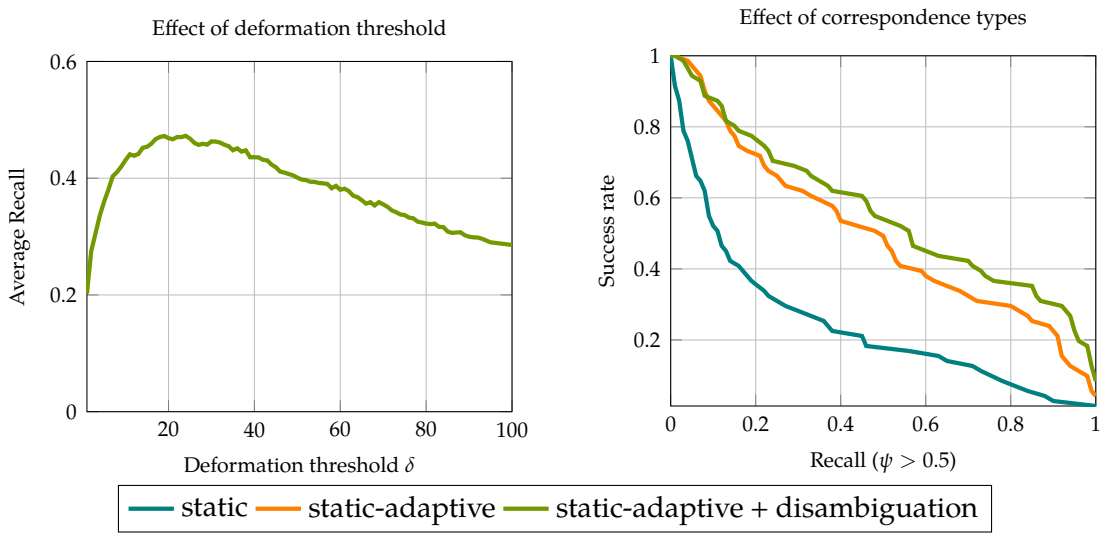


Figure 7.1: Left: Effect of the deformation threshold δ on the average recall achieved by CMT on the Vojir dataset. Right: Effect of different correspondence types. The largest increase in performance can be noticed when switching from static correspondences to static-adaptive correspondences.

certain share to achieving state-of-the-art results.

Clearly, the choice of how the the parts correspondences are obtained has a strong impact on the overall tracking result. In this experiment, we evaluate different combinations of part detectors and descriptors. For the part detectors, we selected GFTT (Shi and Tomasi, 1994), FAST (Rosten and Drummond, 2006), ORB (Rublee et al., 2011) and BRISK (Leutenegger et al., 2011). For the descriptors, we selected BRISK (Leutenegger et al., 2011), FREAK (Alahi et al., 2012), BRIEF (Calonder et al., 2010) and ORB (Rublee et al., 2011). As CMT is completely agnostic of the concrete part detectors and descriptors, we performed an exhaustive comparison of all possible combinations of the above. It has to be noted that all of the considered methods contain internal parameters that can be adjusted. For instance, all part detectors employ a threshold for rejecting candidate point locations. We left all of these parameters at their default values, as an evaluation of those parameters is out of scope for this work. The results of this experiment are shown in Figure 7.2, where each combination of part detector and descriptor was assigned a color according to the legend on the right. The plot is a success plot of the average per-sequence overlap values. It is interesting that no clear winner can be identified from this experiment, suggesting that their performance is dependent on the particular sequence. The reason that the combination of ORB and FREAK performs worst lies in the FREAK descriptor rejecting many parts detected by ORB due to some internal checks, leading to a degradation in performance.

To evaluate the correctness of the heuristic for estimating the scale of the object of interest that was presented in Section 4.2.3, we perform an experiment on the sequence *singer*, where the object of interest undergoes a dramatic change in size. In the top plot of

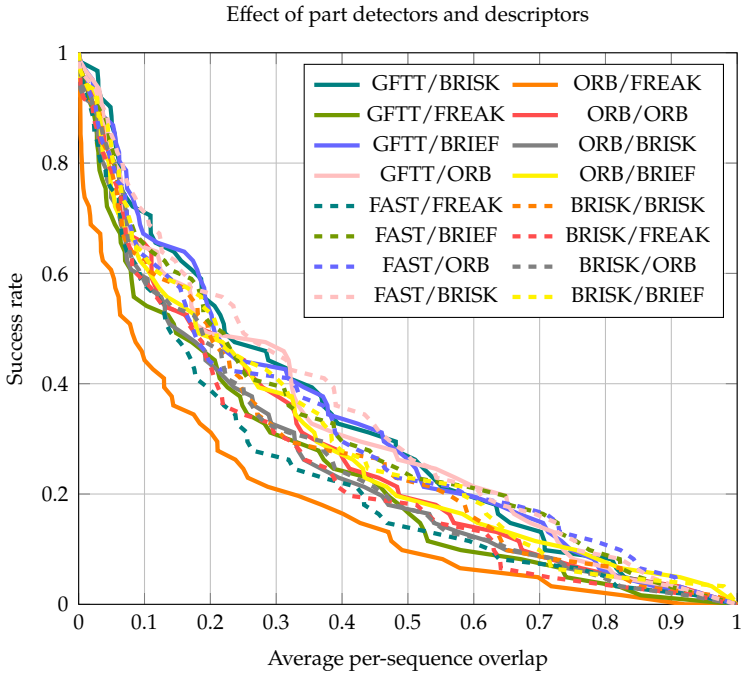


Figure 7.2: Comparison of part detectors and descriptors on the Vojir dataset.

Figure 7.4 two curves are shown depicting the scale of the object of interest in the *singer* sequence. The first curve, shown in green, refers to the scale s_{GT} of the object according to ground truth data. As an indicator for this

$$s_{GT} = \frac{\sqrt{w_t^2 + h_t^2}}{\sqrt{w_1^2 + h_1^2}} \tag{7.1}$$

was used, where w and h denote the width and height of the initial and current bounding boxes. We contrast this curve with the result s of the scale estimation in the DPMOST according to Equation 4.8, shown in blue. It is clearly visible that the two curves are almost identical, a feat that only few object tracking algorithms achieve. To visualize in more detail what is going on in Equation 4.8, we analyze the distribution of the pairwise scale changes

$$S = \left\{ \frac{\|x_i^t - x_j^t\|}{\|x_i^1 - x_j^1\|}, i \neq j \right\} \tag{7.2}$$

that appears there. In Figure 7.4 we show S in the form of a histogram for six frames of the *singer* sequence. Again, the ground truth scale s_{GT} is shown in green and the estimated scale s is shown in blue. Intuitively, the more compact the histogram, the more the pairwise scale comparisons "agree" on the scale of the object. It can be seen that in later frames the histograms get less and less compact. This is due to the fact that the

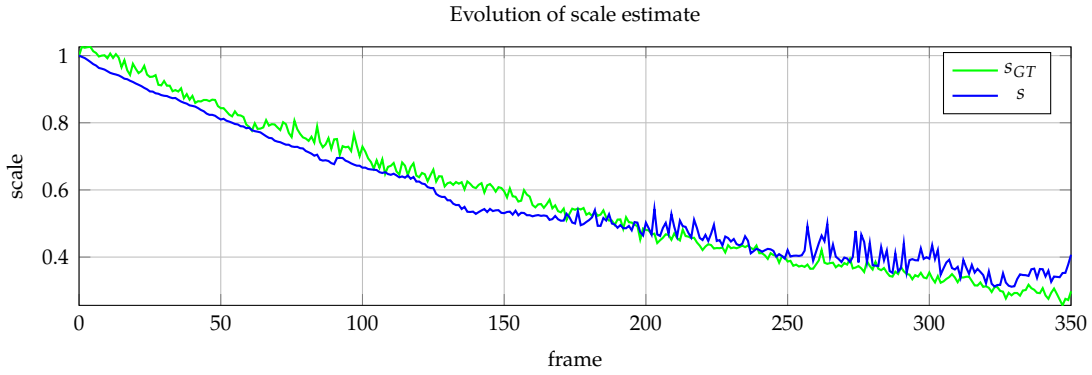


Figure 7.3: Evolution of the scale estimate s on the sequence *singer* compared to the scale defined by the ground truth.

object appears in a deformed way, leading to the drifting of object parts. Additionally, some unrecognized outliers taint the scale histogram. Nevertheless, the median is robust enough to cope with these deviations and is able to yield the correct result for all six frames. It would be interesting to perform a similar experiment for the estimation of the rotation, but for this more precise ground truth data would be necessary.

We implemented RANSAC and a variant of the Generalized Hough Transform in order to investigate the performance difference between our method and well-established methods for robustly estimating outliers and transformation parameters. For RANSAC, we compute an exact solution for a similarity transform between the correspondences. We tested a range of different parameter settings and employed those that yielded best results for our comparison. For the GHT we employ coarse bins of a tenth of the width and height of the image for the x and y dimension, respectively as well as 10 bins for the scaling dimension and 20 bins for the rotation dimension. It has to be noted that we added the comparison to the GHT for reasons of completeness, as it is not practical when used with parameter space of more than three dimensions. The results in Figure 7.5 were computed on the Vojir dataset, showing the success rate with respect to recall. The results show that the restrictive baselines perform poorly compared to our approach, the main reason being their inherent incapability of handling deformations.

7.2 Quantitative Results

In this section we apply the long-term tracking methodology described in the previous chapter to evaluate our proposed approach and compare it to a several other trackers. Clearly, there is an abundance of tracking algorithms available and one faces the challenge of selecting algorithms for comparision. To make sure that we compare to the state of the art, we selected the top 3 ranking trackers SCM, STR and TLD from the OTB evaluation. Additionally, we included FT, a basic part-based tracker as well as the CT tracker that

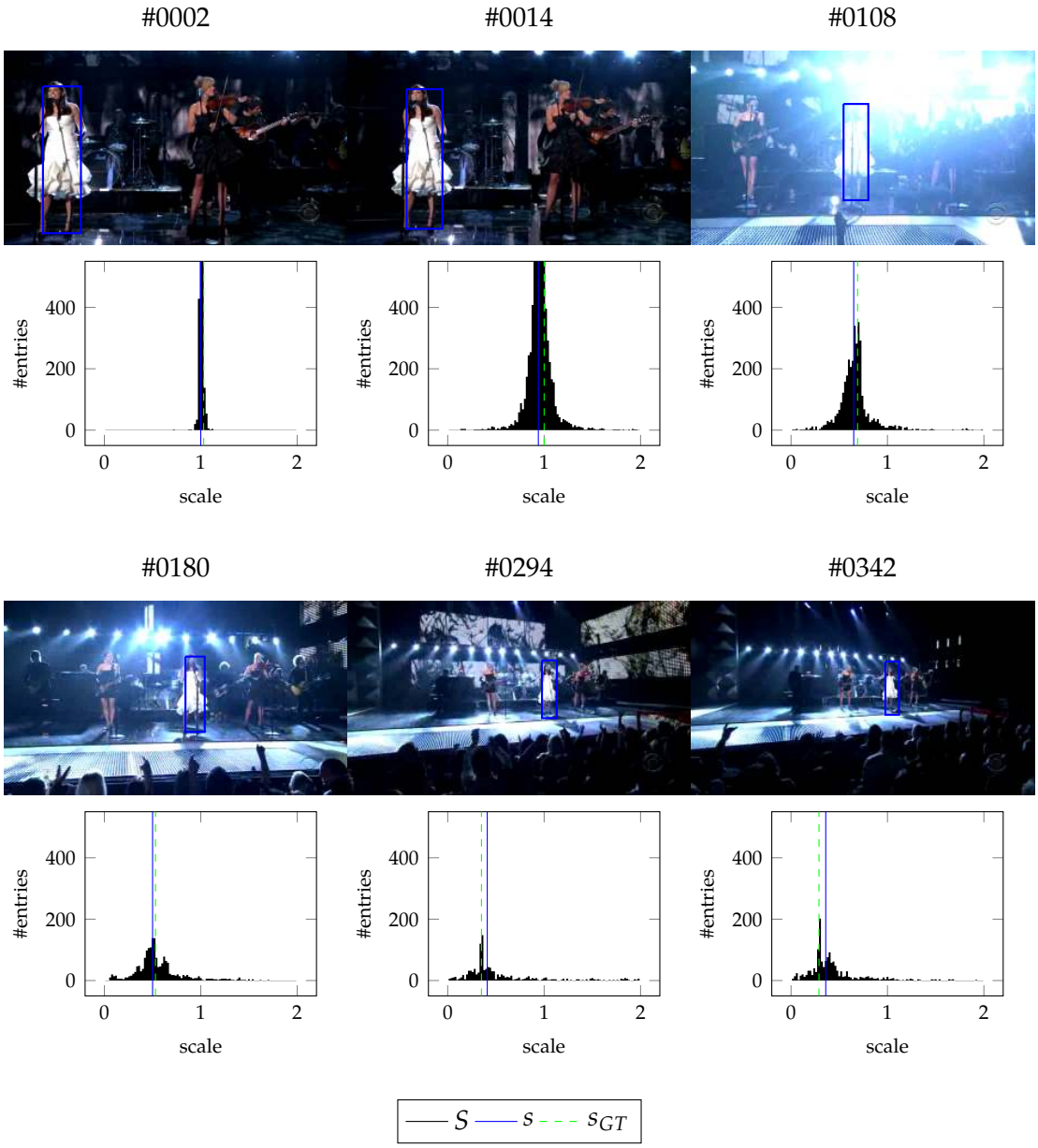


Figure 7.4: Distribution of the pairwise changes in scale S for 6 individual frames. The x axis denotes the scale. The y axis denotes the absolute number of entries in the respective histogram bin. s and s_{GT} denote our estimate and ground truth values for scale, respectively. While the histograms become less compact as the object becomes smaller, the heuristic for estimating s still delivers satisfying results.

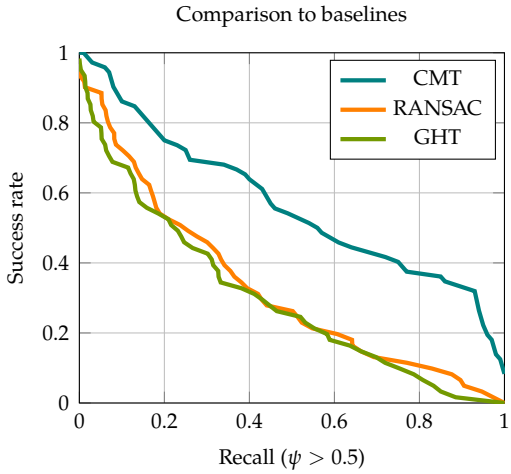


Figure 7.5: Comparison of CMT to RANSAC and the GHT.

Abbrev.	Method
STR	Structured Output Tracking (Hare et al., 2011)
TLD	Tracking-Learning-Detection (Kalal et al., 2012)
SCM	Sparsity-based Collaborative Model (Zhong et al., 2012)
FT	Fragments-based Tracking (Adam et al., 2006)
CT	Compressive Tracking (K. Zhang et al., 2012)

Table 7.6: State-of-the-art tracking algorithms used in the comparison to CMT.

operates at a high processing speed. We deliberately did not include HoughTrack into the comparison, as this tracker was never meant to be employed on long sequences, rendering a potential comparison unfair. An overview of the selected trackers is given in Table 7.6. The source code of all competing trackers was obtained from the websites of the respective authors and all parameters were left at their default values. Care was taken for the implementation of CMT[22] to remain compatible with as many platforms as possible, making a comparison of CMT to newly published trackers easy.

For CMT, we employ the parameters $\delta = 20$ and employ BRISK (Leutenegger et al., 2011) keypoints and descriptors for establishing part correspondences. The parameter θ_ϕ can be set for suppressing uncertain tracking results, but for the evaluation protocol that we use this is of no importance.

For our first, arguably most important experiment we perform our proposed sequence-based OPE evaluation on the Vojir dataset. The resulting success plot of the recall for the individual sequences is shown in Figure 7.7 on the left. To avoid misunderstandings, the x axis denotes the threshold on the recall achieved on individual sequences, while

[22] Available at http://gnebehay.com/cmt

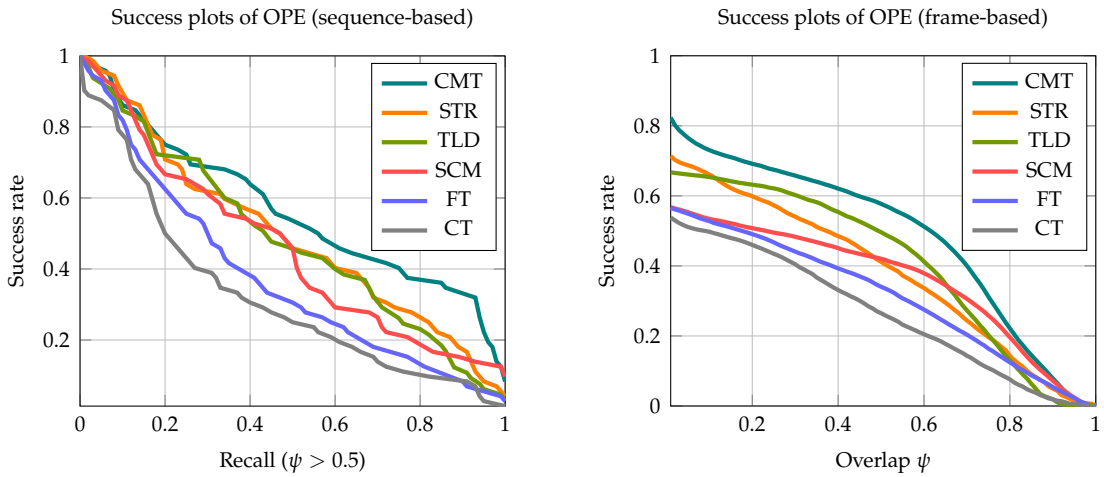


Figure 7.7: The main experiment for evaluating the overall performance of our proposed tracker CMT to the state of the art. Left: Comparison using our proposed methodology. Shown here is a success plot of recall for an overlap threshold $\theta_\psi = 0.5$. Right: Results according to Wu et al. (2013) in the form of a success plot of the overlap measure. Our method dominates both evaluations.

the threshold on the overlap measure ψ remains fixed, as shown on the x axis label. (in contrast to OTB). As one moves to right of the plot, less sequences "survive" the increased requirement on the recall, leading to a reduction in the success rate. Therefore, all curves start in the top left corner with a success rate of 1 and end in the bottom right corner with a success rate of 0. What is now interesting is how the curves bridge the way between those extrema. All trackers achieve roughly equal results in a recall range of 0 to 0.1. After this, results start to diverge and the results of FragTrack and CT deteriorate. The other trackers continue to achieve similar results up to a recall threshold of 0.3, which is when our proposed tracking algorithm CMT begins to dominate the success plot. It is noteworthy that the distance between CMT and the second-ranked tracker STR is especially high for recall values of 0.9, where 35% of the sequences survived. While it has to be considered that on a different set of sequences the results might be slightly different, it is safe to say that CMT achieves excellent tracking results compared to the state of the art. This is especially interesting, as there is no permanent model update in CMT in contrast to all other competing approaches.

It is now interesting to compare the outcome of these experiments to the OTB evaluation methodology proposed by Wu et al. (2013). To this end, we provide the success plot of the overlap measure ψ after concatenating the result of all sequences. This success plot is given in Figure 7.7 on the right. It is now difficult to compare the results between these two methodologies directly, as one is based on frames and the other one on sequences. However, the OTB evaluation displays a similar picture as our proposed methodology, as our method dominates this evaluation as well. Also, the inaccuracy of STRUCK is

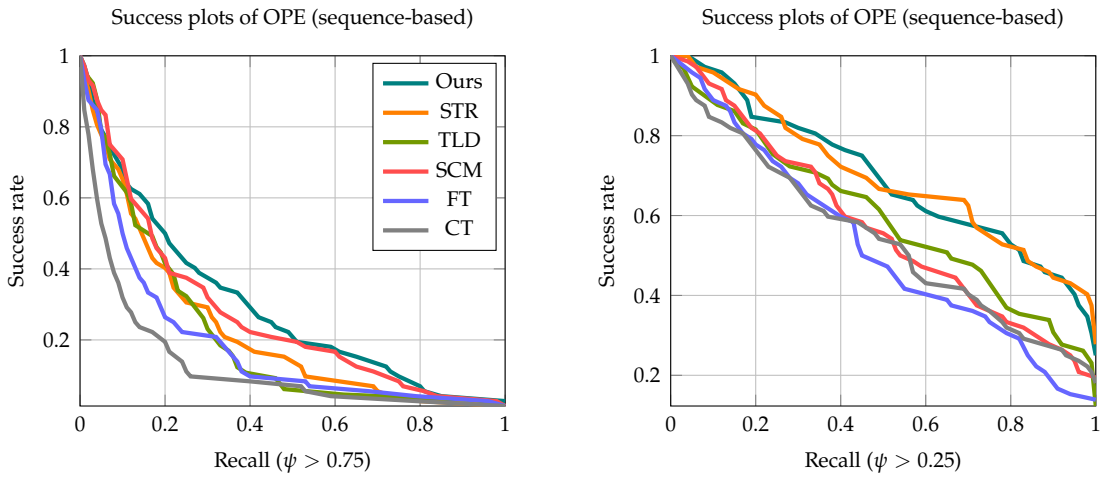


Figure 7.8: Success plots depicting the effect of different thresholds on the overlap ψ. On the left, $\theta_\psi = 0.25$ was used, improving the results of some trackers considerably, especially STRUCK. On the right, $\theta_\psi = 0.75$ was used, leading to a degradation of all results. The legend is the same as in the left plot.

visible, as the decline of its curve is much steeper than the one of CMT or TLD. Again, it has to be noted that there are long sequences in the dataset that bias the result of a frame-based evaluation.

In our second experiment, we investigate the role of the overlap threshold θ_ψ. To this end, we repeat the previous experiment with two different values $\theta_\psi = 0.75$. and $\theta_\psi = 0.25$ This is interesting because it steers the required *accuracy* of the tracker (modulo ground truth ambiguity). The respective success plots for the sequence-based OPE are shown in Figure 7.8. As for the left plot in Figure 7.8, a threshold of $\theta_\psi = 0.75$ is extremely high (especially with Figure 6.6b in mind), leading to a deterioration of the results of all trackers. Clearly, for a lower value of θ_ψ, results will improve for all trackers. It might however turn out that one tracker improves so much more than another tracker that it is able to outperform it. This is interesting, as some applications might not require a high level of accuracy, while others do. The success plot showing an overlap threshold of θ_ψ demonstrates that STRUCK is now on par with CMT, meaning that the results of CMT are more accurate than STRUCK. This is easy to explain, as in contrast to STRUCK, CMT estimates the scale and the rotation of the object of interest, leading to a considerable improvement of accuracy. In fact, it is an interesting idea to visualize the effect of an increased threshold θ_ψ on the average recall over all sequences. We did however not consider such a plot for our experiments.

The final experiment concerns the question of processing time. To show how CMT compares to other state-of-the-art tracking algorithms in this respect, we measured the wall clock time in seconds while performing the experiments from Section 7.2. This way, we obtained a time for each sequence/tracker combination. By dividing the number of

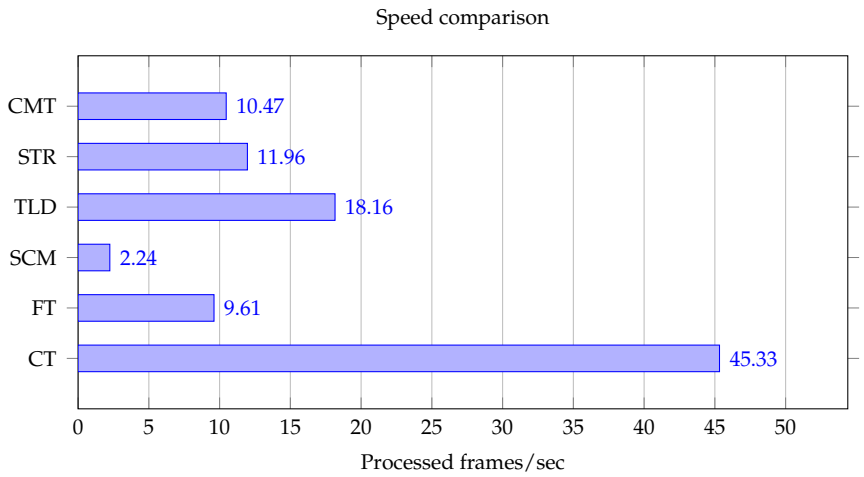


Figure 7.9: Speed comparison. CT is the fastest method by a large margin, while SCM is the slowest. Our method achieves a frame rate of 10.47, placing it fourth in the ranking.

frames in the respective sequence by this time, one obtains the average number of frames that were processed each second. By again computing the average over all sequences, an overall measure for the speed of a tracking algorithm can be obtained, which is the content of the plot in Figure 7.9. This plot clearly shows that CT is the fastest of the considered tracking algorithms (45.33 fps), which is not especially surprising, as it was selected for this purpose. It also shows that SCM is by far the slowest algorithm in the comparison (2.24 fps). CMT (10.47 fps) is ranked fourth out of six trackers. As no effort was put into increasing the speed of CMT during this work, making it faster is an interesting future research direction.

7.3 Qualitative Results

To provide a better impression of the role of the DPMOST in CMT, qualitative results are given in the Figures 7.10-7.13, showing results from 8 sequences[23]. where each sequence was selected for a special difficulty. The first image of each sequence is the initial frame, while the other frames were hand-picked to show interesting events. The individual frames are arranged from left to right and then top to bottom. The white points in the image denote parts from $\mathcal{L}^{\omega}$, while the red points denote correspondences that have been identified as outliers. The blue rectangle denotes the output bounding box of CMT.

In the sequence *singer* (Figure 7.10, top), the main challenge is the scale change of the object. The target starts out by occupying almost the complete vertical image space, but is reduced to a fraction of this size in the last image. While CMT is able to deal with this change successfully, it is interesting to note that still many outliers are no

[23] More results in the form of videos are available at http://www.gnebehay.com/cmt.

longer recognized correctly in the last frame. This problem is due to the non-adaptive deformation threshold δ.

The sequence *liquor* (Figure 7.10, bottom) shows how an object similar to the object of interest is handled by CMT. While in frames 2 and 4 it is clear that no outlier correspondences were established on the similar object (the bottle to the left of the object in frame 2), a considerable amount of adaptive correspondences are "stolen" by the second object in frame 3 after moving it in front of the original target. After the original object becomes visible again, these outlier correspondences continue to be recognized as inliers, until a division of this single cluster occurs, leading to a large number of outliers that are suddenly recognized.

An interesting effect can be observed in the *mountain-bike* sequence (Figure 7.11, top). Here, the object "loses" adaptive correspondences on its boundary from time to time, as depicted in the second frame. As these correspondences are correctly re-established by static ones, tracking of the object still succeeds, demonstrating the interplay between static and adaptive correspondences.

The sequence *juice* (Figure 7.11, bottom) shows the output of CMT for a completely rigid object. This sequence demonstrates that objects that do not appear deformed are easier to track than deformed objects as establishing correspondences poses less problems. In the displayed frames, not a single outlier occurs.

In contrast, the sequence *gym* (Figure 7.12, top) shows an object that is difficult to track due to its extreme intrinsic deformation. In the displayed frames, it becomes clear that the quality of correspondences in such a case can be very low, as there are many soon-to-be outliers in the vicinity of the object of interest. This sequence demonstrates the importance of modeling explicity the deformation of the object of interest, as it is done in the DPMOST, allowing the object of interest to be still tracked successfully.

The sequence *ball* (Figure 7.12, bottom) contains an object exhibiting a repeating texture. This repeating texture poses certain difficulties for static correspondences, as the matching of many similar descriptors is very difficult. Here, the benefit of disambiguated correspondences comes into play, allowing to excluded descriptors in the matching process based on their location on the object. This way, a repetitively textured object can be tracked successfully as well.

The sequence *person occ* (Figure 7.13, top) demonstrates the robustness of part-based object models with respect to partial occlusions. Here, a person disappears partially behind a column and re-appears shortly after. For the non-visible parts no correspondences can be obtained, but since enough object parts remain visible, the bounding box can still be inferred correctly.

The sequence *board* (Figure 7.13, bottom) brings an interesting phenomenon to attention, namely the selection of background parts in the initial frame. As the object of interest is slightly skewed in the first frame, some parts on the background are considered as belonging to the object model. As in CMT, the part models are never updated, these "cuckoo's eggs" remain there forever and cause outliers in every frame. However, as the

numbers of background parts is rather small in this sequence, this poses no problem for successfully tracking this object.

7.4 Conclusion

The experimental evaluation in this chapter has revealed some interesting insights. First and foremost, from the comparison of CMT to the state of the art can be concluded that an algorithm based on the DPMOST is able to achieve excellent tracking results. Furthermore the success of CMT questions the prevalent paradigm of permanently incorporating new appearances of the object of interest into the object model, as the combination of static and adaptive correspondences prove extremely effective to render this model update unnecessary. In summary, one can go as far as to say that CMT is the first part-based tracker that is able to provide robust results in long-term tracking scenarios, while inheriting all the advantages of a part-based tracking method, such as the ability of handling deformations and robustness to partial occlusions. While CMT will not go down in history as the fastest tracker ever invented, it provides a reasonable trade-off between tracking performance and speed to make it suitable for realistic applications. This section concludes the experimental part of this work. The remaining chapter provides a summarization and puts the contributions into a broader perspective.

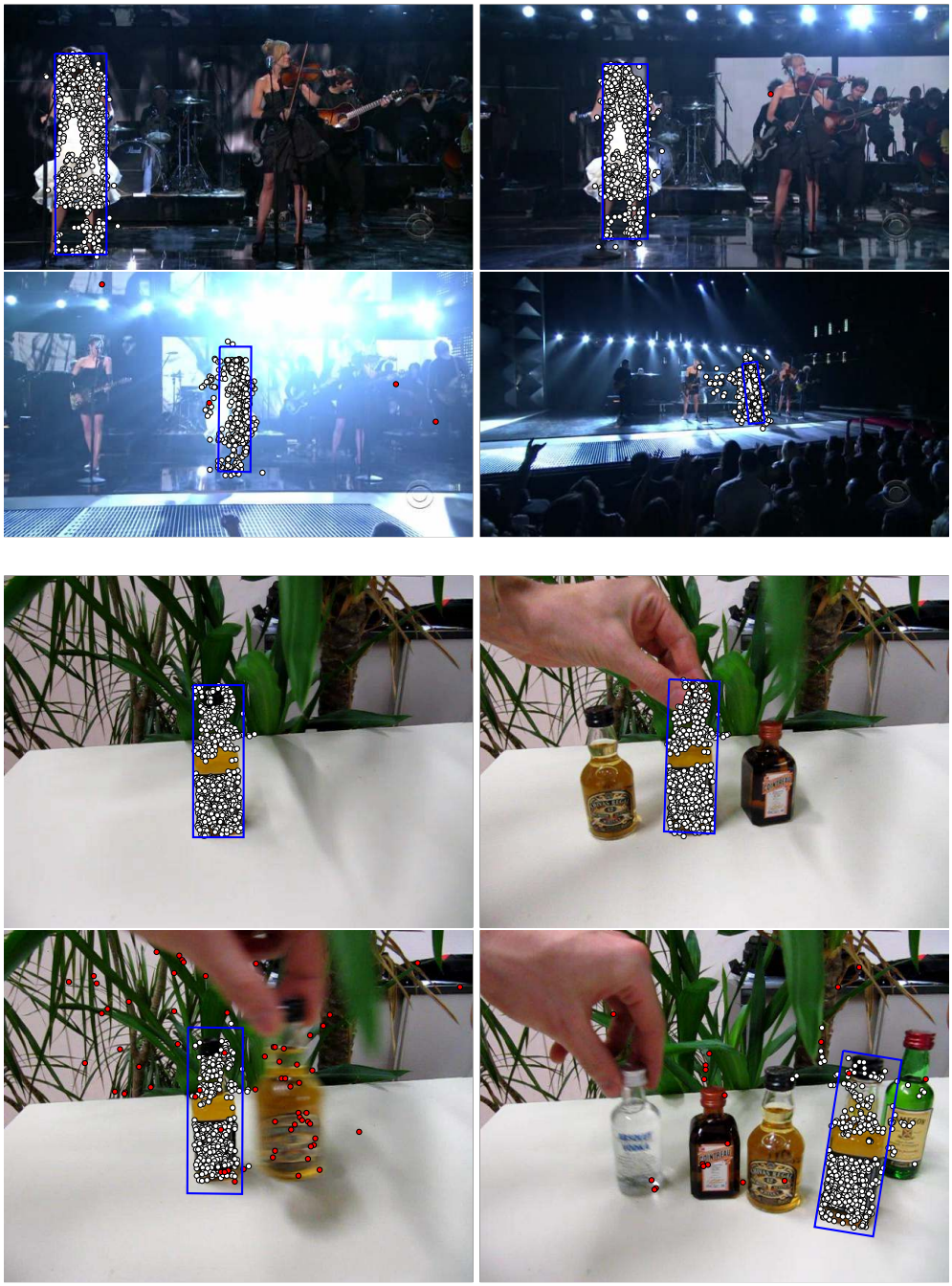


Figure 7.10: Qualitative results on *singer* and *liquor*.

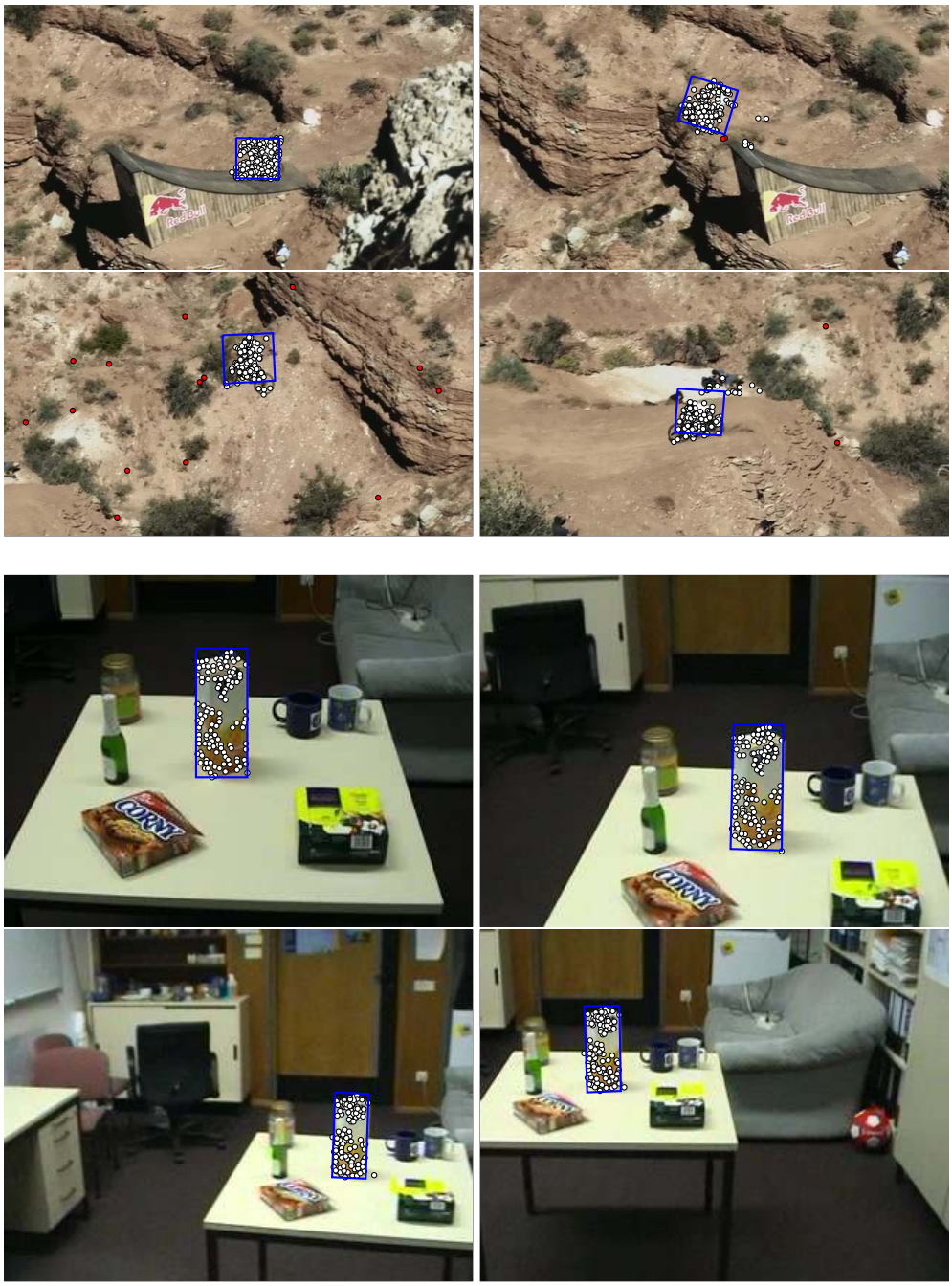


Figure 7.11: Qualitative results on *mountain-bike* and *juice*.

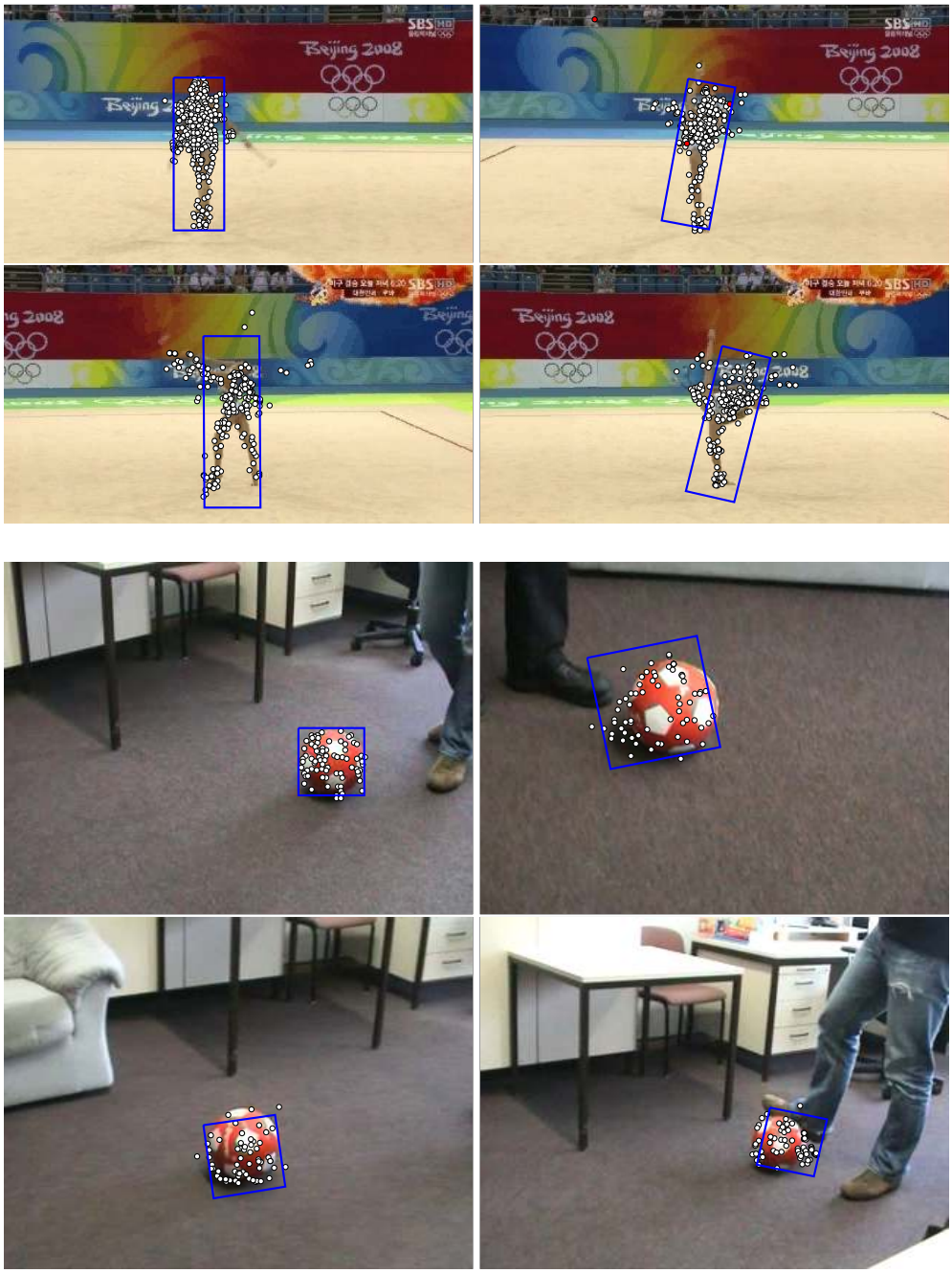


Figure 7.12: Qualitative results on *gym* and *ball*.

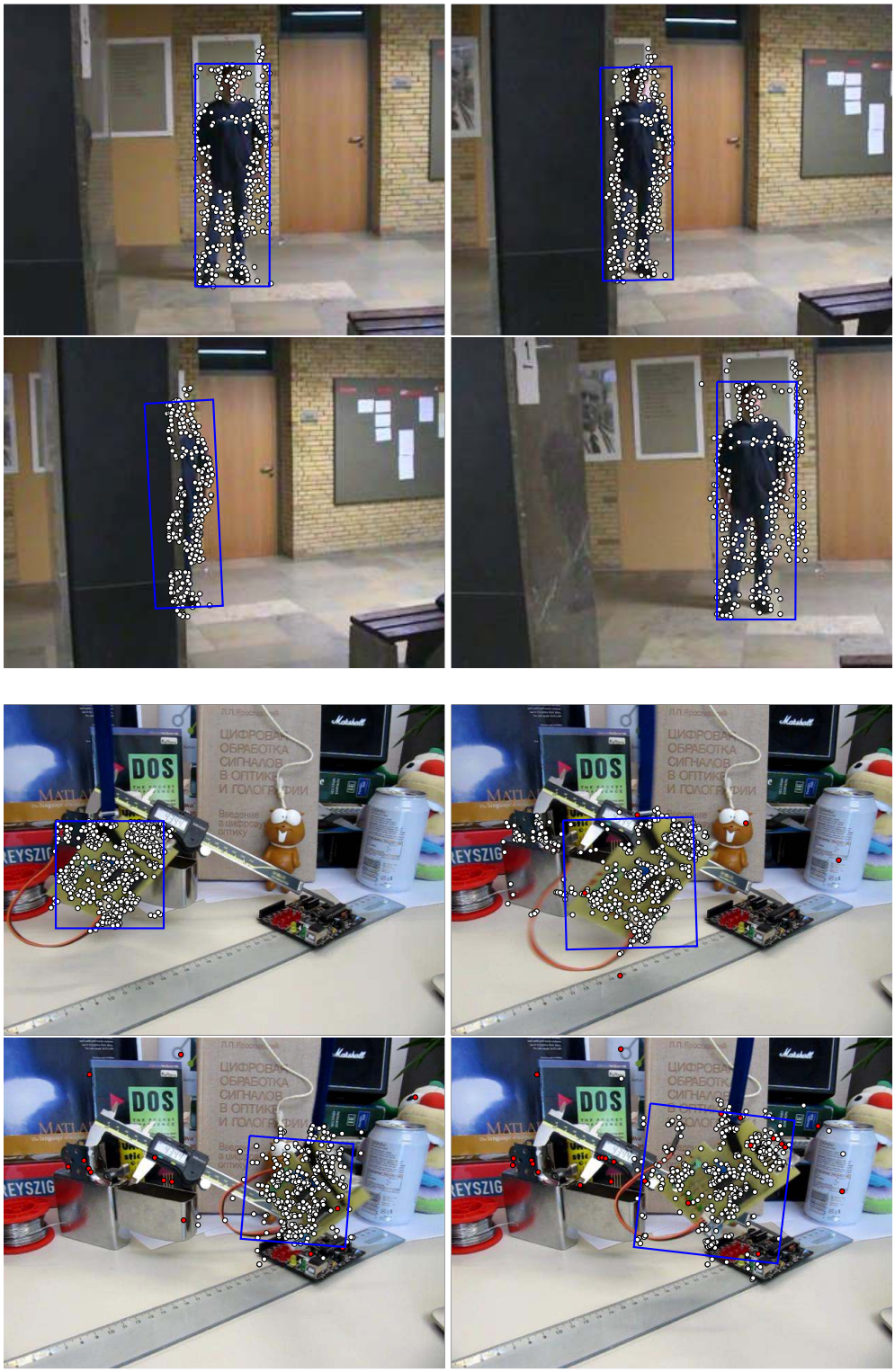


Figure 7.13: Qualitative results on *person occ* and *board*.

Chapter 8

Conclusion

In the last chapter, we have shown how the concepts introduced previously stand up to an experimental evaluation. In this final chapter, we give a summary of the insights gained during this work in Section 8.1. From the current state of this work several routes for future research directions are viable, that we present in the order of decreasing "low-hanging fruitness" in Section 8.2. Finally, we assess the current state of the art in one-shot object tracking to give an outlook about how the field might and should evolve in Section 8.3.

8.1 Summary

In this work we have discussed the topic of one-shot object tracking at length and have seen that part-based approaches to tackling this problem have many advantages over holistic ones. Their main advantage is that they endow object models with redundancy, allowing for failures of tracking individual parts to be handled gracefully. At the same time, part-based models are much more difficult to train than global models. We have argued in this work that the deformation of objects plays a crucial rule in making the problem of one-shot tracking difficult, as deformation can come about not only intrinsically, but also by external factors such as a different view on the object of interest. Our main contribution to the field of one-shot tracking is the novel object model DPMOST that allows for modeling the deformation in star-shaped object models in a natural and principled manner. Additionally, we have shown how the question of maintaining an object model that is both robust and accurate can be answered by static-adaptive correspondences, avoiding a permanent model update at all and instead performing only short-term updates. Our experimental evaluation has shown that these concepts are fruitful and allow for achieving excellent results when implemented in a tracking algorithm.

The topic of one-shot object tracking is attractive from two different perspectives. First, it is a rich source for potential applications as the only input required are an image sequence and a bounding box in its first frame. Second, it is interesting from a scientific point of view as a solution to the problem of one-shot tracking would have great implications on the whole field of computer vision. Part-based methods are one piece in this jigsaw. Representing the object of interest as multiple parts is in general

advantageous, as the loss of individual object parts does not lead to an immediate tracking failure. However, a certain share of additional complexity is introduced by part-based models.

In our proposed deformable part model for one-shot object tracking (DPMOST), The key ingredient to allowing for deformations to be handled is the assumption of connecting parts by interpreting their voting behaviour differently compared to standard star-shaped object models. Instead of accumulating the votes in bins or cells, by computing transitive consensus between votes a natural partitioning of inlier and outlier parts emerges. This concept effectively captures the deformation of the object of interest. In case of changes in scaling and in-plane rotation, heuristics based on pairwise part properties can be applied to address these changes.

Static-adaptive correspondences combine the advantages of performing a robust but inaccurate global search and an accurate, but more fragile local search. We have shown experimentally that great improvements in tracking performance can be obtained by employing this combination. From a biological viewpoint, static correspondences are equivalent to long-term memory, which is no longer subject to change. On the other hand, adaptive correspondences are related to short-term memory, which can as easily be updated as erased without ever affecting the long-term memory.

The tracking algorithm CMT that emerged from this work has some appealing properties, making it suitable for practical applications. First, it is reasonably simple. This property should not be underestimated as a simple algorithm can be better understood by practitioners as well as more easily adapted to personal needs. Second, it is general. This generality allows for instance to exchange the part detector or the descriptor using a minimal amount of effort, allowing for new advances in these fields to be leveraged. Third, it is available in a license that allows virtually unrestricted use even in commercial scenarios.

8.2 Future Work

One aspect of this work that was out of scope is the comparison of different methods for computing the sparse optic flow for establishing adaptive correspondences. The Lucas-Kanade method was chosen as the workhorse for our experiments because of its solid and reliable implementation as well as its cheap computation. However, other methods might work better or might even be faster. To this end, a thorough investigation should be undertaken to settle this question.

Another aspect is the overall computational performance of CMT. While its runtime performance certainly is sufficient for non-critical applications, no effort was undertaken during this work to explicitly speed up computational bottlenecks. Additionally, the aspect of parallelization was not touched all, even though this topic is a fruitful one. For instance, the descriptor matching, the clustering step and the scale and rotation estimation are obvious candidates for parallel processing.

The DPMOST is almost immediately applicable to the tracking of multiple objects in a scene. The transitive consensus and its clustering implementation prove beneficial here, because they allow for the emergence of multiple hypotheses unlike other methods such as RANSAC. One strong advantage of the part-based formulation from this work over sliding-window-based object tracking algorithms such as TLD lies in the fact that our parts formulation is as general as possible. This means for instance that many computations performed while obtaining static correspondences for one object can be re-used for a second object of interest, such as the computation of descriptors. Compare this to the sliding-window approach of TLD, where the global search is performed using the dimensions of the input bounding box and a second object that is added duplicates this effort. Furthermore, the tracking of multiple objects in CMT is not only feasible, but might also lead to interesting ways of improving the single-target results.

Currently, the DPMOST is invariant to changes in scale and in-plane rotation. An obvious question is whether this invariance can be extended to full affine or even perspective transformations. While an answer to this question seems straightforward, it must be noted that these transformations are actually defined on rigid objects. In the case of deformable objects approximations to these transformations can be computed at best. A better question might be how to improve on the bounding box representation for output in general to arrive in an optimal case at a pixel-wise segmentation of the object.

An unpleasing aspect of the DPMOST is the deformation threshold δ that currently has to be set manually or determined by evaluating different values. It is desirable that this threshold be set in a more automated fashion. Ideally, one would be able to identify the maximum deformation of an object of interest and set the threshold accordingly. This task seems however as difficult as updating an object model without introducing errors. Taking this idea a step further, it seems attractive to introduce different deformation thresholds $\delta_1, \ldots, \delta_N$ for each individual part. This makes sense as typically not all object parts exhibit the same potential amount of deformation. It is however unclear how these thresholds should be adjusted accordingly.

One drawback of CMT is obviously the requirement of some sort of texture on the object of interest. On homogeneous objects, neither static nor adaptive correspondences will function well, leaving the DPMOST with nothing to work with. These objects depict one limitation of part-based models in general, namely their inability to address objects where no interesting parts exists. While one could argue that these objects are not particularly interesting, for certain applications they might be still be relevant. In order to solve this, different part detectors and descriptors seem necessary that allow for the detection of parts even in homogeneous object regions. Color information might be a stronger cue in these cases than the descriptors that were dealt with in this work, making use exclusively of gray-scale image information. On anouther route, object contours or edgels can be explored as a replacement for the sparse points that were used in this work.

While we have shown in this work that the omission of an update step is not in contradiction with achieving state-of-the-art results, an interesting research direction

lies in investigating whether such an update step can still be beneficial to overall tracking performance in spite of the temporary model update already performed in adaptive correspondences. Essentially, introducing an update step means updating the reference configuration of parts Z as well as the corresponding descriptors P. How this update should be performed best is not entirely clear. One strategy might be to incorporate all newly found parts in b_t into the object model. This way however, many parts that have been in the model already are inserted there a second time. It may be a good idea to therefore incorporate only parts that are sufficiently dissimilar from existing object parts. On the other hand, it is worthwile to consider removing certain parts from the object model, especially when there is evidence that the part in question has been seen in the background. Also, currently all background parts in the first frame are currently used to improve the results while establishing static-adaptive correspondences. It might be worthwile instead to add background parts to the reference set only if there is an indication that they are consistenly confused with object parts.

8.3 Outlook

In the previous section, we have discussed how to possibly build on the contributions from this work. In this very final section, we assess the current state of one-shot object tracking in general and point out at more fundamental questions that have to be answered before the problem of one-shot object tracking can be considered to be solved. While we have shown in the experimental section that our proposed tracker CMT outperforms the state of the art, let us now put these results into perspective. If we assume that a sequence can be considered "solved" when an algorithm is able to track the object correctly in more than 90% of the frames, then Figure 7.7 tells us that CMT achieves this result on approximately 35% of the sequences of the considered dataset with quite a big margin to its closest follower. From this, one can argue that CMT has a chance of 35% of working reasonably well on any given random sequence. Even though it is clear that in practice also much better results can be obtained when the sequence "suits" the assumptions behind CMT, this number is quite depressing. As standardized tracking benchmarks have appeared only recently, it is currently not possible to gauge the progress in tracking research over a longer timespan. While there certainly is a continuous improvement in the performance of tracking algorithms, at the same time fundamental questions are still unanswered.

As we have seen troughout this work, the question of how to update an object model is closely linked to the stability-plasticity dilemma. Static-adaptive correspondences provide an interesting and novel way to addressing this dilemma. This technique acknowledges the fact that an update to an object model is always error-prone and can lead to unwanted effects. However, ruling out these errors by not updating the model bears the danger of getting "out-of-sync" with the object of interest. To a human, the task of recognizing an updated appearance of an object appears trivial. However, it has to be

considered that human beings went through four billion years of evolution and have been bombarded for years of their life with visual stimuli of all kind of sorts. Expecting a similar feat from an algorithm that is presented with a rectangular area in an image as the only hint about how one object is different from the rest of the universe is slightly unrealistic. The success of representational learning techniques (Krizhevsky et al., 2012) in the field of image recognition is based on two essential factors, namely the creation of the very large dataset ImageNet (Russakovsky et al., 2015) and the advances in parallel computing in the form of GPUs. We therefore argue that the essential ingredient in achieving a similar performance boost in one-shot object tracking as in image recognition is *data*. This data can basically appear in two different forms. A collection of *unlabeled* video sequences could be used to learn general concepts about how the world should be perceived. This data already exists and is accessible in the form of millions of videos that have been uploaded to online video platforms such as YouTube. While arguably these unlabeled sequences are similar to the input that the visual cortex in humans requires to advance, for machine learning techniques this data is much harder to process than *labeled* data. A large dataset of videos containing labeled objects does not exist to date and is tedious and expensive to create. However, we strongly believe that it is worth the effort. Such a labeled dataset would allow for learning how objects change their appearance in videos in a much more principled manner than any hand-crafted part detector and descriptor could ever allow for.

List of Figures and Tables

Publications

G. Nebehay, W. Chibamu, P. R. Lewis, A. Chandra, R. Pflugfelder, X. Yao. "Can Diversity amongst Learners Improve Online Object Tracking?" In: *Multiple Classifier Systems*. 2013, pp. 212–223.

G. Nebehay, R. Pflugfelder. "TLM: Tracking-Learning-Matching of Keypoints." In: *International Conference on Distributed Smart Cameras*. 2013, pp. 21–26.

B. Dieber, J. Simonjan, L. Esterle, **G. Nebehay**, R. Pflugfelder, G. Fernandez, B. Rinner. "Ella: Middleware for Multi-camera Surveillance in Heterogeneous Visual Sensor Networks." In: *International Conference on Distributed Smart Cameras*. 2013, pp. 167–172.

M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, **G. Nebehay**, F. Gustavo, T. Vojir. "The Visual Object Tracking VOT2013 challenge results." In: *Workshop on the VOT2013 Visual Object Tracking Challenge*. 2013, pp. 98–111.

M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, **G. Nebehay**, F. Gustavo, T. Vojir. "The VOT2013 challenge: overview and additional results." In: *Computer Vision Winter Workshop*. 2014, pp. 61–68.

G. Nebehay, R. Pflugfelder. "Consensus-based Matching and Tracking of Keypoints for Object Tracking." In: *Winter Conference on Applications of Computer Vision*. Best Paper Award. 2014, pp. 862–869.

M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Cehovin, **G. Nebehay**, T. Vojir, G. Fernandez. "The Visual Object Tracking VOT2014 challenge results." In: *Workshop on the VOT2014 Visual Object Tracking Challenge*. 2014, pp. 191–217.

G. Nebehay, R. Pflugfelder. "Clustering of Static-Adaptive Correspondences for Deformable Object Tracking." In: *Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2784–2791.

B. Rinner, L. Esterle, J. Simonjan, **G. Nebehay**, R. Pflugfelder, G. Fernandez, P. R. Lewis. "Self-Aware and Self-Expressive Camera Networks." In: *Computer* 48.7 (2015), pp. 21–28.

M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Häger, **G. Nebehay**, R. Pflugfelder. "The Visual Object Tracking VOT2015 challenge results." In: *Workshop on the VOT2015 Visual Object Tracking Challenge*. 2015.

M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, **G. Nebehay**, F. Porikli, L. Cehovin. "A Novel Performance Evaluation Methodology for Single-Target Trackers." In: *Transactions on Pattern Analysis and Machine Intelligence* (2016). To appear.

Bibliography

A. Adam, E. Rivlin, and I. Shimshoni. "Robust Fragments-based Tracking using the Integral Histogram." In: *Conference on Computer Vision and Pattern Recognition*. 2006, pp. 798–805.

A. Alahi, R. Ortiz, and P. Vandergheynst. "FREAK: Fast Retina Keypoint." In: *Conference on Computer Vision and Pattern Recognition*. 2012, pp. 510–517.

A. Andriyenko, K. Schindler, and S. Roth. "Discrete-continuous optimization for multi-target tracking." In: *Conference on Computer Vision and Pattern Recognition*. 2012, pp. 1926–1933.

S. Avidan. "Support vector tracking." In: *Transactions on Pattern Analysis and Machine Intelligence* 26.8 (2004), pp. 1064–1072.

B. Babenko, M.-H. Yang, and S. Belongie. "Robust Object Tracking with Online Multiple Instance Learning." In: *Transactions on Pattern Analysis and Machine Intelligence* 33.8 (2011), pp. 1619–1632.

B. Babenko, M.-H. Yang, and S. Belongie. "Visual tracking with online Multiple Instance Learning." In: *Conference on Computer Vision and Pattern Recognition*. 2009, pp. 983–990.

D. H. Ballard. "Generalizing the Hough transform to detect arbitrary shapes." In: *International Journal of Pattern Recognition and Artificial Intelligence* 13.2 (1981), pp. 111–122.

J. L. Barron, D. J. Fleet, S. S. Beauchemin, and T. A. Burkitt. "Performance Of Optical Flow Techniques." In: *International Journal of Computer Vision* 12.1 (1994), pp. 43–77.

H. Bay, T. Tuytelaars, and L. Van Gool. "SURF: Speeded Up Robust Features." In: *European Conference on Computer Vision*. 2006, pp. 404–417.

G. E. P. Box. "Robustness in the strategy of scientific model building." In: *Robustness in Statistics*. 1979, pp. 201–236.

G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. 2008.

L. Breiman. "Random Forests." In: *Machine Learning* 45.1 (2001), pp. 5–32.

M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. "BRIEF: Computing a Local Binary Descriptor Very Fast." In: *Transactions on Pattern Analysis and Machine Intelligence* 34.7 (2012), pp. 1281–1298.

M. Calonder, V. Lepetit, C. Strecha, and P. Fua. "BRIEF: Binary Robust Independent Elementary Features." In: *European Conference on Computer Vision*. 2010, pp. 778–792.

K. Cannons. *A review of visual tracking*. Tech. rep. CSE-2008-07. Department of Computer Science Engineering, York University, Toronto, Canada, 2008.

L. Čehovin, M. Kristan, and A. Leonardis. "An adaptive coupled-layer visual model for robust visual tracking." In: *International Conference on Computer Vision*. 2011, pp. 1363–1370.

L. Čehovin, A. Leonardis, and M. Kristan. "Visual Object Tracking Performance Measures Revisited." In: *Transactions on Image Processing* 25.3 (2016), pp. 1261–1274.

O. Chapelle, B. Schölkopf, and A. Zien, eds. *Semi-Supervised Learning*. 2006.

M. Cho, J. Lee, and J. Lee. "Feature correspondence and deformable object matching via agglomerative correspondence clustering." In: *International Conference on Computer Vision*. 2009, pp. 1280–1287.

R. T. Collins, Y. Liu, and M. Leordeanu. "Online Selection of Discriminative Tracking Features." In: *Transactions on Pattern Analysis and Machine Intelligence* 27.10 (2005), pp. 1631–1643.

D. Comaniciu, V. Ramesh, and P. Meer. "Real-time tracking of non-rigid objects using mean shift." In: *Conference on Computer Vision and Pattern Recognition*. 2000, pp. 142–149.

C. Cortes and V. Vapnik. "Support-vector networks." In: *Machine Learning* 20.3 (1995), pp. 273–297.

N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection." In: *Conference on Computer Vision and Pattern Recognition*. 2005, pp. 886–893.

M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. "Accurate Scale Estimation for Robust Visual Tracking." In: *British Machine Vision Conference*. 2014.

R. O. Duda and P. E. Hart. "Use of the Hough Transformation to Detect Lines and Curves in Pictures." In: *Communications of the ACM* 15.1 (1972), pp. 11–15.

S. Duffner and C. Garcia. "PixelTrack: a fast adaptive algorithm for tracking non-rigid objects." In: *International Conference on Computer Vision*. 2013, pp. 2480–2487.

M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. "The Pascal Visual Object Classes (VOC) Challenge." In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338.

L. Fei-Fei, R. Fergus, and P. Perona. "One-shot learning of object categories." In: *Transactions on Pattern Analysis and Machine Intelligence* 28.4 (2006), pp. 594–611.

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. "Object Detection with Discriminatively Trained Part-Based Models." In: *Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010), pp. 1627–1645.

P. F. Felzenszwalb and D. P. Huttenlocher. "Pictorial Structures for Object Recognition." In: *International Journal of Computer Vision* 61.1 (2005), pp. 55–79.

R. Fergus, P. Perona, and A. Zisserman. "A sparse object category model for efficient learning and exhaustive recognition." In: *Conference on Computer Vision and Pattern Recognition*. 2005, 380–387 vol. 1.

R. Fergus, P. Perona, and A. Zisserman. "Object class recognition by unsupervised scale-invariant learning." In: *Conference on Computer Vision and Pattern Recognition*. 2003, pp. 264–271.

J. Ferryman and A. Ellis. "PETS2010: Dataset and Challenge." In: *International Conference on Advanced Video and Signal-based Surveillance*. 2010, pp. 143–150.

M. A. Fischler and R. C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography." In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

M. A. Fischler and R. A. Elschlager. "The Representation and Matching of Pictorial Structures." In: *Transactions on Computers* C-22.1 (1973), pp. 67–92.

J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. "Hough Forests for Object Detection, Tracking, and Action Recognition." In: *Transactions on Pattern Analysis and Machine Intelligence* 33.11 (2011), pp. 2188–2202.

D. M. Gavrila. "The Visual Analysis of Human Movement: A Survey." In: *Computer Vision and Image Understanding* 73.1 (1999), pp. 82–98.

M. Godec, P. M. Roth, and H. Bischof. "Hough-based tracking of non-rigid objects." In: *International Conference on Computer Vision*. 2011, pp. 81–88.

H. Grabner and H. Bischof. "On-line Boosting and Vision." In: *Conference on Computer Vision and Pattern Recognition*. 2006, pp. 260–267.

H. Grabner, C. Leistner, and H. Bischof. "Semi-supervised On-Line Boosting for Robust Tracking." In: *European Conference on Computer Vision*. 2008, pp. 234–247.

E. Graether and F. Mueller. "Joggobot: A Flying Robot As Jogging Companion." In: *Human Factors in Computing Systems*. 2012, pp. 1063–1066.

S. Grossberg. "Competitive learning: From interactive activation to adaptive resonance." In: *Cognitive Science* 11.1 (1987), pp. 23–63.

S. Hare, A. Saffari, and P. H. S. Torr. "Struck: Structured output tracking with kernels." In: *International Conference on Computer Vision*. 2011, pp. 263–270.

C. Harris and M. Stephens. "A Combined Corner and Edge Detector." In: *Alvey Vision Conference*. 1988, pp. 147–151.

B. Hemery, H. Laurent, and C. Rosenberger. "Comparative study of metrics for evaluation of object localisation by bounding boxes." In: *International Conference on Image and Graphics*. 2007, pp. 459–464.

J. Henriques, R. Caseiro, P. Martins, and J. Batista. "Exploiting the Circulant Structure of Tracking-by-Detection with Kernels." In: *European Conference on Computer Vision*. 2012, pp. 702–715.

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

Intel Corporation. *Intel SSE4 Programming Reference*. 2007.

M. Isard and A. Blake. "CONDENSATION - Conditional Density Propagation for Visual Tracking." In: *International Journal of Computer Vision* 29.1 (1998), pp. 5–28.

P. Jaccard. "The Distribution of the Flora in the Alpine Zone." In: *New Phytologist* 11.2 (1912), pp. 37–50.

Z. Kalal, K. Mikolajczyk, and J. Matas. "Forward-Backward Error: Automatic Detection of Tracking Failures." In: *International Conference on Pattern Recognition*. 2010, pp. 23–26.

Z. Kalal, J. Matas, and K. Mikolajczyk. "P-N learning: Bootstrapping binary classifiers by structural constraints." In: *Conference on Computer Vision and Pattern Recognition*. 2010, pp. 49–56.

Z. Kalal, K. Mikolajczyk, and J. Matas. "Tracking-Learning-Detection." In: *Transactions on Pattern Analysis and Machine Intelligence* 34.7 (2012), pp. 1409–1422.

R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems." In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45.

D. A. Klein, D. Schulz, S. Frintrop, and A. B. Cremers. "Adaptive real-time video-tracking for arbitrary objects." In: *International Conference on Intelligent Robots and Systems*. 2010, pp. 772–777.

D. Koller, J. Weber, and J. Malik. "Robust multiple car tracking with occlusion reasoning." In: *European Conference on Computer Vision*. 1994, pp. 189–196.

M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Häger, G. Nebehay, and R. Pflugfelder. "The Visual Object Tracking VOT2015 challenge results." In: *Workshop on the VOT2015 Visual Object Tracking Challenge*. 2015.

M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Cehovin, G. Nebehay, T. Vojir, and G. Fernandez. "The Visual Object Tracking VOT2014 challenge results." In: *Workshop on the VOT2014 Visual Object Tracking Challenge*. 2014, pp. 191–217.

M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Cehovin. "A Novel Performance Evaluation Methodology for Single-Target Trackers." In: *Transactions on Pattern Analysis and Machine Intelligence* (2016). To appear.

M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, F. Gustavo, and T. Vojir. "The Visual Object Tracking VOT2013 challenge results." In: *Workshop on the VOT2013 Visual Object Tracking Challenge*. 2013, pp. 98–111.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Conference on Neural Information Processing Systems*. 2012, pp. 1097–1105.

H. W. Kuhn and B. Yaw. "The Hungarian method for the assignment problem." In: *Naval Research Logistics Quarterly* 2.1 (1955), pp. 83–97.

J. Kwon and K. M. Lee. "Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive Basin Hopping Monte Carlo sampling." In: *Conference on Computer Vision and Pattern Recognition*. 2009, pp. 1208–1215.

A. Lehmann, B. Leibe, and L. Gool. "Fast PRISM: Branch and Bound Hough Transform for Object Class Detection." In: *International Journal of Computer Vision* 94.2 (2011), pp. 175–197.

B. Leibe, A. Leonardis, and B. Schiele. "Robust Object Detection with Interleaved Categorization and Segmentation." In: *International Journal of Computer Vision* 77.1-3 (2008), pp. 259–289.

V. Lepetit, P. Lagger, and P. Fua. "Randomized Trees for Real-Time Keypoint Recognition." In: *Conference on Computer Vision and Pattern Recognition*. 2005, pp. 775–781.

V. Lepetit and P. Fua. "Monocular model-based 3D tracking of rigid objects." In: *Foundations and Trends in Computer Graphics and Vision* 1.1 (2005), pp. 1–89.

S. Leutenegger, M. Chli, and R. Y. Siegwart. "BRISK: Binary Robust invariant scalable keypoints." In: *International Conference on Computer Vision*. 2011, pp. 2548–2555.

H. Li, Y. Li, and F. Porikli. "DeepTrack: Learning Discriminative Feature Representations by Convolutional Neural Networks for Visual Tracking." In: *British Machine Vision Conference*. 2014.

X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. Van Den Hengel. "A Survey of Appearance Models in Visual Object Tracking." In: *Transactions on Intelligent Systems and Technology* 4.4 (2013), pp. 1–48.

T. List, J. Bins, J. Vazquez, and R. B. Fisher. "Performance evaluating the evaluator." In: *International Workshop on Performance Evaluation of Tracking and Surveillance*. 2005, pp. 129–136.

D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints." In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

B. D. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision." In: *International Joint Conference on Artificial Intelligence*. 1981, pp. 674–679.

E. Maggio and A. Cavallaro. *Video Tracking: Theory and Practice*. 2011.

M. E. Maresca and A. Petrosino. "MATRIOSKA: A Multi-level Approach to Fast Tracking by Learning." In: *International Conference on Image Analysis and Processing*. 2013, pp. 419–428.

L. Matthews, T. Ishikawa, and S. Baker. “The template update problem.” In: *Transactions on Pattern Analysis and Machine Intelligence* 26.6 (2004), pp. 810–815.

H. P. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. Tech. rep. CMU-RI-TR-3. Carnegie-Mellon University, 1980.

D. Müllner. “fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python.” In: *Journal of Statistical Software* 53.1 (2013), pp. 1–18.

G. Nebehay. “Robust Object Tracking Based on Tracking-Learning-Detection.” Master’s Thesis. Faculty of Informatics, TU Vienna, 2012.

S. S. M. Nejhum, J. Ho, and M.-H. Yang. “Visual tracking with histograms and articulating blocks.” In: *Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8.

M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. “Fast Keypoint Recognition Using Random Ferns.” In: *Transactions on Pattern Analysis and Machine Intelligence* 32.3 (2010), pp. 448–461.

M. Özuysal, P. Fua, and V. Lepetit. “Fast Keypoint Recognition in Ten Lines of Code.” In: *Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8.

F. Pernici and A. Del Bimbo. “Object Tracking by Oversampling Local Features.” In: *Transactions on Pattern Analysis and Machine Intelligence* 36.12 (2014), pp. 2538–2551.

B. Rinner, L. Esterle, J. Simonjan, G. Nebehay, R. Pflugfelder, G. Fernandez, and P. R. Lewis. “Self-Aware and Self-Expressive Camera Networks.” In: *Computer* 48.7 (2015), pp. 21–28.

D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. “Incremental Learning for Robust Visual Tracking.” In: *International Journal of Computer Vision* 77.1 (2008), pp. 125–141.

E. Rosten, R. Porter, and T. Drummond. “Faster and Better: A Machine Learning Approach to Corner Detection.” In: *Transactions on Pattern Analysis and Machine Intelligence* 32.1 (2010), pp. 105–119.

E. Rosten and T. Drummond. “Fusing Points and Lines for High Performance Tracking.” In: *International Conference on Computer Vision*. 2005, pp. 1508–1515.

E. Rosten and T. Drummond. “Machine Learning for High-Speed Corner Detection.” In: *European Conference on Computer Vision*. 2006, pp. 430–443.

C. Rother, V. Kolmogorov, and A. Blake. “"GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts.” In: *Transactions on Graphics* 23.3 (2004), pp. 309–314.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. “ORB: An efficient alternative to SIFT or SURF.” In: *International Conference on Computer Vision*. 2011, pp. 2564–2571.

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.” In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. "On-line Random Forests." In: *Workshop on On-line Computer Vision*. 2009, pp. 1393–1400.

J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. "PROST: Parallel robust online simple tracking." In: *Conference on Computer Vision and Pattern Recognition*. 2010, pp. 723–730.

G. Schindler and F. Dellaert. "A Rao-Blackwellized Parts-Constellation Tracker." In: *Dynamical Vision*. 2007, pp. 178–189.

J. Shi and C. Tomasi. "Good Features to Track." In: *Conference on Computer Vision and Pattern Recognition*. 1994, pp. 593–600.

J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. "Real-time human pose recognition in parts from single depth images." In: *Conference on Computer Vision and Pattern Recognition*. 2011, pp. 1297–1304.

A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. "Visual Tracking: An Experimental Survey." In: *Transactions on Pattern Analysis and Machine Intelligence* 36.7 (2014), pp. 1442–1468.

G. J. D. Smith. "Behind the screens: Examining constructions of deviance and informal practices among CCTV control room operators in the UK." In: *Surveillance & Society* 2.2/3 (2002), pp. 376–395.

C. Tomasi and T. Kanade. *Detection and Tracking of Point Features*. Tech. rep. CMU-CS-91-132. Carnegie Mellon University, 1991.

T. Tuytelaars and K. Mikolajczyk. "Local invariant feature detectors: a survey." In: *Foundations and Trends in Computer Graphics and Vision* 3 (2008), pp. 177–280.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. 1995.

M. Varma and D. Ray. "Learning The Discriminative Power-Invariance Trade-Off." In: *International Conference on Computer Vision*. 2007, pp. 1–8.

P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features." In: *Conference on Computer Vision and Pattern Recognition*. 2001, pp. 511–518.

T. Vojir and J. Matas. "The Enhanced Flock of Trackers." In: *Registration and Recognition in Images and Videos*. 2014, pp. 113–136.

W. Wang and R. Nevatia. "Robust Object Tracking Using Constellation Model with Superpixel." In: *Asian Conference on Computer Vision*. 2013, pp. 191–204.

N. Wax. "Signal-to-Noise Improvement and the Statistics of Track Populations." In: *Journal of Applied Physics* 26.5 (1955), pp. 586–595.

G. Welch and G. Bishop. *An Introduction to the Kalman Filter*. Tech. rep. 95-041. Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

Y. Wu, J. Lim, and M.-H. Yang. "Online Object Tracking: A Benchmark." In: *Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2411–2418.

R. Xu and D. Wunsch. "Survey of clustering algorithms." In: *Transactions on Neural Networks* 16.3 (2005), pp. 645–678.

A. Yilmaz, O. Javed, and M. Shah. "Object Tracking: A Survey." In: *Computing Surveys* 38.4 (2006), pp. 1–45.

Q. Yu, T. B. Dinh, and G. Medioni. "Online Tracking and Reacquisition Using Co-trained Generative and Discriminative Trackers." In: *European Conference on Computer Vision*. 2008, pp. 678–691.

K. Zhang, L. Zhang, and M.-H. Yang. "Real-Time Compressive Tracking." In: *European Conference on Computer Vision*. 2012, pp. 864–877.

L. Zhang and L. van der Maaten. "Structure Preserving Object Tracking." In: *Conference on Computer Vision and Pattern Recognition*. 2013, pp. 1838–1845.

T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: An Efficient Data Clustering Method for Very Large Databases." In: *International Conference on Management of Data*. 1996, pp. 103–114.

Q. Zhao, Z. Yang, and H. Tao. "Differential Earth Mover's Distance with Its Applications to Visual Tracking." In: *Transactions on Pattern Analysis and Machine Intelligence* 32.2 (2010), pp. 274–287.

W. Zhong, H. Lu, and M.-H. Yang. "Robust object tracking via sparsity-based collaborative model." In: *Conference on Computer Vision and Pattern Recognition*. 2012, pp. 1838–1845.