Particle Filter-based Tracking to Handle Persistent and Complex Occlusions and Imitate Arbitrary Black-box Trackers

Kourosh Meshgi

A thesis presented for the degree of
Doctor of Philosophy

Graduate School of Informatics
Kyoto University
Kyoto, Japan

26 August 2015

Particle Filter-based Tracking to Handle Persistent and Complex Occlusions and Imitate Arbitrary Black-box Trackers

— Kourosh Meshgi

Abstract

Occlusions, one of the most challenging problems in visual tracking, degrade the performance of many trackers significantly. Taking various spatial and temporal forms, occlusions have not been modeled completely yet. State-of-the-art solutions fail to handle persistent and complex occlusions, and mostly address partial or temporal occlusions. Additionally, the solutions around these problems are not unified, and researchers limit their solutions to a tiny portion of the problem. Despite the large number of studies of handling occlusion, only a few of them have actually studied the occlusion phenomenon itself and devised solutions for occlusion detection and reasoning. Any comprehensive study over different approaches of occlusion handling is deemed missing.

To address this shortcoming, this study first presents a comprehensive review on the literature. The occlusion problem is defined, its challenges are described, and several research directions to handle it are distinguished. Next, the state-of-the-art solutions and designs in each research direction are described, discussed and compared, and the strength and weakness of them are clarified. This study facilitates the design of further robust trackers to be built upon previous approaches efficiently.

Next, we propose a tracker to detect emergent occlusions, address difficult occlusion scenarios, and perform rapid target recovery after occlusion. This novel method builds upon a particle filter tracker and significantly improves its resilience against various types of occlusions, including persistent and complex occlusions. The objective of this tracker is to extend particles to include an occlusion state, switch observation and motion models based on the occlusion state, weight the observation based on the observation confidence, and employ robust feature fusion. Applied on the Princeton RGBD tracking benchmark, our tracker achieves superior performance and beats the state-of-the-art trackers in terms of tracking performance and speed.

While designing this tracker, a solid understanding about the occlusion is built, leading to proposal of a novel occlusion mask that enhances the observation of any appearance-based tracker, and improves its accuracy significantly during occlusions. To detect occlusions, the proposed method learns the occlusion patterns based on the data collected during tracking with and without occlusions. The obtained mask prevents the observation to be contaminated by data from occluder or background, and works well along with a general occlusion handling scheme in the main tracker.

Another part of this study strives to find the optimal feature set to maximize the tracker performance in various tracking situations even under occlusion. A series of experiments lead to the finding that most of the trackers can be imitated by a unified framework with interchangeable features and their corresponding weights. This method minimizes the tracking mismatches between the target tracker and a tracker that imitates the target tracker. This is made possible by employing dropout mechanism to maximize the similarity between the two trackers above while preventing the solution from overfitting to the limited training data. This mimicking approach is able to shadow many of the state-of-the-art trackers, while it sheds light on the possible inner mechanism of black-box trackers. The imitating tracker also outperforms its multiple target trackers by using their collective tracking knowledge during its training.

In summary, this study proposed an occlusion-aware framework to predict and manage various types of occlusions, which was found to enhance the observation quality to be used by arbitrary appearance-based trackers. It also proposed a unified framework to imitate an arbitrary tracker (even humans) by selecting and tuning the appropriate features to shed light on the feature design procedure. Handling these two issues is crucial for designing visual trackers working efficiently and robustly in real environments.

Dedication

This thesis work is dedicated to my love, Maryam, who has been a constant source of support and encouragement during the challenges of graduate school and living in a foreign country. I am truly thankful for having you in my life.

To my dearest sister, Parmis, who always believed that I can do anything she puts her mind to.

To my unique grandmother, Farkhondeh, who symbolized the best mixture of affection and strength.

To my extended family, Marjan, Mohammad, Sina and Faraz whose heartful support is a lifetime blessing.

And to the best parents in the world, Mandana and Mehdi, who have always loved me unconditionally and whose good examples have taught me to work hard for the things that I aspire to achieve.

Acknowledgments

This project would not have been possible without the support of many people. My principal supervisor, Professor Shin Ishii of Kyoto University, has been a significant presence in my life. His careful supervision is a true gift for the conduct of this study. His critical comments and suggestions on each piece of writing I made have strengthened this study significantly. I will always be thankful for his persistent guidance and encouragement.

I am grateful to Dr. Shigeyuki Oba and Dr. Shin-ichi Maeda, for keeping their hearts and office doors open to me whenever I want a talk. Their wisdom, knowledge and professional experience has been beneficial to this study in many aspects. I wish to thank them for their countless hours of reflecting, reading, encouraging, and most of all patience throughout the entire process. It has been an honor to work with both of them during my candidacy.

Also thanks to my committee members, Professor Toshiharu Sugie and Professor Toshiyuki Ohtsuka, who offered guidance and support and dedicate their valuable time to revise this manuscript. Thanks to the Ministry of Education, Culture, Sports, Science, and Technology (MEXT) of Japan for awarding me a generous *monbukagakusho* scholarship and to Professor Michiyuki Matsuda who supported me with a researcher position in *seimei doutai* project[1], providing me with the financial means to complete this project. In addition, I want to thank Mr. Tomohiro Nakamichi, the director of R & D division in 3D MEDiA Co. Ltd., for his support and understanding to handle the parallel work-study lifestyle close to end of my Ph.D. course.

Thanks to my numerous colleagues and friends, Shahram Kalantari, Abolfazl Nadi, Behnam Ghalei, Somayeh Niluyal, Alireza Jalili, Henrik Skibbe, Naoki Honda, Yohei Kondo, Masayuki Kouno, Hiroshi Moriyuka and Konstantine Torgasin who endured this long process with me, always offering support and encouragement. I am truly grateful to my first and most influential academic supervisor, Dr. Saeed Shiry Ghidary, for every single advice he gave me. There is no day in academia that I don't remember his words.

I am grateful for the assistance given by Sam Hare and Amir Saffari for their technical assistance with *STRUCK* tracker and to Andreas Zweng for providing the `PETS2009` annotated dataset.

Finally, I would like to thank my family, Mandana, Mehdi, Farkhondeh and the dearest Parmis for the support they provided me through my entire life. A special thanks to Marjan and Mohammad, my aunt and uncle-in-law whose love and support made it possible to overcome the hardship of establishing a life in Japan, and to my cousins Sina and Faraz who made this process a lot more fun. In particular, I must acknowledge my angle and girlfriend, Maryam, without whose love, encouragement, wisdom and assistance, I would not have finished this thesis.

[1]Platform for Dynamic Approaches to Living Systems from MEXT.

Contents

List of Algorithms

List of Tables

List of Figures

1
Introduction

"One never notices what has been done. One can only see what remains to be done."

— Marie Curie

Visual tracking is increasingly demanded in various applications ranging from human-computer interfaces, to crowd analysis, video processing, surveillance, automation, and medical purposes. This task involves keeping track of the spatial and temporal changes of one or multiple target(s) in a video sequence.

Surveillance is undoubtedly the most popular application of visual tracking, ranging from public security (e.g. detecting suspicious behavior and unattended objects) to crowd analysis (tracking, detection, counting, interaction analysis), public transport, congestion handling, tourism, and retail. Moreover, tracking applications in human safety (e.g., advanced driver assistant systems, pool underwater monitoring), entertainment (e.g., sports, animation, games) and medical imaging directly improves the quality of everyday life. Advanced human-computer interaction has been made possible using visual tracking as human behavior analysis (action, gesture, expression, motion analysis, etc) can be interpreted by machine and on the other hand, machines open their world to humans with virtual and augmented reality environments. Object tracking is now a key player in video communication and compression, and semantic video analysis (story extraction, frame grouping, retrieval) is progressing massively thanks to tracking technologies. In the current robotics era, tasks such as robot navigation, motion path planning and industrial inspection are build upon tracking target objects. Many of the object tracking applications tend to understand the world and events happening in it (e.g., structure from motion, estimation of coherent motion trajectories, support object segmentation, interpret temporal behavior of objects, super resolution). Higher level of understanding is achieved by combining certain lower level computer vision building blocks including object tracking, occlusion handling, and unusual motion detection.

Visual tracking is about keeping track of the spatial and temporal changes of one or more targets in a video sequence. Numerous surveys of visual object tracking have been published to investigate the state-of-the-art tracking algorithms from various aspects. Yilmaz et al. [1] provided a good frame of reference to review the literature, describing tracking methodologies, features and data association for general object tracking. Cannons [2] focused on low-level tracking techniques using different features or statistical learning methods for target representation. Several more specific surveys and benchmarks provide deep insights into the trends in designing trackers and their performance for different tasks. Li et al. [3] focused on detailed review of the existing 2D appearance models. From an algorithmic view, a well-designed taxonomy is presented in [4], in which all algorithms are partitioned into 35 different categories. Detection and tracking humans, dominates a big portion of the literature and consequently rich surveys are available for either pedestrian detection [5], human behavior recognition [6] and crowd analysis [7]. Many trackers are focusing on more specific targets such as sport players which are explored and discussed in [8].

By a close look at tracking domain, many issues are identified that make robust visual tracking a challenging task. Many researches attempted to provide a complete list of challenges mentioned in the field and categorize them. Table B.1 elaborated a comprehensive categorization of challenges mentioned in the literature on the basis of their sources.

When applied to video sequences in real-life situations, trackers should cope with numerous difficulties caused by these challenges. Numerous studies have been conducted to address these issues. Various adaptive appearance models are proposed [9–14], abrupt or significant motions are investigated [15–17], irregular scale changes have been focused [18], target/non-target confusion problem has been investigated [19], long-term tracking were aimed [20] and background clutter is studied [21, 22].

Trackers usually focus on one or a few of them at a time, yet no ultimate solution has been found. For example a successful tracker in handling illumination variations may has difficulty in coping with appearance changes cause by viewpoint variations. Many trackers use motion prediction to better anticipate object motion, but fail in tracking bouncing objects. Trackers with detailed models about the object appearance may fail on an articulated motion [23]. However, large-scale benchmarks [23–26] suggests that *TLD* [20], *STRUCK* [27], *MIL* [12], *FBT* [28] and *SCM* [11] have superior overall performance dealing with different challenges.

1.1 Problem Domain

Occlusion, one of the biggest challenges of visual tracking, impedes many trackers by corrupting observations, decaying the template accuracy, or introducing distracting occluders to the tracker.

Occlusion happens when a portion (or the whole) of the target disappears from the observed scene due to obstruction of the camera's line-of-sight to the target (Table B.1). This phenomenon appears due to numerous reasons and is frequently seen in real world video sequences. Complex interactions between objects, large displacement, shadow casts and dense crowds are instances of the cases in which temporal and/or spatial occlusions may occur. There are many difficult problems that trackers should address to handle occlusion completely, such as appearance change during occlusion, persistent occlusions, re-emergence of occluded object, emergence of the initially-occluded object in the middle of the scene, temporal silhouette (merging) of multiple objects, split observation of single objects, and fragmented trajectory of target objects.

In this study, *offline tracking* is excluded, and we focus on *causal tracking*, in which objects are tracked in a frame-by-frame basis. The term *causal tracking* have been coined for the tracker that do not pass through observations of different times several times, instead they only use the last observations and store the necessary information into their specialized data structures. *Online tracker* are those trackers that satisfy the real-time processing requirements (e.g., processing several frames per second). This set of trackers are mostly causal trackers, as the process of several frames in each time point is computationally demanding and usually defies the real-time processing limits.

Occlusion reasoning, which is another interesting sub-problem of occlusions in visual tracking, is out of the scope of this study. It is the process of determining the occlusion relationship between multiple targets and their relative depth order explicitly, with a particular interest in localizing the targets accurately. Layer presentation [29–31], competitive pixel assignment [32, 33], probability maps [34], and simultaneous segmenting-ordering-tracking [35] are the most successful trends in multi-target tracking under occlusions. This study briefly review the trends in occlusion reasoning for the sake of completeness, however, it narrows its scope to single target tracking.

1.2 Organization of This Study

This study first reports on the trending literature of occlusion handling in visual tracking (Chapter 2). We first explore the visual tracking realm and pinpoint the necessity of dedicated attention to the occlusion problem. Our comprehensive study suggests that although occlusion detection facilitated tracking impressively, it has been largely ignored. The literature further showed that the mainstream of the research is gathered around human tracking and crowd analysis. We then present a novel taxonomy of types of occlusion and challenges arising from it, during and after the emergence of an occlusion. Next, we focus on an investigation of the approaches to handle the occlusion in the frame-by-frame basis. Literature analysis reveals that researchers examined every aspect of a tracker design that is hypothesized as beneficial in the robust tracking under occlusion. State-of-the-art solutions identified in the literature involves by is not limited to various camera settings, simplifying assumptions, appearance and motion models, target state representations and observation models. We then discuss the identified trends and their merits/demerits.

Empowered by this extensive knowledge, we design a framework (Chapter 3) to tackle the more complicated forms of occlusion that is rarely addressed. Occlusions, which can be long lasting and of a wide variety, are often ignored or only partly addressed due to the difficulty of their treatments. We propose an occlusion-aware particle filter framework that employs a probabilistic model with a latent variable representing an occlusion flag. The proposed framework prevents losing the target by prediction of emerging occlusions, updates the target template by shifting relevant information, expands the search area for an occluded target, and grants quick recovery of the target after occlusion. This algorithm employs multiple features from the color and depth domains to achieve robustness against illumination changes and clutter, and the probabilistic framework accommodates the desired feature fusion. This method is applied to the Princeton RGBD Tracking Dataset, and the performance of our method with different sets of features is compared with those of the state-of-the-art trackers. The results revealed that our method outperformed the existing RGB and RGBD trackers by successfully dealing with different types of occlusions.

In a quest to find better features to incorporate in our proposed tracker, we observe that a *multi-cue particle filter tracker* start to behave like some of the popular trackers in certain scenarios. We start to ask ourselves if it is possible to imitate different trackers with a unified framework. Consumed by this question, we proposed a unified framework to imitate the behavior of any given tracker by selecting the features and tuning their weights automatically. However, the limited set of training video may not cover all of the potentials of target tracker, and the imitating tracker may learn a partial copy of the tracker. Furthermore, the training process were prohibitively long to examine all of the feature combinations to find the most effective one. In Chapter 4, we propose a non-linear feature fusion framework, "*MIMIC*" that imitates many popular trackers by mixing a pool of heterogeneous features. *MIMIC* framework consists of two known subtasks, feature selection and feature weight tuning. These subtasks, however, tended to be suffered from overfitting problem when the set of available videos are limited. To address this issue, we incorporate dropout algorithm into the framework, which grants the trained *MIMIC* tracker a high degree of generalization as demonstrated by extensive experiments in the manuscript. Imitating the behavior of an arbitrary visual tracking algorithm facilitates many higher level tasks such as tracker identification and efficient tracker-fusion. It is also useful for discovering the features in a black-box tracker, or learning from several trackers to form a super-tracker. We conduct several experiments to testify the effectiveness of the proposed method and promotes the application of this framework in different related tasks to visual tracking.

The idea of employing an occlusion mask were introduced in our proposed framework, but it was eclipsed by the power of the main principle of this framework. Later, we further investigated this technique to detect occlusions through learning the foreground probability distribution functions apart from the main idea (Chapter 5). In this approach, the target is divided into regular grid cells and the likelihood of occlusion is determined for each cell in a data-driven fashion. This technique generates an occlusion indicator for each of the cells. By learning corresponding distributions of this indicator for each cell, using a diverse set of videos and targets, we obtain a set of occlusion probability distributions that is universally applicable to any video or object. By assigning an occlusion likelihood to different cells of an observation (i.e., by creating an occlusion mask), our proposed approach provides a confidence measure for different parts of input observations and can be coupled with many generic tracking methods. In this study, we adopt four particle filter-based trackers – *multi-cue PFT*, *IVT*, *L1T*, and *L1APG* – to examine the effectiveness of the proposed occlusion mask. The experiments show that utilizing the proposed occlusion mask undermines the erroneous parts of observation, allows for a more robust template update, and mitigates distraction by occluders. The method is then evaluated on a set of popular challenging videos. The quantitative results highlight the tracking accuracy improvements obtained by the proposed technique, whereas quantitative results demonstrate its success to perform tracking under different occlusion scenarios.

As we learned that the gridding is beneficial for object tracking by proposing the occlusion mask in Chapter 4, we pursue to the idea to verify its effectiveness in tracking apart from the complicated occlusion problem. Using color information in object tracking is a prudent choice, but the vast variety of choices and difficulties of obtaining a desirable stable result, unnerves many scholars. This make the color histograms an ideal topic to examine the effectiveness of the gridding scheme. Color histograms, as a compact and robust representation is frequently used in tracking while it suffers from lack of spatial information about colors. Besides, comparison and updating such histograms in a meaningful and efficient manner is challenging. In Chapter 6, we implement the idea of gridding for color histogram. Gridding grants specific statistical property to the histogram through a decomposition phase followed by a recombination stage. To verify the superiority of this technique, we combine the gridded HOC with modern similarity functions and model update techniques in RGB colorspace and perform a thorough benchmark. This comparison reveals that our proposed gridding in combination with almost all of the established similarity measures, enhances the tracking performance.

Finally, in Chapter 7 we discuss the proposed technique, elaborate the future research directions, and conclude this manuscript with a brief review of the proposed methods, their achievements, and the publications resulted from them.

The videos, charts, and other supplementary material related to the contents of this manuscript are publicly availabe in the authors webpage at the following URL: http://ishiilab.jp/member/meshgi-k/project-phd.html

1.3 List of Publications

- K. Meshgi and S. Ishii, “The state-of-the-art in handling occlusions for visual object tracking,” *IEICE Transactions on Information and Systems*, vol. E98-D, no. 7, pp. 1260–1274, 2015
- K. Meshgi and S. Ishii, “Expanding histogram of colors with gridding to improve tracking accuracy,” in *MVA’15*. IEEE, 2015, pp. 475–479

2
Tracking under Occlusion: an Inclusive Review

"Don't reinvent the wheel. Just realign it."
— Anthony J. D'Angelo

2.1 Introduction

In contrast with the wealth of literature in visual tracking, the occlusion problem has received little attention. Gabriel et al. [39] formalized the occlusion problem in terms of the notion of "blobs" (i.e., a group of objects) and proposed two ways to approach the problem: merge-split approach and straight-through approach. Recently Lee et al. [40] revisited the occlusion problem in object tracking, yet there is no clear understanding of the general occlusion problem [41]. Besides, the literature offers many diverse strategies, to which a comprehensive and systematic inspection seems necessary.

Our contributions in this chapter are summarized into three:

- a comprehensive review of state-of-the-art techniques for occlusion detection, reasoning, and handling;
- a formal definition of challenges in occlusion problems;
- and a new approach to attribute the occlusions based on extent (severity), duration and complexity.

Following this introduction, in section 2.2 the occlusion problem is defined and categorized, then challenges within it are described thoroughly. The ways in which occlusion is tackled in the literature are comprehensively studied in section 2.3, followed by section 2.4 that skims the procedure and material to evaluate tracker performances under occlusion. Following the substantial literature review, effective approaches to robust tracking under occlusion and future research directions are discussed in section 2.5.

2.2 Occlusion Taxonomy

Occlusion happens when observation of a target object (or key attributes to identify the target object) is not available in order for the camera to keep tracking the spatial location of the target while the target is still present in the designated scene [40]. From the camera viewpoint, several objects may be present in a single line of sight. Different depth ordering of target and distractor objects lead to partial or complete viewing obstruction of the target object [41].

2.2.1 Occlusion Categories

Early studies of visual tracking either ignored the occlusion problem or claimed to handle just partial occlusion owing to their robust design. Recent studies classify the occlusion into more general categories [34]:

- *Dynamic (inter-object) occlusion* is the outcome of overlapping with another object, which is closer to the camera;
- *Scene (background) occlusion* happens because of (still) objects inside the background model that are actually located closer to the camera;
- *Apparent occlusion* occurs because non-visible regions emerge due to pose and/or shape changes, silhouette motion, out-of-plane rotations, shadows, or self-occlusions.

Another type of occlusion may arise in multi-camera setups. An object is considered occluded between a time when it leaves the filed-of-view of a camera and another time when it enters the field-of-view of another camera or return to the previous one [40]. This type of occlusion is called *Blindspot Occlusion*. This categorization is based on the occluder class, which is unable to distinguish the performance of a tracker in different occlusion scenarios.

By attributing occlusion, the ability of a tracker to handle different types of occlusion can be assessed. We propose three intuitive attributes to describe each occlusion: extent, duration, and complexity.

- *Extent* of an occlusion is defined over the key features of the target. In partial occlusion, some of the key features of the target are hidden from the camera, while a full occlusion is the case that the target object is entirely invisible to the camera while knowing that the target is still in the scene.

- *Duration* of the occlusion can be short or long: Temporal or short occlusions happen frequently in an urban scene where the duration of occlusions are short and limited; Persistent or long occlusions, on the other hand, usually require dedicated treatments to fully employ the dynamics of the target. This is especially troublesome for generative model-based approaches which employ multiple hypotheses to find the target after occlusion. Here, a consistent bound between the definitions of long and short could be important. It is reasonable to set the border threshold as a portion of unoccluded frames in which the target was observed, as the tracker accumulates information about target (e.g., 8% of unoccluded frames).

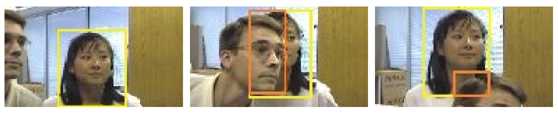
- Considering the *Complexity* of the occlusion, if one of the key characteristics (e.g., appearance, orientation, motion direction, size, number of blobs and distance from camera) of the target changes drastically during occlusion, the occlusion is complex, otherwise it is a simple one.

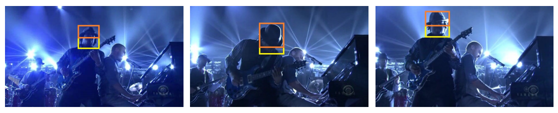
By combining these attributes, eight different occlusion kinds are characterized (Figure 2.1).

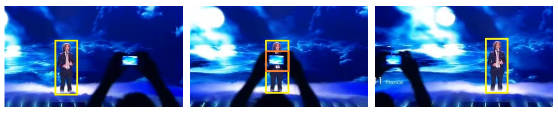
While easier occlusion cases (e.g., simple temporal partial occlusion) are handled by many recent trackers, more complicated ones (e.g., complex persistent full occlusion) are rarely handled (Figure 2.2), which emphasizes the importance of redirecting more attention to tackle the latter ones.

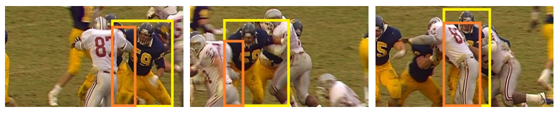
2.2.2 Occlusion Induced Challenges

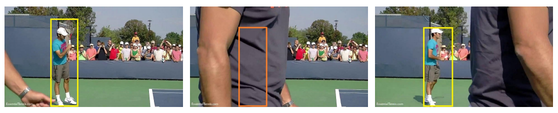
There are plenty of challenges which a tracker should deal with during and after an occlusion, some of which are less frequently attended in the literature.

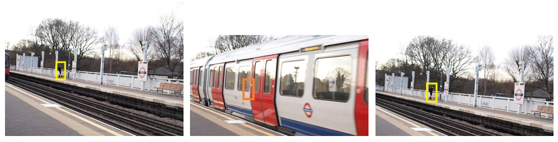

(a) PTS

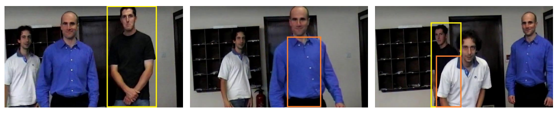

(b) PLS

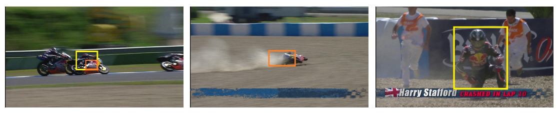

(c) PTC


(d) PLC


(e) FTS


(f) FLS


(g) FTC



(h) FLC

Figure 2.1: Examples of eight occlusion cases yielded from three binary attributes: extent (Partial/Full), duration (Temporal/Long), and complexity (Simple/Complex). Left and right panels illustrate before and after occlusion, respectively, while middle panel shows the scene during the occlusion. The yellow box indicates the target while orange parts show the occluded part of the target.

During an occlusion, due to missing information, trackers' ability to localize their targets becomes limited. Some applications require the *localization of the occluded target*. The trackers are thus required to provide an estimation of the target location when it is partly or completely invisible.

The next problem is known as *split & merge problem*, which is essentially the task of assigning foreground pixels to the objects when the objects do not have one-to-one correspondence to foreground blobs, either due to that a single object appears as multiply fragmented foreground blobs (split) or when multiple objects form a single foreground blob (merge) [42]. This problem arises in the approaches where foreground pixels are separated from background ones and then target objects should be identified in the foreground blobs. The merge case happens when objects are grouped, are placed close to each other, interact, or overlap each other in the camera line-of-sight. On the other hand, partial occlusion and accidental alignment (e.g., portions of a moving object are accidentally very similar to those of objects in the background, resulting in misdetection of actually moving pixels) can fragment a silhouette into temporally or spatially separated elements [43]. To enumerate possible outcomes of split and merge phenomena in a scene, the notion of object support is presented. Object supports are hypothetical object regions, parts of which may not be visible due to occlusion. The study in [41] classified possible occlusion states into seven (OC-7) by considering the spatial relations between the object support and the detected foreground.

In addition to the object-foreground relationships, *varying number of objects* in the scene increases the complexity of the tracking scenario. A good multi-target tracker would be able to detect changing numbers of objects in the scene –by adding or removing objects when appropriate– and also able to handle both occlusion and split events. To handle the emergence of new objects, which is known as the *birth problem*, trackers either use a global object detector, monitor possible object entry points [44], or initiate an object when an unidentified moving blob is detected. In the occlusion case, an object can appear in the middle of the scene as it separates from a group or emerges from an occlusion that has covered it since the beginning of the tracking session. Contrarily, an object may leave the scene, either for being hidden by an occluder or by joining a group, and hence becomes indistinguishable. This is called the *death problem*. To handle varying

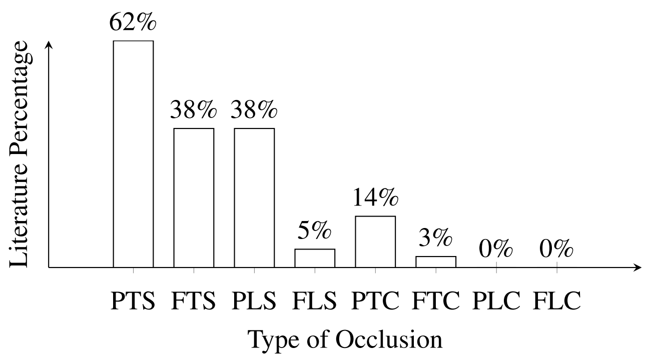


Figure 2.2: Statistics of trackers in two recent benchmarks [23] and [25]. These two benchmarks examined 36 distinct trackers in total including the most recent and well-established ones. Some of these trackers were unable to continue tracking even in case of minor occlusions.

number of objects requires a concrete strategy to address birth and death problems.

When several targets overlap each other, it is sometimes necessary to determine their order along the camera's line-of-sight, i.e., in the depth direction. The task of disseminating the objects in the depth order is called *occlusion reasoning*. Having depth information in the case of stereo cameras, multiple cameras and range-finders makes the task trivial, but in the case of single fixed camera the problem turns to a big challenge.

Early trackers assumed that the target appearance does not change considerably throughout the tracking (e.g. [44]), while in real-world scenarios this assumption is typically violated. Hence, it is crucial to update the model to account for appearance variation. Updating the template with the latest observation is vulnerable to partial observations, in which the data is partly missing, or contaminated with irrelevant data from occluder or background. Especially in the case of partial occlusions, the trackers which only adopt model update without considering the observation reliability will fail, since their updated templates are apart from the current target appearance (i.e., drifted away). This argument holds for learning trackers in which non-target or occluded samples hinder model learning of the correct target appearance. To handle this *model drift problem*, the model should be updated slowly or selectively, to keep the memory of target appearance; contrarily a faster reactive update scheme is required to cope with frequent target variations [34].

After the complete resolution of the occlusion, in a phase called occlusion recovery, the tracker is expected to recover the target once again. The target may have changed during the occlusion or appear somewhere other than expected in the scene. A common problem in the occlusion recovery of multi-target trackers is *identity loss*. If the target ID is a new one, it means that the tracker suffers from re-identification problem, while giving wrong ID to the tracker (identity switch) could indicate model drift and/or confusion problem. The *re-identification problem* corresponds to trackers' strategy to handle birth & death and model drift problems. *Confusion problem* in multi-object trackers arises when the targets are "identical", in the sense that the same model is used to describe each of the targets [45]. In such cases, the number of identity switches between them increases, so more discriminating feature or prior knowledge (such as estimated reappearance location) is required to resolve the problem. Furthermore, other problems such as unlikely trajectory for the similar targets [19], delayed recovery of target location and disturbed scale adaptation for the recovered objects can be included to this list. Figure 2.3 illustrates the scope of main problems induced by occlusion in tracking timeline.

Next section provides a rich overview of research directions to handle occlusion and related problems in visual tracking.

2.3 Handling Occlusions

Occlusions, if not handled, result in errors which may eventually cause the tracker to drift away from the target or lose it in the middle of tracking scenario. Some trackers assume controlled environments, in which several assumptions are satisfied. Such assumptions try to simplify the task, by introducing a new concept in tracking, or just by dealing with certain aspects of the occlusion. Instances of rather unrealistic assumptions are color-separability of objects [30, 34], smooth camera motion, stationary changing background, static target appearance, limited amount of spatial and temporal occlusion, limited interaction between objects, single class of objects in the scene, known birth/death time of objects or at least known number of them at each time [43]. There are other assumptions which facilitate tracking and hold in specific applications, such as stationary camera which eases background subtraction and sufficiently large objects to support strong appearance models. Moreover, there exist some assumptions derived from human visual system such as object permanence (i.e., partially or fully occluded objects, even though not observable, still exist in the close proximity of their occluders and move with them before they re-emerge) [30].

Occlusion handling is the task of minimizing the impact of occlusion on the tracking, which is achieved by granting robustness to the tracker against occlusion (passive handling) or preventing/compensating the disturbing effects of the occlusion (active handling). The occlusion handling has been more studied in discriminative tracking approaches, while generative models address the occlusion by definition since they maintain a large set of hypotheses, which by covering the target have a chance to survive occlusion and target recovery [34]. In this study, we present several directions identified in the literature to deal with occlusions.

2.3.1 Robust Representation

Robust representation has been proposed to solve the occlusion problem by using robust statistics [46–49], patch matching [50–53], spatially biased weights on target observations [54], codebook learning [55, 56], feature selection [57], and discriminative classifiers [20, 28]. Furthermore, many trackers tackle occlusions indirectly by statistical analysis [9, 10, 58].

Holistic templates made from raw intensity values are the most popular representations for tracking. Many researchers, however, have pursued more advanced templates to handle different conditions of tracking, especially occlusions. To better account for appearance changes and handling occlusions, many ideas were applied to trackers (Table 2.1). Learning appearances introduced a new turn into appearance modeling as the researchers attempted to incorporate more intelligence into the trackers. A good representative of this category is based on discriminative models in which a binary classifier is trained online to discriminate the target from background [59]. Utilizing robust features and their combination is another trend in the literature. A good discussion on feature fusion in Bayesian framework was provided in [37]. Additionally, the fusion of different

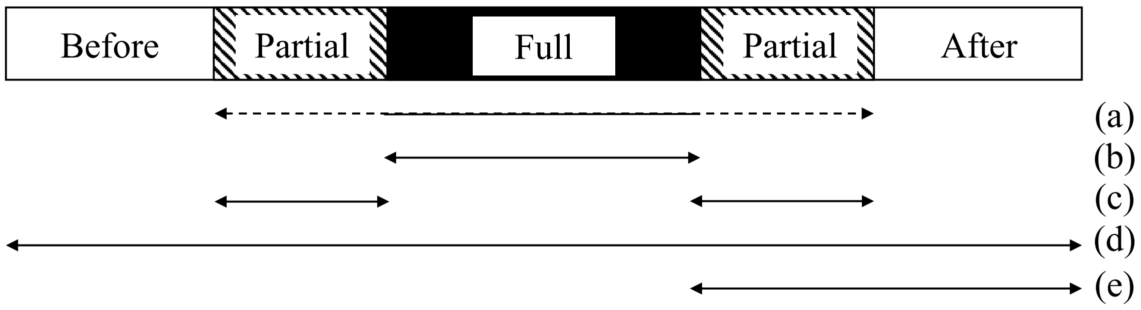


Figure 2.3: The scope of occlusion induced challenges: (a) occluded target localization (b) varying number of objects (c) split & merge problem, occlusion reasoning (d) model drift problem (e) identity loss/switch, confusion problem

Table 2.1: Robust representations to handle occlusions

Extension to Holistic Templates	Superpixels [66, 67]
	Local templates (patches) [52]
	Sparse representation [68, 69]
Learning Approaches	Integrate sample labeling into learning tracker [27]
	Codebook learning (using bag of features [55] or sparse representation [56])
	Feature selection [57]
	Discriminative classifiers (Learning from labeled and unlabeled data [70], incremental [10, 59], hough-forests [71])
Robust Features	Local features (Local templates, MSER, SIFT, SURF, corner features [3])
	Saliency detection features [72]
	Combining features (Multi-cue raw-pixel, spatial-color histogram, appearance and depth domains features [37])
	Using different feature sets [73]
System-level Fusion	Self-organizing models and integration methods [74]
	Fusion of multiple detectors using RJ-MCMC [75]
	Hybrid system of trackers and detectors
	Sampling over a pool of simple trackers [76]

cues from a number of detection and tracking algorithms have been used to produce more robust trackers. There are plenty of robust representations against occlusions such as active contours, wavelet-filter based and covariance matrix representations [46]. Moreover, subspace-based tracking approaches [10] handled appearance changes well. To better handle appearance variations, some approaches recently proposed integrating multiple representations [60]. A good discussion over appearance models can be found in [3].

Most of these appearance models are susceptible to contamination by long-term occlusions due to their blind update strategies [61]. Model update problem, however, can be efficiently managed by using leaky memories [62], online mixture model [58], dictionary updating [63], online boosting [64], and incremental subspace update [10]. Discriminative models, on the other hand, strongly depend on the sample collection part to make the gradually trained classifier more robust [12, 27, 65]. Despite the progress in appearance models, preventing model drift in adaptive models is still far from perfect. Besides, insufficient attention has been paid to contingency methods such as model drift recovery, e.g., [66].

2.3.2 Motion Models

Dexterous motion models improve the sampling accuracy for target candidates and promote occlusion handling. Estimating the state of the target has been proved useful in resolving issues of the occlusion especially in persistent or complex occlusions where the target is likely to continue changing while the model cannot be updated [23, 25]. Motion models enable trackers to fuse target motion with appearance model to continue predicting its state until it reappears.

If formulated in an *optimization* framework, the task is to find a plausible motion of the target which minimizes the distance of the path to the location of the target in consecutive frames while maintaining several constraints. If the objective function is differentiable with respect to motion parameters, gradient decent methods can locate the target efficiently [77, 78]. However, these objective functions are usually non-convex or non-linear [24]. Unless an explicit occlusion term is embedded into the energy function, this class of motion models cannot handle full occlusions, e.g., in [79]. Still this category can handle partial complex occlusions in the cost of heavy computation.

Generally targets can be found close to its previous location, thus a *uniform search* around the location will find the target efficiently [27]. Such explicit position search may lose the targets whose motions are unexpectedly fast. To alleviate this problem, probabilistic *Gaussian motion models* are utilized [10, 68]. By putting more weight on locations closer to previous location, this solution poses more bias on the target motion than uniform search, hence, fails easier in the case of fast and abrupt motions [23]. Since the search area is fixed in these schemes, they are unable to handle persistent occlusions.

Dense sampling as utilized in [12, 27, 64, 80] is one of the simplest solutions to handle large search space, but it suffers from high computational complexity. Hence, stochastic search algorithms have been widely used since they are relatively insensitive to local minima and computationally efficient. A natural choice of the sampling algorithm is the linear motion model, described by *Kalman filter*. Although prediction of motion as applied in [69] alleviated the fast motion problem, it is not applicable to general tracking easily. This is due to the fact that explicit motion modeling in complex scenes is difficult and not generalizable. Advanced Kalman filters can estimate trajectories in some cases of occlusion but degrade when the target objects, occlusion extents, or the distractors increase. Constant prediction of target location enables Kalman Filter motion models to handle full persistent and/or complex occlusions for motions with smooth and linear trajectory.

By handling non-linear non-Gaussian motions, *Particle filters* surpass Kalman filters, handle temporal occlusions, and hence have become very popular. Particle filters are also widely used in multi-target tracking, being enhanced by partitioned sampling [45], Adaboost learning module to differentiate targets, ensemble of particle filters, and MCMC-based particle filters. These schemes are only capable of handling partial temporal occlusions. Nevertheless, if the correlations among objects are not exploited, generic importance sampling becomes inefficient as the number of targets increases, and in addition the joint state representation leads to high computational cost. Moreover, particle filter-based trackers with insufficient number of samples cannot generate statistically significant modes leading to fails in tracking multiple targets. Particle filters combine information obtained by sampling with assumed motion patterns (e.g., constant velocity) so they suit partial or temporal full occlusion scenarios. Yet, persistent or complex occlusions impede particle filter trackers.

Parametric motion models also gained some popularity by enabling trackers to learn target motion patterns [53, 77, 81]. Motion estimation is approached by parametric motion models that utilize predictors such as linear regression techniques or parametric second order linear system with online parameter tuning [81]. More rich models describe rotation [27], scale [77], shear and affine or 2D projective deformations of the target [53]. They are especially useful for complex partial and self occlusions. Motion prediction governed by *optical flow* is an interesting alternative [82]. Another attractive idea is to simultaneously perform tracking and detection as in [70]. This approach is proved to be very successful in handling various forms of partial occlusions, either persistent or complex.

These motion models improve the accuracy of tracking, and are effective at handling full occlusions in which the object trajectory remains unchanged, provided that the model matches its trajectory prior to occlusion. Most real-world video sequences, however, do not follow such well-defined motion patterns. Additionally, these models lead the tracker to a permanent target loss if they cannot recover the target after it reappears. To enrich these models, researchers utilized context information [19, 83, 84] to predict the target's whereabouts, which may require tracking non-target objects or features. *Context information* assists trackers, especially when the target is fully occluded or leaves the image region. Information such as local visual information surrounding the target [19, 84] or auxiliary objects in the scene [83] is recognized to be useful in this regard. Such knowledge empowers the tracker to deal with persistent full occlusions [24], and solves the invisible object localization problem mentioned earlier. Using object permanence is another form of utilizing the context information in which, the tracker explicitly follows the occluder and limits the search space to its proximity, since the target is expected to emerge (if at all) from the boundaries of this occluder [30].

Motion estimation is not limited to the whole objects, as sub-object extensions can be found in the literature. Using a separate Kalman filter for every image patch [50] and tracking circular redundant image patches with mean-shift [41] are two good examples of such approaches. Complex temporal partial occlusion is effectively handled by those methods, while longer occlusions trouble trackers using this type of motion models. Although motion models estimate the target location and narrow down the whole scene to a smaller region of interest (ROI), errors in this prediction may result in losing the target permanently.

2.3.3 Foreground Tracking

Foreground trackers cast occlusion reasoning as a classical segmentation problem: classify foreground pixels into several sub-regions according to prior knowledge and track the targets based on segmented regions.

Motion-based trackers and appearance-based trackers are two families of such trackers. These methods track connected regions that roughly correspond to the 2D shape of objects based on their dynamic models [85]. The advantages of such models are their real-time applications, model-free description, and the large amount of information which is available to the tracker due to pixel level accuracy of the target description. These trackers try to associate every pixel to an object and are thus sensitive to occlusions.

In order to detect foreground pixels, the background is usually subtracted from the video using methods like temporal median filter, Gaussian mixture models, and codebooks. To simplify the task, some studies assumed a stationary camera or color separability between objects. The foreground is then segmented into its primary entities, "blobs". Each blob may contain more than two objects due to object proximity to one another, related occlusions and image noise, so that a blob may be composed of elements of one or more than two actual physical objects, which over time may shift from one observable blob to another [43]. An object, on the other hand, may be split into several blobs or merge together to form a bigger blob in various cases of occlusion (recall OC-7). Hence, the occlusion problem is reduced to an association problem of the blobs to the objects for which finding a unique solution in a real scene is challenging [86], due to (i) image changes; (ii) the presence of non-rigid or articulated objects and their non-uniform features; (iii) multiple moving objects, especially similar or crossing objects; (iv) ambiguous matches, e.g., one blob corresponds to several objects, when objects split or merge; (v) erroneous segmentation; and (vi) changing features.

Solving the association problem formed the core concentration of many studies. Finding an exclusive correspondence between different objects by using joint probabilistic data association filter was one of many attempts to formulate a solution to this problem. Occlusion handling is considered in later studies. People interaction was typically handled using Bayesian networks. The split and merge problem, however, has been the biggest challenge of this category of trackers. Searching for all possible changes of blobs leads to an expensive combinatorial search [87]. Besides, due to fragmentation, target identities are switched by object interactions. Additionally, if the number of objects is varying or unknown, an ambiguity arises when using generic models to track objects that may be fragmented or grouped. This phenomenon is known as fragment-object-group ambiguity and demands an estimation of the number of objects and association of foreground blobs with objects simultaneously.

Early responses to the fragmentation problem were to avoid it using a distance-based criterion to cluster blobs and track those near each other [88], which leads to loss of resolution. This solution, however, is not effective in scenes where objects have grossly different sizes, because of unwanted grouping of objects especially in densely populated scenes [43]. Allowing merge/splits while tracking cluster of pixels gives rise to another ambiguity in which objects are not distinguished from fragments/groups, and/or object IDs before and after group interaction cannot be associated.

In order to distinguish a split blob from a single target or a single merged blob from several targets in multi target tracking, [89] used analytically solvable particle filter, [32] used thresholding over blob sizes, and [30] utilized a constrained optimization formulation exploiting object permanence constraint (explained earlier), [90] utilized a traditional Bayesian multi-target tracker over virtual blobs, and [91] formulated the problem as a multiple association problem. Methods based on pixel level regions were also proposed in the literature, e.g., [34]. As mentioned, a good foreground tracker should handle various number of targets. Handling the birth and death problem may assist this task. Listing active and passive objects [41] and growing and shrinking motion regions are two typical solutions for this problem.

Advanced foreground trackers perform segmentation and tracking simultaneously by not using a separated background subtraction scheme [92, 93], yet they are unable to handle severe occlusions.

2.3.4 Model-based Tracking

Trackers of this class directly describe the target and attempt to track it explicitly throughout the scenario. In this case each blob contains only one object, and can be tracked individually without being merged in the possible event of occlusion. In order to describe the target, three methods dominate the literature: classification approaches, feature based techniques, and deformation models.

Trackers of the first category, known as tracking-by-detection approaches, contain a trained object detector or a generic object detector trained online during tracking. Such detectors are based on state-of-the-art machine learning techniques such as boosting [22, 64], semi-boosting [65], multi-instance boosting [12] and variations of support vector machines, e.g., [27, 94, 95].

Features which are almost invariant to the appearance change, pose or occlusion play a crucial role in robust tracking under occlusions. Haar-like features [64], histogram-based appearance features (e.g., HOG) [78, 96], histogram of relative optical flow (HOF) [97], features learned from deep learning models, depth, segmentation and motion [98] are involved in such successful features. Features are either chosen manually or automatically from a set of features [99].

Deformable models have advantage when tracking non-rigid objects, by employing high resolution prior knowledge where all motions and appearances of the model components are well-defined and by regarding occlusions as missing information in the process of tracking.

Generic detectors model the whole object in a single template. Such detectors assume that objects are fully visible so their performance degrades in the case of partial occlusions [94]. On the other hand, part-based models which mainly model the translational deformation of parts are typically more resilient against occlusions. Part-based deformable models such as [95] sum up the scores of part detectors, and the existence of an object in the input window is indicated with a relatively high total score. In this model, an occluded part may have very low score which ends up to a low total score. Therefore, some trackers rely on the detection score of the part to estimate its visibility [98], combining the responses of part detectors to form a joint likelihood model, or calculating a weight for each part based on its appearance difference from the background. In addition to known object parts, meaningless patches (fragments) are also tracked in [47] while independently moving entities are clustered probabilistically in an unsupervised fashion. The authors of [98] hypothesized that the key to successful detection of partially occluded humans is to utilize additional information about which body parts are occluded. Having such extra knowledge along with other information from motions, depth, and segmentation enables the tracker to compensate partial occlusion effectively; i.e., when the occluded parts are identified, their effect should be appropriately removed from the final combined score. Deformation score in combination with appearance score can also promote a more accurate tracking using part-based detectors [94, 95]. So far all studies shown above assumed independence among different object parts and hard-thresholded the detector score to determine visibility. A step toward more realistic assumptions was taken in [100] where an expert described the relationship between parts in a rule-based fashion, whose idea was later extended in [101] where the visibility relationships among parts were learnt systematically from training data using a discriminating deep model.

The recovery from occlusion in object trackers is straight forward. If the search area and target template perform well, the target is recovered immediately after it reappears. However, model drift reduces the discrimination power of the model so that more accurate search mechanism is required to compensate this artifact. Employments of context information, motion models, and auxiliary trackers are beneficial for empowering the tracker's search mechanism. To handle the model drift problem, forgetting memories for templates and sampling from non-occluded target appearance for learning module are two established solutions while advanced attempts to bring more robustness to template update have been made, e.g., [58].

2.3.5 Mode Switching Trackers

Every tracker has its own characteristic which suits for specific application and yet no dominant general purpose tracker is released by the community. Naturally, changing the circumstances may hinder tracking of some algorithms, while some others may work perfectly. For instance, a tracker may be good in handling variations in illumination, but may not necessarily be able to cope with appearance changes of the object caused by variations in object viewpoints. Also a tracker might predict motion to better anticipate its speed, but it may have difficulty in following bouncing objects. As an another example a tracker may make a detailed assumption of the appearance, but then may fail on an articulated object [23]. Balancing the trade-off between different trackers heavily depends on the task in hand and hence scenario conditions. One of the most disrupting changes in the environment is occlusion. Some researchers believe that by switching trackers, or adapting some tracker modules to new circumstances, better trackers can be constructed. Mode-switching trackers attempt to handle occlusions either by switching between trackers [76, 85, 102, 103] or within a tracker by switching different modules within a tracker [60, 81]. Whereas the former suffers from compatibility issues, the latter is promising for occlusion handling.

Switching between trackers have been studied in [85] where the tracker resorts to a particle filter tracker when the main part-based tracker undergoes some occlusion. In a study by Wu and Nevatia [102], a global part-based tracker switched to an individual mean-shift tracker for each part when the data association failed. Utilizing a mean-shift tracker in normal condition and switching to a particle filter in the case of poor performance constitute another successful solution [103]. Kwon and Lee [76] presented a switcher which chose only the required trackers, suitable for current condition of tracking, from a tracker pool.

Within trackers there are many modules which by operating differently cope better with the situation. Switching motion models and soft/hard switching between two sampling methods in a particle filter tracker as in [81] are good examples of such methods. These method are based on fixed switching criteria, which render them sensitive to parameter settings. To address this issue, [37] proposed an adaptive switching method which alters tracker dynamics, switches motion models and sampling methods, and recovers quickly from occlusion.

2.3.6 Multiple Camera Scenarios

Visual tracking with multiple cameras significantly reduces the challenges introduced by occlusion in different scenarios at the expense of simplicity of the tracking algorithm. In order to minimize the occlusion, cameras with face downward or omni-view (360°), ultra-wide bird-view cameras, multiple static cameras, stereo cameras, multiple automatically driven cameras, and even non-overlapping cameras have been used, each bringing its benefits and disadvantages into tracking [39, 40, 104–107].

With several cameras covering a scene from different viewpoints, a target that is invisible to one of the cameras may still be visible from other cameras, which reduces the probability of full occlusion. This observation also suggests that in multiview monitoring, the videos obtained from different cameras must be "fused" to handle occlusion [39]. Reasoning about occlusion relations between objects in such scenarios has been incorporated in several trackers using Bayesian networks.

2.3.7 Occlusion Detection

Despite its importance, occlusion detection is rarely addressed explicitly in the literature. It is understandable since the wide variety of occlusion scenarios makes it difficult to find a reliable occlusion detection metric. On top of that, not all of the occlusion detection methods are generic, and some of them is limited to specific application or tracker architecture (e.g., particle filter tracker). State-of-the-art occlusion detectors are based on the foreground to background ratio [34, 87], structured sparse learning [63] and training a patch-based occlusion classifier [61]. Depth data further provide additional information to spot occlusions [108]. Employing such detectors enables trackers to intelligently update their templates [62].

In the case of foreground trackers, the ratio between the number of observable points (the one in the foreground blob) and points of the appearance model provides a clue about occlusion, as the low value of this ratio indicates an occlusion [34, 87]. Large deviations from appearance model [58], heuristic criteria on proximity and size changes of blobs, robust region description near keypoints, and thresholding the sum of likelihoods in particle filters [109] are good methods of detecting occlusions. More sophisticated method such as using structured sparse learning [63] or learning occlusions by likelihood [61] are modern solutions which are promising. Using histogram of depths to detect occlusions [108] is yet another recent approach which emerged after the advent of cheap range sensors.

Occlusion detection enables trackers to prepare for eminent occlusions, to change their state during occlusion, and to monitor the target reappearance for a quick recovery. These favorable characters would resolve template update problem [62]: the tracker stops model update during full occlusion, or performs a partial model update if it can detect the occluded areas of target in the case of partial occlusion [61]. Such knowledge is also crucial for handling varying number of objects and the birth/death problem. Effectively managing the identities of objects throughout the course of occlusion (to prevent ID loss/switch) is another benefit of such detectors. Besides, such detectors facilitate target recovery right after the target reappears. For instance if the detector is global, it finds the target after the occlusion using a global search. Further application of occlusion detector is to signal motion models to prepare for occlusion state. For instance in [37], the region of interest(ROI) introduced by the motion model is expanded gradually, once the occlusion is detected, in order to cover trajectory changes during complex persistent occlusions. Depending on actual applications, the occlusion pattern could be learnt from annotated data which will increase the robustness of this scheme. Monitoring other moving objects of the scene to predict possible occlusions, is another bright idea which elevates the effectiveness of occlusion detection.

2.3.8 Occlusion Reasoning

Occlusion reasoning is the process to determine the occlusion relationships between objects explicitly and then to localize the objects accurately. This is one of the most challenging problems in visual tracking because of partial visibility of occluded objects and ambiguous correspondence between objects and their features. Layer presentation [29–31], competitive pixel assignment [32, 33], probability maps [34], and simultaneous segmenting-ordering-tracking [35] are the most successful trends in multi-target tracking under occlusions. Simplified versions of this task are handled for rigid objects using bounding contour of the motion mask, using 3D camera calibration information and using multiple calibrated cameras [40].

Pixel level analysis provides many insights into occlusion reasoning. Using probability maps [34], using “disputed” pixels [32], using non-linear feature voting strategy [86], assuming occlusion relationships as a function of only relations in previous states [110], or oppositely assuming it to be only dependent on the current state [111] are among successful solutions in handling occlusion reasoning.

Another approach to occlusion reasoning is to utilize layer representation. Layer information is crucial for estimating where a fully occluded object resides after its region splits [30]. Automatically decomposing the video into constituent layers sorted by depth, predicting self-occlusions using layered template and kinematic model, decomposing video into layers and employing EM to infer objects’ appearances and motions, defining background layers, and ground plane constraints are instances of this genre [30, 39].

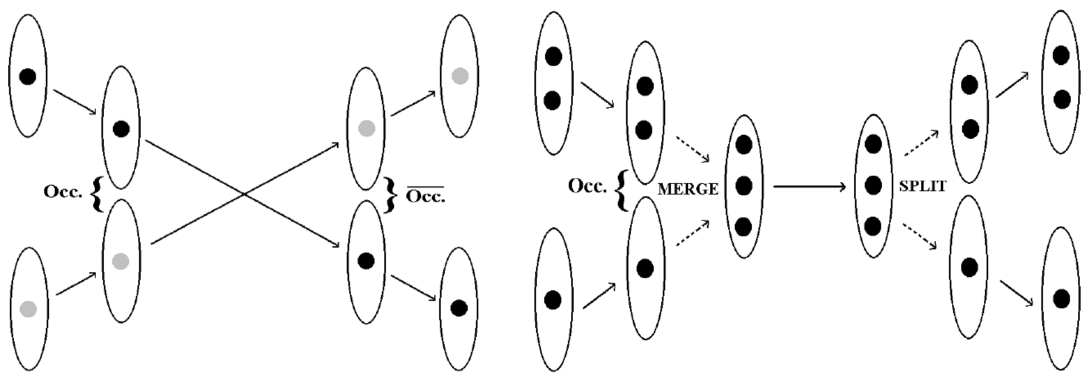
Inferring the occlusion relation of the targets is another popular approach found in the literature. Using Bayesian networks, competitive optimization using species based particle swarm optimization, and competitive game-theory based inference [33] indicate the large potential of using different AI solutions in handling this problem.

2.3.9 Summary of Literature

The diverse range of solutions proposed to tackle occlusion was characterized into eight major groups, each of which covers a different aspect of the occlusion problem. *Robust representations* provide robustness to partial observation along with other invariances such as rotation and illumination invariance, and are required for real-world trackers. Partial occlusions are handled by many modern representations, while complex occlusions still trouble many of such encodings [3]. Any problems in representation will result in model drift under partial or full occlusion and hence lead to track loss.

Motion models perform the smart selection of ROI, in which a target is expected to appear and shrink the search space significantly. This is specially useful in the case of complex partial occlusions, or even simple full occlusions. However, not all of motion models are eligible to handle persistent occlusions in which the object keeps moving while being invisible to the camera. Still hardly any of them are capable of dealing with complex persistent scenarios in which the target alters their course and speed while being invisible to the camera. Inappropriate motion model will result in track loss of the target, which requires additional mechanisms to relocate the target and continue tracking it.

Foreground tracking employs spatio-temporal segmentation techniques to embody the moving target and its accompanying objects. Using the extra “context” information in the blob facilitates handling full



(a) Model-based Tracking (b) Foreground Tracking

Figure 2.4: Model-based versus foreground tracking. In the former each blob contains only one object while in the latter a blob involves parts of one or more objects [39].

occlusions [84] while the fragmentation and merging events in the blob, often caused by complex partial occlusions, paralize this kind of scheme and thus require sophisticated designs. On the other hand varying number of objects complicates the assignment process in case of multi-target tracking. Yet this scheme provides a strong basis to handle simple partial occlusions and object interactions (Figure 2.4b).

Rapid expansion of the object detectors, especially with the advent of deep learning in computer vision [112], emphasizes on the role of *model-based tracking* that tries to find a match for the modeled target in the ROI. As the essence of tracking is locating the target in consecutive frames, this module plays a crucial part in most of the trackers, yet it is very vulnerable to occlusions. This scheme is defenseless against full occlusion, but is a powerful tool against partial occlusions, even the complex ones. Typically, more abstract models are robust against partial occlusion in the expense of higher mismatches. If the model needs to be updated frequently, it is subject to model drift problem which requires other mechanism to incorporate (Figure 2.4a).

When a strategy fails in tracking, *switching* to another strategy might help maintaining the performance of tracker. Complex partial occlusions, persistent occlusion, and even full occlusion are some of cases where ordinary trackers rarely work and thus need special treatments. Switching mechanisms monitor switching criteria and switch between strategies. For instance in [81] trackers switch to more occlusion-robust strategy when dealing with full occlusion or in [37] the primary tracker is able to deal with partial occlusions, yet switches to another strategy when dealing with complex and/or persistent occlusions.

Occlusion detection enables trackers to take appropriate action, e.g., stop model update or start a global search for the object. There exist few attempts on full occlusion detectors while partial occlusion detection relies on detailed models which in turn require expensive calculation per frame and are not feasible for online tracking or handling complex occlusions.

Occlusion reasoning additionally brings about the depth order of the objects, which can be used for more intelligent tracking. Yet this category of schemes are working with simple partial and full occlusions and their performance still heavily depends on the scenario [40].

Finally, using *multiple cameras*, by increasing redundancy, shrinks the possibility of both full and partial occlusions. Table 2.2 summarizes the occlusion handling components and illustrates their relation to the occlusion attributes. This section demonstrates that handling occlusions requires multiple components to cooperate. For instance to handle complex and persistent occlusions, [37] combined robust representation, non-linear motion model, adaptive switching method, and occlusion detection. Moreover active handling of occlusion is achieved by schemes which actively change tracking strategy according to tracking scenario and input data. The prominent active occlusion handling schemes include model updating, adaptive motion modeling, adaptive strategy switching, occlusion detection, and occlusion reasoning.

The enormous amount of ideas into occlusion handling, reasoning and detection have made it clear that these tracking issues should gain more attention. To evaluate these ideas, however, an agreed evaluation framework is required to compare the efficiency and effectiveness of them.

Table 2.2: Occlusion handling components: extent (Partial/Full), duration (Temporal/Persistent), complexity (Simple/Complex), and Active handling (Active/Passive).

Component	Extent	Duration	Complexity	Active
Robust Representation	P	-	S / C	P
Motion Models	F / P	T	S / C	P / A [1]
Foreground Tracking	P	T	S / C	P
Model-based Tracking	F / P	T / P [2]	S	P / A [2]
Mode Switching	F	-	S / C [3]	P / A [3]
Multiple Cameras	F / P	-	C	P [4]
Occlusion Detection	F / P	-	S / C [5]	A
Occlusion Reasoning	P	T	S / C	A

[1] *adaptive motion models*
[2] *with model update*
[3] *adaptive switching criteria*
[4] *here, fixed camera case is assumed.*
[5] *yet to be proposed*

2.4 Evaluation

Given the variety of occlusion circumstances in tracking, and the diverse solutions proposed to tackle this problem, it is surprising that the number of evaluation video sequences for this specific task is small. Moreover, a comprehensive and established standard to compare the ideas is lacking in the literature. This section gathers the attempts to promote occlusion handling by providing infrastructures (i.e., data and protocols) to evaluate the ideas comprehensively.

2.4.1 Criteria

Many measures for evaluating the tracking performance have been proposed, typically by comparison with ground truth considering the target presence [23]. This condition, however, reduces the applicability of those measures in the scenarios where the performance of the tracker under occlusion is also important. Ideally, a tracker is expected to track the target when it is present and to change its status to occlusion when the target is fully occluded. A good indicator of tracker's success in each frame is partial overlap that is defined as the ratio of spatial intersection between ground truth and system output over the spatial union of them. Defined in PASCAL framework, the overlap above 50% (overlap threshold = 0.5) is accepted as a tracking success which is later extended to account for occlusion handling [108]. VACE framework further extended it to multi-target tracking cases (*SFDA* metric [113]) and a threshold-free measure [24]. To account for important role-players of multi-target tracker performance such as number of objects detected and tracked, missed objects, false positives, fragmentation in both spatial and temporal dimensions, and localization error of detected objects in a single score, there is VACE *ATA* metric, which is an advanced version of the CLEAR metric with consistent object IDs [113]. Another metric is proposed in [52], which is the introduction of F-score into the tracking realm. An area based version of this score, $F1$-score, is later introduced by the same author. Good reviews over such criteria can be found in [23].

Regarding the output state of the tracker and the occlusion state of the ground truth, different kind of error can be imagined. *Type I error* occurs when the target is visible, but the tracker's output is far away from the target. *Type II error* occurs when the target is invisible, but tracker outputs a bounding box. *Type III error* occurs when the target is visible, but the tracker fails to give any output [108].

In order to demonstrate a way to analyze and measure occlusion robustness, here we formalize the occlusion problem and introduce a criterion to measure tracker's performance dealing with occlusions. Inspired by [108] and [113] here we propose a metric which supports multi-target tracking under occlusions. The metric accounts for tracker's accuracy and supports occlusions.

In a scenario with T frames, a total number of N targets are annotated which are not always present in the scene, either they enter/exit the scene in the middle of the scenario or they are occluded, thus a subset of those targets are visible in the frame $t \in \{1, \dots, T\}$. The annotation of target $j \in \{1, \dots, N\}$ in frame t is denoted by B^*_{jt} and its presence is denoted by a binary flag $s^*_{jt} \in 0, 1$, where one means the target is visible (at least partly) and zero otherwise. In the frame t, the tracker locates the target in area $\hat{B}_{jt}$ and $\hat{s}_{jt} = 1$ or announces that the target is not found/occluded by setting $\hat{s}_{jt} = 0$. The overlap of the tracker's belief about the target extent and the ground truth is a good indicator of the tracker accuracy and calculated as $v_{jt} = |B^*_{jt} \cap \hat{B}_{jt}| \div |B^*_{jt} \cup \hat{B}_{jt}|$. Here, $\cap$ and $\cup$ are intersection and union operators for areas of the image, and $|.|$ operator counts the number of pixels embodied in an area. The score is comprised of four components: For each tracker at time t the score is negative when the visibility state of the target and tracker mismatches for a target (*type II and III* errors). This score is positive when the tracker's output and the annotation overlap or when the absence of a target in the scene is correctly detected. In order to prevent type I error, the overlap value is thresholded with τ using the Heaviside step function $\chi(.)$ The final score is the sum of scores for all targets averaged along the scenario.

$$R = \frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{N} \left(\hat{s}_{jt} s^*_{jt} \chi(v_{jt} - \tau) + (1 - \hat{s}_{jt})(1 - s^*_{jt}) - (1 - \hat{s}_{jt}) s^*_{jt} - \hat{s}_{jt}(1 - s^*_{jt}) \right) \tag{2.1}$$

This criterion handles accuracy of tracking as well as changes in the number of objects detected and tracked, full occlusions, birth/death cases, ID loss/switch events, re-identification problem, confusion problem, and false occlusion alerts. It is an extension of the criterion introduced in [108] to handle multi-target tracking and punish errors, and is more reliable than VACE *ATA* [113] because of handling *type II* and *type III errors*. Since some trackers may produce outputs that have small overlap ratio over all frames while others give large overlap on some frames and fail completely on the rest, τ must be treated as a variable to produce a fair comparison [108]. Accordingly, Wu et al. [24] proposed the success plot (success score versus threshold) and considered its area under curve (AUC) as the metric for measuring the performance. Using the proposed score R, the AUC of this plot is calculated as $AUC = \int_0^1 R(\tau) d\tau$.

2.4.2 Datasets

With the ever increasing volume of datasets released online, the lack of datasets to comprehensively evaluate occlusion becomes more evident. Many datasets for different computer vision tasks have been released so far, while a few of them referenced a portion of their videos as related to occlusion. Table 2.3 shows the list of datasets including videos with occlusion. It is clear that such datasets are not specifically focused on occlusion, with few videos including only a subset of possible occlusions. So far, valuable attempts have been made to alleviate this shortcomings such as [114] which created 64 simulation video sequences to experiment the effectiveness of each tracking method in various occlusion scenarios. The authors of this study tried to isolate other tracking challenges such as shadow, illumination changes and moving background from the occlusion scenarios. Also [108] provided an RGBD dataset with high diversity, including deformable objects, various occlusion conditions, and moving camera in different scenes.

2.4.3 Guidelines from Benchmarks

Performance of a tracker varies in different scenarios and a fair comparison of them seems necessary to effectively evaluate trackers. To this end, several parallel benchmarks have been conducted in recent years. Wu et al. [24] carried out large scale experiments to understand how trackers work; especially they analyzed the initialization problem of object tracking comprehensively. A novel quantitative performance evaluation methodology was proposed in [26], which considered the tracking accuracy and durability to compare adaptive trackers versus non-adaptive ones. The experimental survey presented in [23] aimed to evaluate trackers systematically and experimentally on large number of video fragments as they believed that most of the studies used less than ten videos or special datasets for their evaluation.

The performance anaylysis on the occlusion subset of videos in [24] revealed that the trackers detailed in [11,27,70,126,127] outperformed others. The results reinforce the role of structured learning and sparse representation in occlusion handling. It was also deduced that local sparse representations are more effective than the ones with holistic sparse templates (e.g., [128]). The results further revealed that trackers tend to perform better in short sequences rather than long scenarios and the background information is critical for effective tracking. The data showed that motion model or dynamic model is crucial for object tracking, especially when the target motion is large or abrupt.

For modern trackers such as [27, 28, 70, 76, 129], under no/little motion relative to camera, even full temporal occlusions are not much of problem according to [23]. This benchmark demonstrated that occlusion with less than 30 % of target extent is now considered a solved problem. In contrast, it revealed that

Table 2.3: Datasets including videos for evaluating occlusion robustness of trackers

Name	Description
PETS	Various tasks, *2001-9*
CAVIAR [115]	Video surveillance
VIVID [116]	Video surveillance
ETIESO [117]	Smart House, *2007*
ViSOR [118]	Video surveillance, *2008*
SARC3D	People Reidentification
i-LIDS [119]	Multiple camera tracking scenario, *2008*
ETH People [120]	Moving platform people detection, *2008*
TUD People [121]	TUD-Brussels moving platform people detection
Caltech Pedestrian [122]	Pedestrians, *2009*
ALOV++ [123]	Challenging videos benchmark, *2010*
TRECVid	Large video benchmark, *2010*
Diamler [98]	Pedestrians observed from a vehicle (Multi-cue), *2010*
RGDB People	Multi-Kinect videos of people, *2011*
3DPeS [124]	People Re-identification
SimOcc [114]	64 Simulated occlusion videos, *2012*
TB-50 [24]	Popular videos benchmark, *2013*
Princeton RGBD [108]	Kinect video with occlusions, *2013*
VOT14 [125]	Visual tracking challenge, *2014*

for videos with large motion and full occlusion, most trackers have difficulty in reacquiring the target when it reappears. The results proved that [70] is overally the best tracker to handle occlusions, yet there is no tracker to handle all occlusion scenarios perfectly. Moreover, this study suggested that [71] works well for some occlusion scenarios characterized by small and fast-moving targets.

The focus of [26] is to compare the adaptive trackers with non-adaptive ones in several scenarios including partial occlusions and temporal full occlusions. For the partial occlusions [27] exhibits the best performance followed by [10] whereas other trackers [12, 47, 80] are effective only in some partial occlusion scenarios. Maintaining a good balance between stability and adaptation to appearance changes and a stable model update strategy to compensate for the over-simplistic state sampling strategy are recognized as the key elements of a successful tracker handling occlusions. Some other partly successful strategies in handling partial occlusions are identified as online feature selection [12] and deploying external labelers for the sampling and labeling stages [65, 130]. The authors argued that strong priors on target appearance are effective solutions for partial occlusions, but limit adaptability to appearance changes. They also mentioned that [70] is effective at handling occlusions but is unable to handle permanent appearance changes. Besides, this study emphasizes the role of stable model update strategy in occlusion handling (e.g., subspace or manifold update when using target-wise features in [10]) and hold overfitting to appearance changes responsible for model drift in trackers like [60]. To handle this overfitting issue, the study suggests the temporal smoothness to be enforced on the model update. Surprisingly, the study showed that a non-adaptive solution like [47] is more effective than many adaptive trackers (e.g., [10, 12]), during partial occlusions. For the temporal full occlusions the results suggest that current scale-adaptive trackers (except [70]) cannot handle this kind of occlusion well. During this experiment, even the temporal occlusion is too long for [10] to keep up, [10, 12, 70] have problems in dealing with rapid motion and consequent motion blur, and [27] has problems in handling out-of-plane rotations. Based on experiment outcomes, this study votes for [27] because of its effectiveness in dealing with rare and continuous appearance changes and robustness to partial and total occlusions and misaligned initial states.

The brief benchmark in [108] brings the flavor of error-type analysis into benchmarks. High *Type II* errors (where the target is occluded, but the tracker outputs a bounding box) is typical for the trackers without an explicit occlusion handling mechanism like [12, 13, 27, 53]. High *Type III* errors (where the object is present but the tracker cannot locate it) suggest that trackers like [65, 70] are sensitive to target appearance change or partial occlusion. Conservative approaches, which do not produce output with low confidence, often fail in tracking the target and fall into this type of error.

According to these benchmarks, *TLD* [70], *SRUCK* [27], *VTS* [76], *ColorFBT* [28], *L1O* [129], *SCM* [11], *LSK* [126], *ASLA* [127] and *FBT* [59] are successful trackers when dealing with occlusions, with the first two being emphasized for their robust performance even in challenging scenarios. Moreover, mix results have been reported about *IVT* [10] and *FragTrack* [47] which require further investigations [1].

2.4.4 Occlusion Scenarios

There are numerous possibilities for occlusion scenarios, each of which has different characteristics and requires a special kind of treatment. Several attempts have been made to describe the occlusion space, but only scratched its surface. Lee et al. [114] simulate 64 occlusion scenarios using different motion patterns (8 uniform trajectories and 8 non-uniform trajectories) and occlusion types (no occlusion, partial occlusion, full occlusion, long occlusion). In their experiment they used two rigid convex objects so that many complexities of dealing with non-rigid objects are relaxed. Meanwhile they tried to keep other factors in the scene constant. These scenarios were exaggeratedly simplified, contained no complex occlusion and even the duration of long occlusion was not enough to make the mean-shift tracker [78] drift away from the target - which is the famous shortcoming of this tracker against full occlusions.

In another study, Guha et al. [131] claimed that all occlusions can only have 14 states (OCS-14) regarding to the target being static/dynamic, degree of visibility, and state of object isolation. However, since different scenarios may result in similar states, the tracker may require different modules to be embedded to handle the same type of occlusion. The same argument applies to 3 attributes introduced for occlusion in this article. These attributes (extent, duration, and complexity) yield 8 states that only describe the occlusions, but give no information about the way of occlusions.

To approach the problem of enumerating all possible occlusion scenarios, it is important to analyze the role-players involved in scene formation: camera, light, object, and scene background [123].

Cameras observe the scene and provide the essential information to the tracker. The data obtained from a single camera lack geometric information since it maps 3D world onto a 2D image plane. Yet the viewing angle of the camera significantly affects the scene complexity, ranging from overhead cameras in surveillance scenarios, to low altitude cameras such as those mounted on mobile robots. Stereo vision and RGB-D sensors try to compensate this shortcoming by providing partial 3D information about the scene, but provide tracker with the valuable depth information for visible surfaces. Multiple camera configurations, especially overlapping ones, shrink the chance of occlusion, but require pre-calibration or real-time image registration among cameras. Camera movements complicate the tracking task drastically as they introduce rapid pose change, dynamic background, and different levels of self- and scene- occlusions. If there are more than one cameras in the scenario, the depth separability becomes an important factor in creating novel scenarios. Occlusions due to objects which are spatially far but overlapped in image plane (i.e., have large distance in z direction) are more easy to handle for occlusion reasoning methods.

Illumination of the scene is another important role-player in the tracker performance. Sudden changes in illumination degrade the performance of both model-based and foreground tracking modules. Cast shadows, by altering the object appearance, imposing self-occlusion, and causing shadow-to-object resemblance problem, further challenge appearance trackers as well as introducing a complex non-linearity to motion models. Outdoor scenes and indoor stage plays are two typical scenarios subject to drastic light change which involves full occlusions or partial occlusions due to cast shadows.

Scenes are another source of frustration for trackers, since they are composed of background and non-target objects which may provide scene-to-object occlusions. Urban scenes especially are full of distractions, clutter, and partial to full occlusions caused by static background objects (e.g. traffic signs, benches, etc.) and non-target ones (e.g., cars on the street).

The targets themselves are a great source of variation. Number of targets in a scenario can be one or more, and a varying number of targets in a scene (e.g., surveillance scenarios) challenges tracker, especially when the scenario embodies occlusions. Detection/tracking in crowded scenes (e.g., airports, subway platforms, etc.) are now getting more attention in literature. Numerous instances of partial to full occlusions in a crowded scene require dedicated treatments which introduce concepts like crowd modeling, collective action recognition, interaction analysis, etc. to assist occlusion handling. A survey on crowd tracking readers can be seen on [7]. The variety of object classes [43] (e.g., in point-of-view video footage from urban scenes), and confusion problem (e.g., tracking a student in uniform in a school, a biker in Tour de France) are two more common challenges happening in real-world scenarios.

Targets which undergo non-rigid transformations and out-of-plane and those taking complicated poses and articulated motions create self-occlusions, which trouble model-based tracking. This is a common

[1]Here, the most common tracker names in the literature or the naming in corresponding benchmark is used.

challenge in tracking non-rigid targets such as faces and pedestrians. Physical model shortcomings and high computational cost for heavily detailed models are common in such cases.

Although motion models try to capture the dynamics of the target, relative motion of the target to the camera still challenges many trackers. Bouncing, shaking, and other kinds of sudden trajectory changes along with partial or full occlusions are considered as the most challenging scenarios in many studies. Extreme cases of such scenarios include mobile camera and moving targets in an uncontrolled environment [120, 132]. The results of [23] reveal that large motions along with full occlusions impede most of the trackers, thus even the linear motions with large target displacements in a scenario involving occlusion are considered as challenging for many trackers.

Inter-object interactions constitute another challenge which may cause the occlusion in videos. Partial occlusions, split-merge events (e.g., sports scenes with complex partial occlusions), grouping-fragmentation (e.g., a group of target pedestrians walking across a busy street), and full occlusions are typical scenarios emerged from target interactions.

Current occlusion datasets do not try to minimize the effects of other factors (scale variation, background clutter, etc.) in their occlusion scenarios and the attempts to simplify such videos (e.g., [114]) are very premature.

2.5 Conclusions and Future Directions

This study provided a comprehensive overview of the occlusion problem in online visual tracking. Based on the new categorization, occlusions are defined based on their three intrinsic attributes: duration, spatial extent, and complexity. Many studies tackled partial temporal occlusions, and significant progress has been made so that temporal partial occlusions with the spatial extent of 30% are considered as solved [23]. On the other hand, few attempts have been made to handle persistent or complex full occlusions, and even temporal full simple occlusions are still challenges for many trackers. This study collected the main problems caused by occlusion and provided the best practices in order to facilitate designing more robust trackers. By categorizing the state-of-the-art solutions to the occlusion problem, this study discussed the merits and demerits of each solution and illuminated the landscape for future research.

The thorough analysis in the survey highlighted effective approaches for robust tracking and provided potential future research directions in this field. Better foreground segmentation schemes, considering the split and merge events, dealing with varying number of objects, and incorporating other visual clues (such as context and motion patterns) into formulation of the association problem are recommended approaches to advance foreground trackers. By providing more robust detectors using latest breakthroughs in object categorization and fine-level object detection, the tracking-by-detection approaches would increase the accuracy of trackers. Feature detector and tracker fusion are trending solutions in this area while simultaneous localization and detection proved to be a successful strategy initiated by [70]. Moreover, part- and patch-based solutions and robust appearance models (locally sparse, discriminative, and occlusion-invariant) are the essence of recent successful trackers. Advanced motion models such as parametric models, stochastic sampling and adaptive motion models by the guidance of, e.g., optical flows or context information, seem to be another successful strategy. Cheap access to depth information provides the opportunity to use this rich source of information to disambiguate occlusion situations, to keep track of targets, to design powerful features, and to partially compensate the 3D-2D projection data loss. Robust detection of occlusions improves the tracker performance significantly and there are lots of work to do in this track. Utilizing hybrid models and switching mechanisms in order to compensate the demerits of different trackers with one another sounds promising. Occlusion reasoning is yet to be formalized, and by doing so, many ideas can be applied to this field. Formulating it as a competitive phenomenon between objects, decompositional approach and using context are a few directions in which preliminary studies gained success.

Dedicated evaluation frameworks and benchmarks to study off-the-shelf tracker under various occlusion cases promote research in this field. Further investigations and surveys on occlusion are required and databases covering all aspects and circumstances of occlusions are yet to be made. Additionally more detailed criteria provide more insights into the dynamics of the tracker during and after occlusion and help designing better trackers.

3 Occlusion-Aware Particle Filter Tracker

"Pure mathematics is, in its way, the poetry of logical ideas."

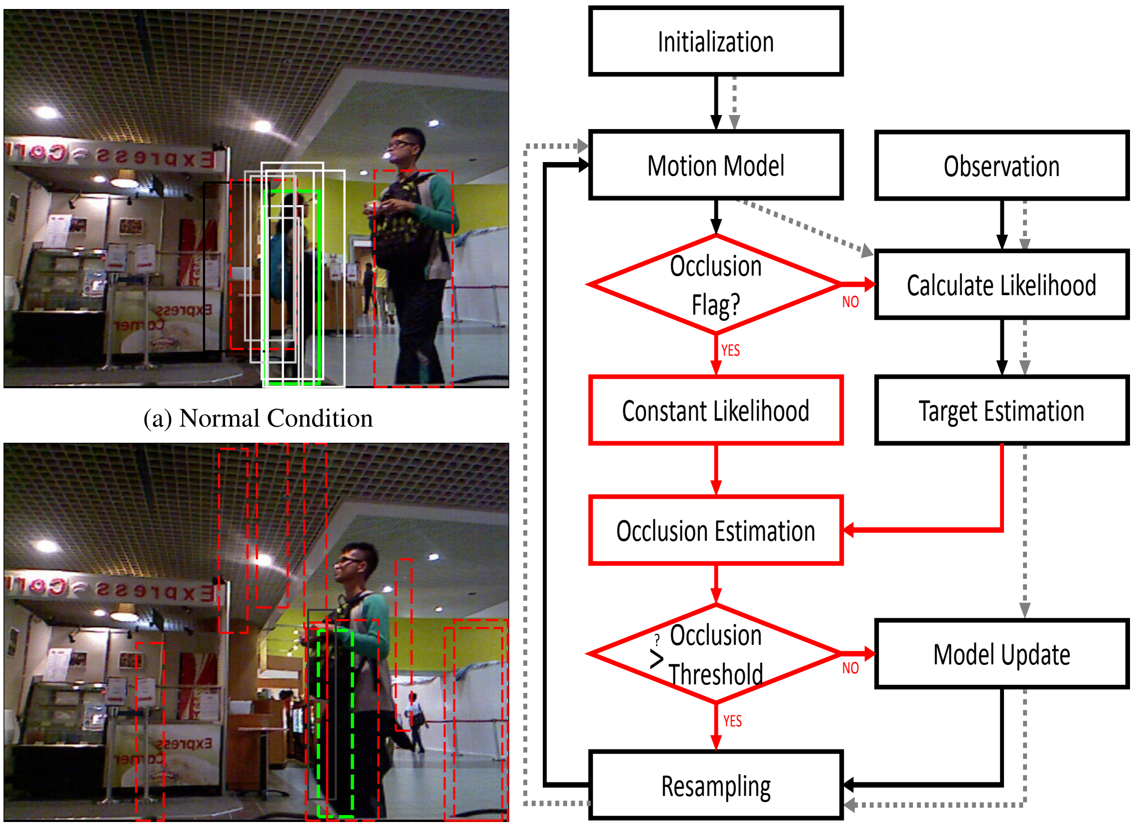
— Albert Einstein

3.1 Introduction

Among the many challenges in visual tracking, occlusion is one of the most intractable issues because the modeling and generalizing of occlusions are not straightforward [61]. Occlusions make a part or the whole target object (i.e., the object to be tracked) invisible to the sensor, where the duration of the invisibility is often unknown beforehand. Occlusions may be cast upon the target by another moving object, static background objects, or the target object itself [34]. Persistent (i.e., long-term) and complex occlusions are among the most challenging forms of occlusions. The former pertains to full occlusion which may last for several frames, whereas the latter occlusions cause drastic changes in some of the key characteristics of the target object (e.g., appearance, orientation, motion direction, size, and distance from camera).

Current state-of-the-art trackers can handle up to 30% partial occlusions [23], whereas more extensive, full, or complex occlusions trouble almost all of the trackers. Most of the existing occlusion handling techniques have critical limitations, such as the need for multiple cameras, camera calibration, strong models, and environment understanding. More importantly, many trackers are not aware of the current occlusion state of a target and lack a mechanism to recover the target after it reappears. To alleviate these issues, we propose a tracker that detects emergent occlusions, addresses difficult occlusion scenarios, and quickly performs target recovery after occlusion. This novel method builds upon *particle filter trackers* (*PFT*s) and significantly improves their resilience against various types of occlusions (including persistent and complex occlusions). The objective of this tracker is to extend particles to include an occlusion state, switch observation and motion models based on the occlusion state, weight the observation based on the observation confidence, and employ robust feature fusion.

More specifically, we propose a binary flag to attach to each particle to express the state of the tracker's belief regarding the particle occlusion. Based on this "occlusion flag", the particle goes through a feature-based template matching process (no-occlusion case) or branches to an occlusion case in which all of the occluded particles are treated uniformly. The latter case guarantees rapid expansion of the search area to accommodate possible trajectory changes during occlusions. The state transition model devised for the occlusion case allows quick recovery of the reappearing target and robust prediction of emergent occlusions. Moreover, the occlusion detection stops the model update to prevent the model from being corrupted by irrelevant data, which in turn leads to resolution of the model drift problem. Predicting occlusions and preemptive suspension of the model update are plausible solutions to the model drift problem. Such occlusion-awareness not only resolves the problem, but also accelerates the target recovery if handled effectively.



(a) Normal Condition

(b) Occlusion case

(c) OAPFT vs. PFT flowchart

Figure 3.1: The dynamics of occlusion-aware particle filter (*OAPFT*) in normal conditions (a) and occluded conditions (b). Brighter boxes are more similar to the target and red boxes are marked as occluded. The green box is the estimated target location. The comparative flowchart of the *PFT* and the proposed *OAPFT* is illustrated in panel (c).

Following this introduction, the proposed method is outlined in section 3.2 and its design is elaborated in section 3.3. The algorithm is then compared with the original *PFT* (3.4.3), with versions of itself using different feature sets (3.4.2) and with state-of-the-art trackers (3.4.4). The manuscript is then summarized with discussions and future works in section 3.5.

3.2 Occlusion-Aware Tracking

This study proposes a tracker that exposes the particle filter to a probabilistic treatment of occlusions. This tracker is able to handle persistent and complex occlusions and to achieve rapid occlusion recovery while benefiting from the fusion of various features collected from the color and depth channels. This section presents an overview of particle filter trackers (*PFT*s), followed by the basic idea underlying our proposed method.

3.2.1 Multi-cue Particle Filter Tracking

Particle filters use several candidates for a target, all of which are sampled around the expected target location considering its motion pattern, and they suit non-linear and non-Gaussian scenarios. These candidates resemble the target to a certain degree. This similarity serves as the weight in the linear combination of all particles voting for the new location of the target. The stochastic sampling of posterior possibilities provided *PFT*s with various extensions. Of these, *Condensation PFT* [133] was initially developed to track objects in cluttered environments using edge information, but it was later extended to use kernels [44], to fuse multiple cues [96], and to employ sparse coding [134].

A particle filter involves two iterative steps: a prediction step, which predicts the target in the current frame based on previous observations as a prior distribution, and an update step, which maintains samples'

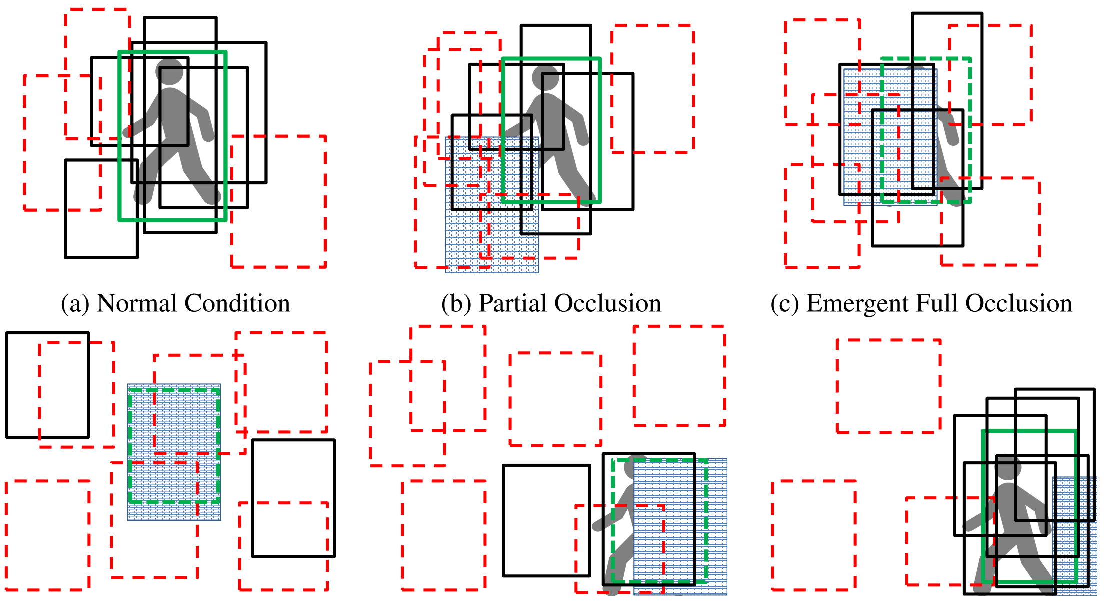


Figure 3.2: Phases in occlusion recovery. Black: non-occluded particles, Red: occluded particles, Green: estimated target, Dashed: occlusion flag set.

(particles') weights to construct a posterior distribution according to the framework of incremental Bayesian inference [133].

In the context of visual tracking, let $\{Y_1, Y_2, \ldots\}$ be the observations (e.g., color and depth information taken by a RGB-D sensor) and $\{X_1, X_2, \ldots\}$ be the states (e.g., position and size of the bounding box of the target in the video frame), where Y_t and X_t denote the observation and state variables at time t, respectively. At each time t, a general Bayesian filter approximates the prior distribution of X_t given all previous observations $Y_{1:t-1} = \{Y_1, Y_2, \ldots, Y_{t-1}\}$ up to time $t-1$ as

$$p(X_t|Y_{1:t-1}) = \int p(X_t|X_{t-1})p(X_{t-1}|Y_{1:t-1})dX_{t-1}, \tag{3.1}$$

where $p(X_{t-1}|Y_{1:t-1})$ is the posterior at the previous time and $p(X_t|X_{t-1})$ is known as the motion model. Following the Bayesian rule, a new observation Y_t leads to the posterior distribution at this time as

$$p(X_t|Y_{1:t}) = \frac{p(Y_t|X_t)p(X_t|Y_{1:t-1})}{p(Y_t|Y_{1:t-1})}, \tag{3.2}$$

where $p(Y_t|X_t)$ denotes the observation likelihood. According to particle filter, the posterior distribution $p(X_{t-1}|Y_{1:t-1})$ is approximated as empirically a finite set of N particles, $\{X_{t-1}^{(j)}\}$ ($j = \{1, \ldots, N\}$) with importance weights $\pi_{t-1}^{(j)}$. The next candidate samples at time t, $X_t^{(j)}$, are drawn from the empirical prior distribution $\sum_{j'=1}^{N} \pi_{t-1}^{(j')} p(X_t|X_{t-1} = X_{t-1}^{(j')})$ in the prediction step, and then in the update step the weights of the samples, $\pi_t^{(j)}$, are updated as $\pi_t^{(j)} \propto p(Y_t|X_t = X_t^{(j)})$.

Inspired by [96], the mutually independent multi-cue observation model is written as

$$p(Y_t|X_t^{(j)}) = \prod_{i=1}^{F} p_i(Y_t|X_t^{(j)}) \propto \prod_{i=1}^{F} \exp\left(-\gamma_i D_i\left(\phi_i(Y_t^{(j)}), M_t^i\right)\right), \tag{3.3}$$

where F indicates the number of features, γ_i denotes the relative importance of feature $i \in \{1, \ldots, F\}$ against other features, $Y_t^{(j)}$ is the image patch induced by particle $X_t^{(j)}$ from the observation Y_t at time t, and $D_i\left(\phi_i(Y_t^{(j)}), M_t^i\right)$ is a distance function between an extracted feature vector $\phi_i(Y_t^{(j)})$ and template M_t^i for feature i. The template is obtained by extracting features from the available annotation in the beginning of the tracking scenario. Finally, the particle filter approximates the target location via a weighted sum of its particles

$$\hat{X}_t = \sum_{j=1}^{N} \pi_t^{(j)} X_t^{(j)}. \tag{3.4}$$

Regular *PFT*s, such as [44, 135], can handle partial and temporal occlusions, benefiting from many scattered particles, but their accuracy degrades when there are changes in the target's trajectory during occlusion. The tracker is unable to capture target deformations and illumination changes when the target template is fixed throughout tracking. Hence, like many other trackers, a *PFT* should update its template when obtaining new observations [96]. To attenuate appearance changes, researchers attempt to combine several features [136] or to learn a robust subspace to update the template [10]. Nevertheless, for longer occlusions, an occluded target cannot be recovered because of model drift [137]. This argument also applies to full and persistent partial occlusions. In another attempt to improve *PFT* performance and robustness against occlusions, [134] proposed *L1 minimization PFT* (*L1T*), which can conceptually handle occlusions and observation corruptions. However, [61] reported that the performance of this tracker was not satisfactory in practice, which is partly because *L1T* does not consider the occlusion status in its likelihood computation and template update. Additionally, complex occlusions impair many trackers including *PFT*s, due to drastic template changes during occlusions. Moreover, drastic trajectory changes of a target (e.g., bouncing) may render specialized motion models useless, such that the target is easily lost due to an improper search region that does not include the target.

Several researchers have tried to expand the abilities of the *PFT* to handle a wider range of occlusions. *Condensation PFT* has an inherent drawback called the "outlier problem". This problem occurs when particles with contaminated measurements and low probability shift the posterior approximation away from the real target distribution. In an extreme condition, many particles may be located around outliers, leading to an erroneous approximation. Occlusion, clutter, moving distractors, and insufficient sampling of the target dynamics can cause this problem. To alleviate the problem, several versions of particle filters (e.g., *Auxiliary PFT* by [138]) have been proposed to alter the trackers in different scenarios. Hybrid and switching particle filters have also emerged to compensate the shortcomings of one tracker with the merits of another. However, balancing the trade-off between different trackers depends on the task at hand and the scenario conditions, and it heavily relies on the occlusion type. Switching between *PFT*s and other types of trackers (e.g., *Mean-shift Tracker* in [85]) is the subject of several studies, yet the inconsistency between different aspects of the trackers impedes successful switching and ignores occlusion handling on many occasions. In [109], the motion model of the *PFT* switches to the "random walk" in the event of occlusion to facilitate target recovery. Once the total likelihood of the particles falls below a certain threshold, the system reports an occlusion. However, the likelihood degradation is not uniquely caused by occlusions but also by trajectory changes. In another study by [81], a shared pool of particles are sampled by two different versions of particle filters, one that is good at handling occlusions and the other powerful in accurate localization, based on pre-determined linear switching functions. This over-simplified approach suffers from parameter sensitivity and is unable to handle various occlusions that require high flexibility.

There are two possible ways to improve particle filter approximations:

1. improve the positioning of particles, and
2. enhance the approximation to be more faithful to the real distribution by accounting for the factors involved in the approximation.

Although the former was the goal of aforementioned studies, the solutions fail if the observation model cannot well explain the occlusion situation. In this study, we focus on the latter approach by directly considering occlusions in the particle filter approximation. In addition to the outlier problem, *PFT*s usually suffer from other issues, such as model drift, local optima of the feature space [135], noise and other defects in feature calculations. Additionally, the lack of an occlusion detection module and the blind search for an occluded target degrade their performance significantly in persistent and complex occlusions. In the next section, we propose solutions to address these issues, which together constitute our "occlusion-aware particle filter tracker".

3.2.2 Full Occlusion Detection and Handling

To handle persistent and complex occlusions, the algorithm should predict/detect emergent occlusions to prevent the template from updating with irrelevant data. During the occlusion, the particles should not be disturbed by non-target parts of the scene. Besides, the search area should be expanded gradually to capture the target because its course and/or velocity may change during the occlusion. After the occlusion, however, the particles should converge to the target to continue tracking. To realize these requirements, in this proposed method, a binary occlusion flag is introduced to the state representation of every particle. The target's bounding box $bb_t^{(j)}$ is characterized by its location (x, y) and scale (w, h), and moreover an occlusion flag z, $X_t^{(j)} \equiv \langle bb_t^{(j)}, z_t^{(j)} \rangle = \langle x_t^{(j)}, y_t^{(j)}, w_t^{(j)}, h_t^{(j)}, z_t^{(j)} \rangle$.

These flags, $z_t^{(j)}$, partition the entire particle population into two fractions: those that are considered as

Algorithm 1: Occlusion-aware Particle Filter Tracker

```
input  : Target initialization $bb_1$
output: Target state $\hat{X}_t = \langle \hat{bb}_t, \hat{z}_t \rangle$

Initialize target template
Create N particles ($bb_t^{(j)} \leftarrow bb_1, z_1^{(j)} \leftarrow 0$)
for $t \leftarrow 2$ to $T$ do
    for $j \leftarrow 1$ to $N$ do
        if $z_t^{(j)} == 1$ then particle is occluded
            Apply occlusion case motion model (eq(3.11))
            Occlusion likelihood (eq(3.6))
        else particle is not occluded
            Apply motion model to the particle
            Template matching likelihood (eq(3.5))
    Normalize particles' likelihoods (section 3.3.2)
    Approximate new target state $\hat{b}_t$ (eq(3.8))
    Calculate occlusion state $\hat{z}_t$ from all particles (eq(3.9))
    if $\hat{z}_t == 0$ then
        Update template (eq(3.13))
    Resample particles
```

occluded, $\mathcal{J}_t^1 \equiv \{z_t^{(j)} = 1\}_j$, and those that are not, $\mathcal{J}_t^0 \equiv \{z_t^{(j)} = 0\}_j$. In the beginning of the algorithm, a few of the particles are stochastically marked as occluded (Figure 3.2a) and start to scatter away from the target. If they stumble upon an occluder, they remain occluded, switch their observation model and start to duplicate via particle filter resampling (Figure 3.2c). If many of the particles are marked as occluded, the target is recognized as being in an occlusion state, and the model update stops (Figure 3.2d). During the occlusion, the occluded particles move in a random walk pattern (i.e., switch motion pattern), scatter from the last known position of the target and search a wider area as time proceeds (Figure 3.2e). During occlusion, if the tracker finds a particle similar to the target with high likelihood, that particle is replicated multiple times in the population, and in a few frames (up to three frames in our experiments) the target is recovered from the occlusion state (Figure 3.2f) so that the tracking continues.

Formally speaking, the observation is augmented with a binary latent variable to enable the sampling method to arbitrarily select between two motion models and likelihood computations. In no-occlusion situations, the algorithm works similarly to the original PFT (i.e., eq(3.3)), with a few particles marked as occluded, illustrated by red in Figure 3.1a,

$$l(Y_t^{(j)}|X_t^{(j)}, z_t^{(j)} = 0) = \prod_{i=1}^{F} \exp\left(-\gamma_i D_i\left(\phi_i(Y_t^{(j)}), M_t^i\right)\right), \tag{3.5}$$

where $l(.)$ denotes the (unnormalized) likelihood function.

During occlusion, however, the occluded particles over-count the non-occluded particles, causing the occlusion flag of the estimated target to be set and preventing the model update. The particles start to scatter quickly, as shown in Figure 3.1b, which illustrates the particle's state three frames after the onset of a full occlusion. To enable the algorithm to detect an occlusion without bias over different positions in the frame, the likelihood of occluded particles is assumed to obey a uniform distribution, which essentially means:

$$l(Y_t^{(j)}|X_t^{(j)}, z_t^{(j)} = 1) = L_{occ}, \tag{3.6}$$

where L_{occ} is a constant tuned by training (see section 3.3.3). The proposed observation model for each particle is obtained by combining eq(3.5) and eq(3.6),

$$\begin{aligned} p(Y_t^{(j)}|X_t^{(j)}) \propto (1 - z_t^{(j)}) l\left(Y_t^{(j)}|X_t^{(j)}, z_t^{(j)} = 0\right) \\ + z_t^{(j)} l\left(Y_t^{(j)}|X_t^{(j)}, z_t^{(j)} = 1\right). \end{aligned} \tag{3.7}$$

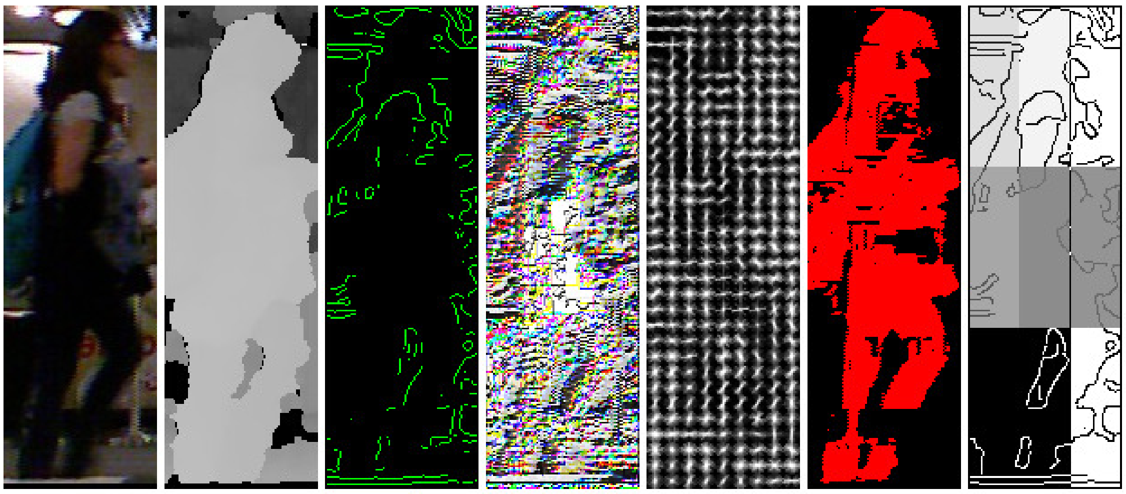


Figure 3.3: Observation and feature space: Template image observed from different channels and transformed to various feature spaces; from left to right: (1) RGB color, (2) Depth (brighter := closer to camera), (3) Edges (described by LoG), (4) Texture (described by LBP), (5) HOG domain, (6) Foreground mask, and (7) Probability mask (brighter := less occlusion probability).

By introducing eq(3.7), instead of eq(3.3), into the normal PFT framework, the particle weights $\pi_t^{(j)}$ are obtained by normalizing the observation likelihoods, making the estimation of the next target location straightforward. Because only the non-occluded particles contain localization information of the target, we modify eq(3.4) into

$$\hat{bb}_t = \sum_{j \in \mathcal{J}_t^0} \pi_t^{(j)} bb_t^{(j)}. \tag{3.8}$$

Similarly, it is possible to estimate the occlusion state of the target using all of the particles:

$$\hat{z}_t = u\left(\sum_{j=1}^{N} \pi_t^{(j)} z_t^{(j)} - \delta_{occ}\right), \tag{3.9}$$

in which δ_{occ} is an occlusion threshold, and $u(x)$ is a step function that returns one if $x > 0$ and zero otherwise. These equations provide the target localization $\hat{X}_t \equiv \langle \hat{bb}_t, \hat{z}_t \rangle$ at time t.

Assuming an independent temporal smoothness on the target bounding box $\hat{bb}_t$ and its occlusion flag $\hat{z}_t$, the motion model is given by

$$p(X_{t+1}|\hat{X}_t) \equiv p(bb_{t+1}, z_{t+1}|\hat{bb}_t, \hat{z}_t) = p(bb_{t+1}|\hat{bb}_t)p(z_{t+1}|\hat{z}_t) \tag{3.10}$$

which initializes the search area for the next frame, $t + 1$. The motion model for the non-occluded particles is arbitrary (e.g., constant velocity), whereas for the occluded particles the motion model of the bounding box, $p(bb_{t+1}|bb_t)$, is a zero-mean Gaussian $\mathcal{N}(0, \Sigma_{occ})$ with a diagonal covariance matrix,

$$bb_{t+1}^{(j)} = bb_t^{(j)} + \mathcal{N}(0, \Sigma_{occ}) \;,\; j \in \mathcal{J}_t^1 \tag{3.11}$$

where $\Sigma_{occ} \equiv diag(\sigma_x^2, \sigma_y^2, \sigma_w^2, \sigma_h^2)$ is the system noise for a multivariate normal distribution describing the uncertainty of the occluded target. The transition model of the occlusion flag, $p(z_{t+1}|z_t)$, is a 2×2 probabilistic matrix and was determined by cross-validation (see section 3.3.3).

The dashed gray arrows in Figure 3.1c depict the particle filter tracking pipeline. The additional steps of the proposed method are colored red in the same flowchart, with the solid arrows indicating the control flow of the algorithm. The whole algorithm is displayed in Figure 3.2 and Algorithm 1. [1]

3.2.3 Partial Occlusion Handling and Scale Adaptation

In addition to full occlusions, the proposed tracker copes with a significant portion of partial occlusions. When a target is partially occluded by another object or background, the observation contains irrelevant

[1]A preliminary non-peer-reviewed version of this algorithm was presented in [139].

data. The normal particle filter tends to modify the observation window (here, the bounding box) to maximally match the template. This process is usually not quick enough to prevent the template from being contaminated by irrelevant data in the new observation.

Furthermore, adapting to the ever-changing scale of the target requires a variety of particles to sample the image with different sizes and locations. In this process, larger bounding boxes that have more overlapping area with the target are preferred. Such bounding boxes, however, increase the chance of containing the occluded data, and should be punished. In a typical bounding box, some parts contain many foreground pixels and others contain many background pixels; such non-uniformity may produce a non-uniform information distribution of the foreground object over the bounding box [44].

To address these issues, we propose a "2D projection confidence" for evaluating the observation, which is robust even when background data are included in the observation due to the simplified box approximation and is easily integrated into our likelihood model consisting of multiple features.

In the proposed approach, first, a background detection algorithm is employed to construct the background mask of the frame t. Next, the background mask of the observation bounding box of particle j is clipped out of it. The bounding box is then divided into a 3×3 regular grid of cells. In each cell, the ratio of foreground pixels to all pixels is calculated. This simple grid pattern, however, is not robust against object deformation and rotation, hence, we use an empirical distribution of these measurements to gain the desired types of robustness.

The nine ratio values obtained for the cells are vectorized as $R_t^{(j)} = \{r_t^{(j(c))}\}$, in which $c = \{1, \ldots, 9\}$ is the cell number. We obtain nine empirical probability distributions ω_c of these r_c values, one for each cell (training procedure is elaborated in section 3.3.1). Each distribution indicates the probability of each cell to contain useful information. The confidence of the measurement in the cells is calculated from $\{r_t^{(j(c))}\}$ and the observation confidence mask is constructed by

$$\phi_{2D-proj}(bb_t^{(j)}) = \{\Omega_c(r_t^{j(c)}; \theta_c)\}\ ,\ c = \{1, \ldots, 9\}, \tag{3.12}$$

where θ_c's are the trained parameters of the distributions $\Omega_c \equiv \{\omega_c\}$. These confidence measures rate the observation quality to update the template in an intelligent manner. Moreover, they repel the estimated target location from the occluder by punishing partially occluded particles.

The idea of using observation masks seldom appeared in the literature. Spatially biased weights on target observations [54] masked the whole observations based on geometric considerations and the ratio of foreground pixels to all pixels was employed in [87]. The occlusion mask presented in [61] is closely related to the proposed measure (Figure 3.4). Similar to this study, in the proposed method the target is divided into a regular grid, with each cell treated independently. [61] determined the binary state of occlusion for each cell using a single "universal" classifier. This classifier was trained in terms of the patch likelihood associated with the cells in the target. In contrast, we determine the degree of occlusion for each cell, and by using nine different distributions, one for each cell, we are able to determine the confidence degree of different cells. This real-valued confidence vector has superior performance in many cases, for instance, when a partially occluded cell contains a prominent feature to assist tracking (e.g., tracking a pedestrian with a yellow T-shirt). In addition, the classifier in [61] is prone to be disturbed by illumination and sampling distribution changes.

Using kernels to put larger weights on the central pixels rather than those in peripheral parts of the observation has been successful in handling minor partial occlusions, which are more likely to happen in the edges of the observation. Accordingly, [44] used a radial kernel that puts more emphasis on the observation around the center of a bounding box, whereas [78] utilized other kernels (e.g., Epachnikov or Gaussian) for the same reason.

Intuitively, this confidence measure constitutes a spatial filter to detect the rough occupancy of the target's shape in the bounding box, in a data-driven manner. The coarse nature of this gridding provides robustness against deformations or articulations. As a consequence, this measure assigns low probability to the occluded cells.

The proposed scheme also facilitates scale adaptation for the PFT. It is achieved by punishing excessively small or unnecessarily large particles. The background mask of such particles varies significantly from the template; in the former case the bounding box covers a portion of the object while in the latter case, the foreground ratio of cells deviates from that of the original template because of the scale change. Hence, the particles with improper scale are assigned with less probability and their effects on the estimated bounding box are undermined. This measure can also be adaptive to specific objects (e.g., pedestrians) as in [139]. Furthermore, it compensates degeneracy cases in which the tracking performance can be deteriorated due to some cells lacking good features to track [140].

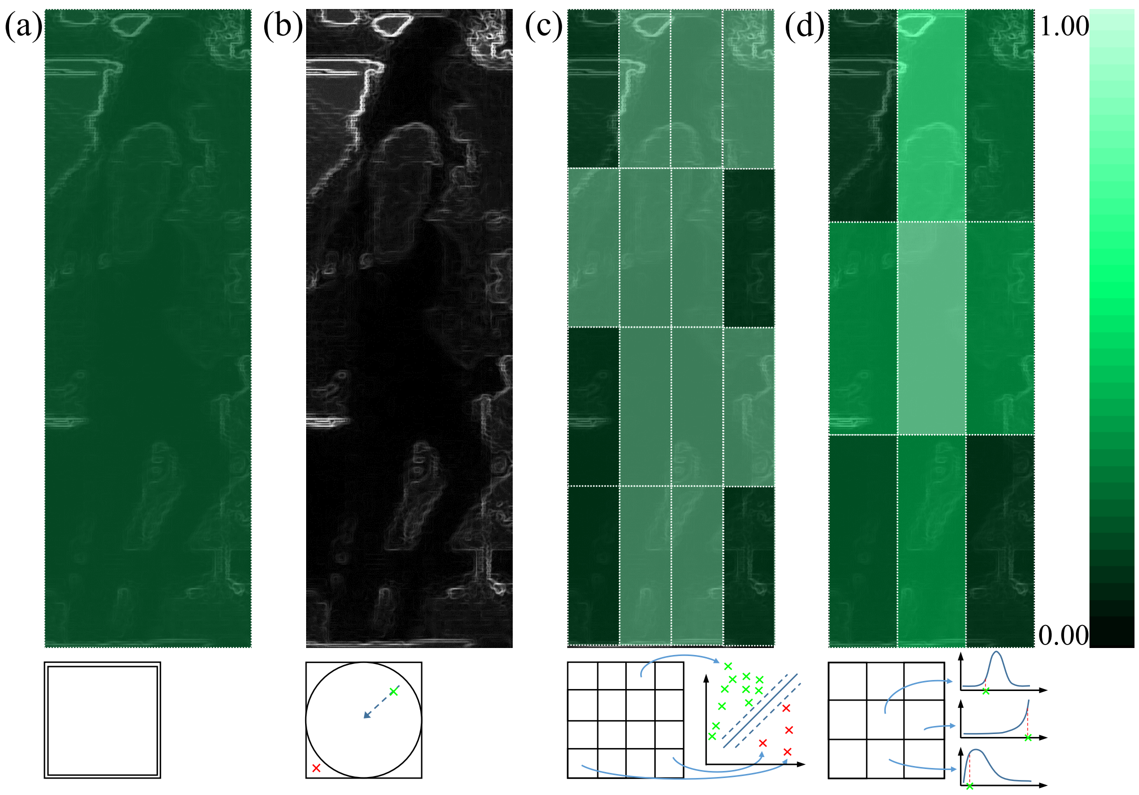


Figure 3.4: Different types of observation masks: (a) a fixed weight on all pixels, but the weight differs for different observations [54, 87]; (b) kernel defined weights for pixels [44, 78]; (c) binary mask for each cell of the regular grid obtained by the occlusion detector [61]; (d) importance mask for each cell of the regular grid obtained from the occlusion probability distributions (proposed).

3.2.4 Template Update

Our template update strategy is inspired by [62]; the update stops once an occlusion is detected, i.e., $\hat{z}_t = 1$. The template update follows a leaky bucket strategy, with an individual forgetting factor ξ_i for each feature i. Considering the reliability of the observation represented as the aforementioned confidence measure, $\eta_t = \prod_{c=1}^{9} \phi_{2D-proj}(\hat{r}_t^{(c)})$ [2], the model update is enhanced; that is, less reliable observations have less contribution to the updated template,

$$M^i_{t+1} = \begin{cases} M^i_t & , \hat{z}_t = 1 \\ \eta_t \xi_i M^i_t + (1 - \eta_t \xi_i)\phi_i(\hat{Y}_t) & , \hat{z}_t = 0 \end{cases} \tag{3.13}$$

where $t > 1$, $\hat{Y}_t$ is the observation induced by the estimated bounding box $\hat{X}_t$, and $\phi_i(\cdot)$ is an employed feature vector (for more details, see below).

3.3 Learning from the Data

In this section, we first describe how we tuned the parameters in the calculation of the 2D projection confidence measure. Next, we describe the normalization method and settings for parameters used for the motion model and employed features. These processes were performed in a data-driven manner and thus can adapt to arbitrary tracking scenarios.

[2]In order to obtain the overall confidence of the observation, we assume independence between cell observations and aggregate them all together. Here the likelihood of each cell was computed as the probability of that cell having foreground ratio $\hat{r}_t^{(c)}$ based on the Beta distribution estimated a priori (see Section 3.1). The foreground ratios were extracted from the estimated target location $\hat{bb}^c$.

Table 3.1: Letter codes of the features and the employed dissimilarity measures

Code	*Feature Name*	*Dissimilarity Function*	*Parameters*
C	Histogram of Colors [139]	KL Divergence	Adaptive Binning, 40 bins
D	Histogram of Depth	Chi-Sqaure	256 bins
E	Template of Edges [142]	Hausdorff	–
G	HOG [143]	L2	36 bins, Signed Orientation
S	3D Shape Parameters [144]	L1	Grid Size 10×10 pixels
T	Histogram of Texture [145]	Bhattacharyya	64 bins for each color channel

3.3.1 Obtaining the Occlusion Mask

To calculate the 2D projection confidence measure in a data-driven manner, for each cell c, we trained a distribution ω_c of the foreground ratio by tracking different targets in different sequences. This process requires background subtraction techniques to locate foreground pixels and count them. In this study, an appearance-based background subtraction algorithm proposed by [141] was used. To collect data, we ran the proposed tracker without the 2D projection confidence measure on training sequences clipped from annotated videos, such to include non-occluded and partially occluded frames. Each bounding box detected by the tracker in each frame of a training sequence was compared with the available ground truth, and accepted if the normalized intersection between the two exceeded 50%. The foreground ratios were then calculated for the set of accepted bounding boxes. Finally, a distribution ω_c was fit to the ratios obtained for each cell c, yielding nine trained distributions, Ω_c, with their parameters θ_c's. Having observed the data, we discovered that a *Beta distribution* appropriately explains the variations in the data, using only two parameters for each distribution. This scheme was embedded into our proposed framework similar to features, with the difference that its template was always kept fixed, $M_t^{2d-proj} = \{0, \dots, 0\}$.

3.3.2 Robust Feature Integration

The proposed tracker accommodates an arbitrary number of features to be fused. Feature fusion increases the observation richness, smooths the feature-space easier to approximate, decreases the sensitivity against feature noise, and enhances the observation-target mapping. In this study, we prepared a rich pool of features coming from the depth channel in addition to the color channels; the depth feature fosters handling of appearance changes, camera movements and spatial disambiguation of similar objects. The corresponding dissimilarity functions were chosen based on the previous studies, and their parameters were set as recommended in the original papers. Note that we will later examine in details the effects of different features and also present the optimization way to choose the best feature subset from the pool of features. The selected features are listed in Table 3.1.

To realize robust feature fusion, the following normalization on the likelihood was performed. First, for each feature, the likelihood of all of the particles was rescaled to reside within the range of $[\epsilon, 1]$ $(0 < \epsilon \ll 1)$. Then, the likelihood of a non-occluded particle was calculated as the product of all of the features. Finally, the likelihood of all of the particles (including occluded particles with a constant likelihood) was normalized to form a proper observation probability of the particles. This normalization solves the outlier problem by undermining the effects of possible outlier particles. In addition, it handles possible feature failure caused by a sudden illumination change for instance. Additionally, the normalization prevents feature dominance cases in which one feature assigns a very low or very high likelihood value to all particles and impedes approximation based on difference between the particles.

3.3.3 Parameter Tuning

The proposed algorithm has several sets of parameters. Feature-related parameters (γ_i and ξ_i) control the feature fusion and the template updating. The dynamics of the particle filter are governed by the noise variance of the object motion model, the number of particles and the occlusion flag transition matrix. Moreover, the ability of the algorithm to detect emergent occlusions and to safely recover from them depends on the occlusion threshold and the occlusion-case likelihood.

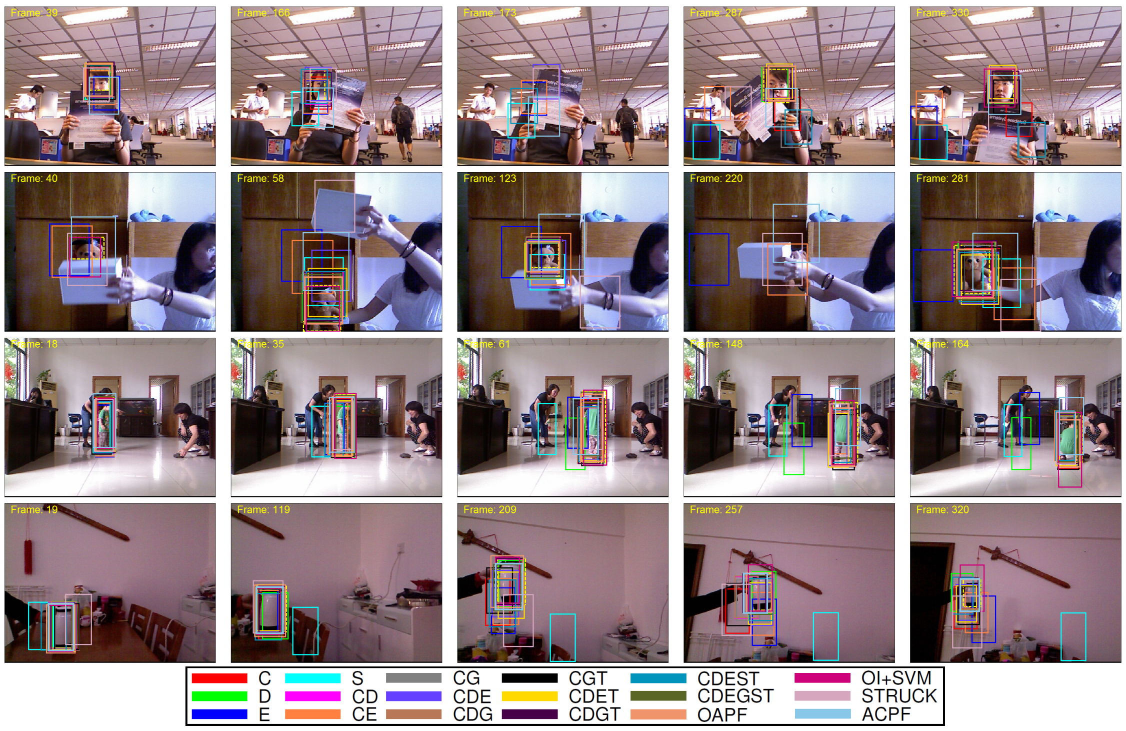


Figure 3.5: A visual review of the trackers' performance on four other annotated sequences. Note that some features are not effective in isolation and lose the target during tracking (e.g., S in the lower sequence). For more information, refer to the supplementary material and the project webpage: `http://ishiilab.jp/member/meshgi-k/oapft.html`.

Benefiting from the feature normalization procedure above, the tracker is not very sensitive to the parameter γ_i so that it is set at an appropriate constant. The number of particles (N) and the noise of the object motion mode (Σ_{occ}) were determined heuristically; the noise variance is set at three times the largest value change of the corresponding bounding box parameter, and 2000 particles is used for both *PFT* and *OAPFT* considering the video size of 640×480 pixels. The model forgetting factors (ξ_i) were determined by cross-validation on the training set.

The other parameters, the occlusion related parameters (L_{occ}, δ_{occ}) and the motion model (state transition matrix) of the occlusion flags $p(z_{t+1}|z_t)$ were highly correlated so that changing one affected the others. These four parameters (the state transition matrix has two degrees of freedom) were then simultaneously optimized by simulated annealing, i.e., in each optimization iteration, the values of the four parameters were updated by means of Boltzmann exploration, one after another.

3.4 Experiments

The proposed occlusion-aware particle filter tracker was evaluated in various tracking scenarios. First, we examined the effects of different feature sets by examining the performance of the main body of our tracker, i.e., the occlusion-flag-enhanced *PFT*. After that, the full framework including the 2D projection confidence measure (Section 2.3) was compared with well-established occlusion-robust RGB and RGBD trackers, by using five annotated videos. Next, our method was compared to its baseline *PFT* to illustrate the effect of the binary occlusion flag and its benefits for the tracker. Our full tracker was further compared to state-of-the-art RGB and RGBD trackers using the 95 videos registered in Princeton Tracking Dataset [108]. The evaluation was based on the dataset benchmarking system with unpublished annotation. The videos used in the experiments, the output of all of the trackers for each video, and the respective detailed evaluation plots are available in `http://ishiilab.jp/member/meshgi-k/oapft.html`.

3.4.1 Data and Criteria

The Princeton Tracking Database [108] contains 100 RGB-D video sequences captured by Microsoft Kinect in which five videos, `new_ex_occ4`, `face_occ_5`, `bear_front`, `child_no1` and `zcup_move_1`, with a total number of 1202 frames are annotated, and the other 95 videos are not annotated. Each sequence includes a single target from a set of object categories (human, animal, or rigid object), sizes and motion patterns. The sequences are constructed to have different occlusion types (partial, split and merge, temporal full, persistent full, and complex occlusions, in addition to many articulations and self-occlusions), so that a single sequence may have one or more of these occlusions with different extents and durations.

We used the five annotated videos for the first two experiments, in which the results were obtained by a leave-one-out procedure, that is, after the parameters were determined based on four videos, our tracker was tested using the remaining video. Repeating this process five times by changing the test video, we obtained an average performance of them. Due to random nature of *PFT* and *OAPFT*, we ran these trackers five times in each experiment, and the average performance over the five runs was reported.

The performance of each tracker was evaluated in terms of several criteria. The overall performance was measured using the tracking success score S_t, which is the ratio of the normalized intersection between the estimated target area and the real target area, if not occluded. As defined in [108], this criterion is calculated by

$$S_t = \begin{cases} \frac{|\hat{bb}_t \cap bb^*_t|}{|\hat{bb}_t \cup bb^*_t|} & , \hat{z}_t = z^*_t = 0 \\ 1 & , \hat{z}_t = z^*_t = 1 \\ -1 & , \hat{z}_t \neq z^*_t \end{cases} \tag{3.14}$$

where $|.|$ denotes the area of the region, and $\hat{bb}_t$ and bb^*_t denote the estimated and real target bounding boxes, respectively. The occlusion states of the estimated and real targets are denoted by $\hat{z}_t$ and z^*_t, respectively. To measure the tracker's overall performance on all video frames, we counted the number of frames in which the success degree was larger than a given threshold t_o. The area under the curve (AUC) of the resulting plot (success frames vs. t_o) is reported in Table 3.2.

To provide better insight into the algorithm's outcomes, the errors of the central point location (CPE) and scale (SAE) were defined as the L2-norm difference of the center position (x, y) and scale (w, h) between the estimated bounding box and the true bounding box, respectively. The average values for all video frames are also presented in Table 3.2.

Along with these measures, the reliability of the tracker, i.e., its ability to address occlusions, was calculated. In a false tracking case, the tracker did not recognize that the target was in fact occluded; *FT* denotes the ratio of such cases. Such errors are also known as *Type II* errors. In contrast, a target miss (*Type III* error) case occurred when the target was visible but the tracker failed to track it, assuming the target was still in the occlusion state; *MI* denotes the ratio of such cases. Furthermore, a mismatch case was detected when the estimated bounding box had no overlap with the true bounding box; *MT* denotes this ratio. This criterion was introduced by the PASCAL VOC challenge [146] to count a *Type I* error if the normalized overlap between the estimated and true boxes was less than a fixed threshold (e.g., 0.5).

3.4.2 The Effects of the Features

To explore the optimal set of features, we prepared a pool of features from the intensity, color, and depth domains. To obtain the best feature subset from the pool, we performed a leave-one-out cross validation on the training set, and chose a feature subset with the highest average cross-validation test score. Note that when optimizing the feature set, only the *occlusion-flag-enabled PFT* without the 2D confidence measure was employed to particularly evaluate the occlusion handling ability of this part of the proposed tracker.

In addition to the optimization of the feature subset, to comprehend the effects of various combinations of features in our pool, the *occlusion-flag-enabled PFT* (section 3.2.2) was examined with different feature subsets. The best performing subset is later tested against the whole dataset in Section 3.4.4. For the sake of brevity, we use letter codes of Table 3.1 for the employed features. For instance, tracker CDE used three features: histogram of colors (C), histogram of depth (D), and template of edges (E). For comparison, three existing trackers were also examined in this experiment: *(i)* an adaptive color-based particle filter (*ACPF*), which is a particle filter with a radial kernel (Figure 3.4) and a leaky-memory model update for its histogram of color feature [44]; *(ii)* an RGBD SVM tracker with an occlusion indicator (*OI+SVM*) which uses a latent SVM with HOG features extracted from the color and depth images and an additional *optical flow tracker* to keep track of the occluders [108]; and *(iii)* a kernelized structured output SVM (*STRUCK*) which uses its learning unit to directly estimate the object transformation between frames, and employs Haar-like features and intensity histograms [27]. Two recent benchmarks of visual tracking [23, 25] reported, based on their rich evaluation datasets, that *STRUCK* is one of the best RGB-based tracking algorithms handling occlusions; thus, it was selected to represent state-of-the-art RGB trackers in this experiment. The color

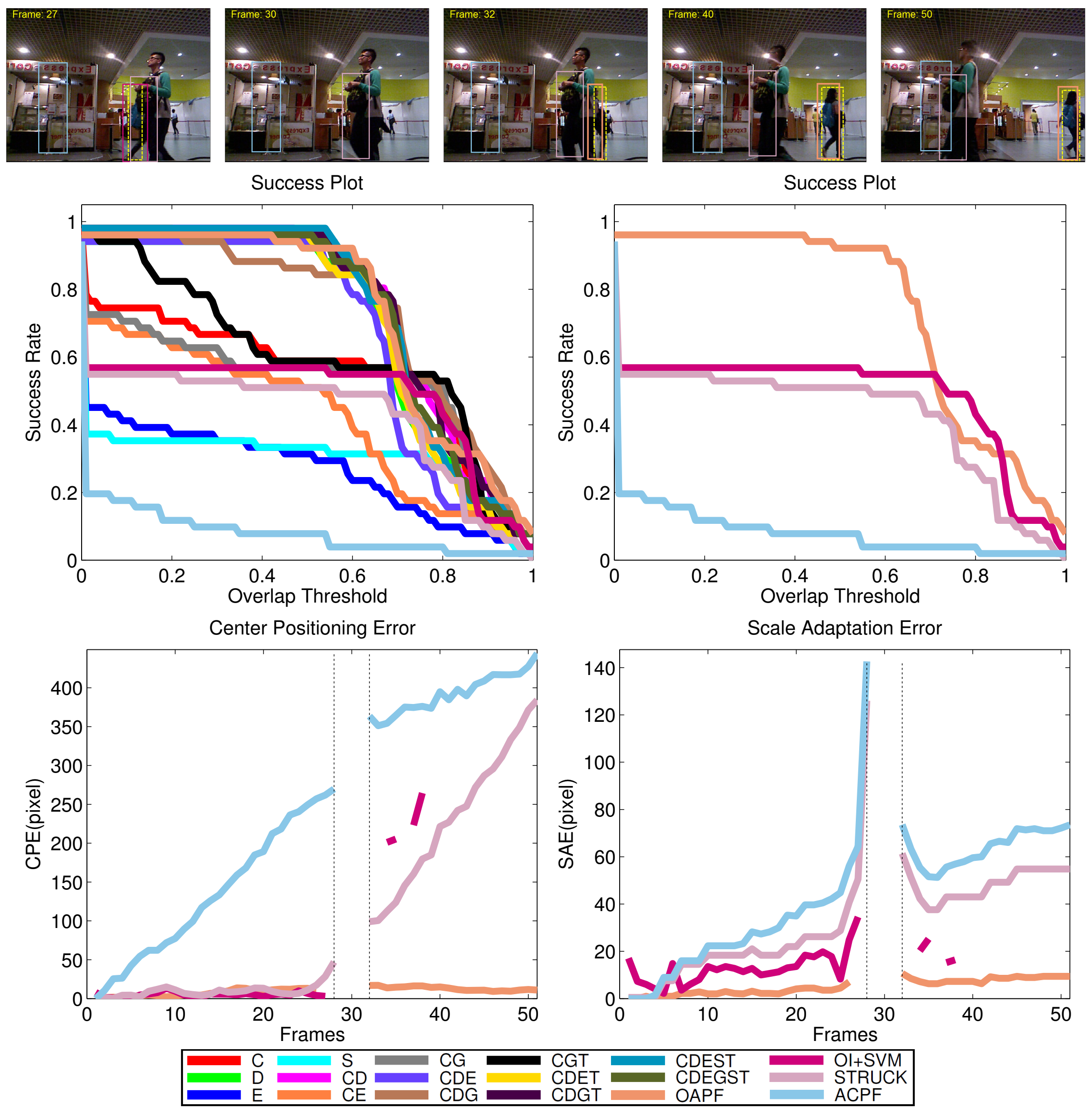


Figure 3.6: Qualitative (upper panel) and quantitative (lower panel) comparison of the proposed tracker *OAPFT*, *OI+SVM*, *ACPF* and *STRUCK*. A tracker disappears from the charts only when it has no output for that frame either due to target loss or occlusion state. The ground truth is marked with a yellow dashed line. This figure shows that the *ACPF* tracker was trapped in the background clutter and could not follow the target. *OI+SVM* lost the target and then tried to recover the target around frame 38 (see the line segments in second and third plots) but failed. In the performance plots, the proposed *OAPFT* shows the best tracking performance. Our proposed method shows small false positive and false negative rates in detecting occlusions (the starting value is close to 1 in the success plot), and it also recovers the target effectively after occlusion because it has a very small localization error (small values of central position errors and scale errors). The success rate of our trackers with different feature subsets is presented in the lower rightmost panel as well. Refer to the text for a detailed discussion on the effects of different feature sets and to the project webpage for further detailed analysis of all video sequences.

feature in our algorithm was inspired by [139] and is different from the $8 \times 8 \times 8$ RGB histogram used in [44].

Finally, our proposed tracker, *OAPFT* is compared against all of these trackers. *OAPFT* used the optimal feature set for the occlusion-flag-enhanced part of the tracker as well as the 2D confidence measure to enable the full potential of the tracker. This "2D confidence measure" is a part of the occlusion handling mechanism of the tracker, and by enabling it we prevent the model drift in long sequences, enhance scale adaptation and improve localization.

Table 3.2 compares the tracking algorithms and provides more insight into feature subsets in terms of the AUC and CPE values. In addition to the aforementioned criteria, the execution time of the trackers in frames per second (FPS) is presented. All of the experiments were conducted on a 3.50 GHz 64-bit Pentium IV computer on which the proposed tracker (*OAPF*) and *ACPF* were implemented in Matlab. *STRUCK* was implemented in C++, and *OI+SVM* employed a Matlab/C hybrid approach.

Table 3.2: Performance evaluation of the algorithms. The *occlusion-flag-enabled PFT* with different features in the upper panel, *OAPFT* with the optimized feature set and the 2D projection confidence measure (proposed) in the middle panel, and *OI+SVM* [108], *STRUCK* [27], and *ACPF* [44] in the lower panel. Each metric, averaged for all videos, is presented. False tracking (FT) is only applicable to the trackers that are capable of detecting full occlusions. Bold font shows the best value for each metric.

Tracker	cc[#]	AUC	CPE	SAE	MI	FT	MT	FPS
C	🟥	53.99	34.44	14.04	**0.0**	0.8	11.4	8.2
D	🟩	61.02	31.53	17.57	**0.0**	1.6	20.6	**14.4**
E	🟦	21.09	90.95	23.63	12.6	**0.0**	81.6	8.2
S	🟦	26.41	124.07	21.05	3.4	0.6	121.2	1.2
CD	🟪	68.52	16.26	15.83	**0.0**	0.8	2.4	8.3
CE	🟧	37.81	56.66	20.37	12.0	0.6	39.4	5.6
CG	⬜	54.76	34.61	12.72	**0.0**	1.6	26.4	6.1
CDE	🟪	58.20	24.63	17.93	2.8	0.8	**0.0**	5.7
CDG	🟫	69.31	15.85	15.15	**0.0**	0.8	2.6	5.9
CGT	⬛	55.14	28.06	13.66	**0.0**	1.6	3.8	3.8
CDET	🟨	63.48	19.49	14.81	**0.0**	1.4	**0.0**	3.7
CDGT	🟪	74.14	12.30	11.81	**0.0**	1.2	**0.0**	3.8
CDEST	🟦	54.94	34.20	16.25	2.8	0.8	29.6	1.0
CDEGST	🟩	72.94	12.03	11.25	**0.0**	1.6	**0.0**	0.9
OAPFT	🟧	**76.50**	**9.59**	**7.32**	**0.0**	2.4	**0.0**	0.9
OI+SVM	🟪	69.15	9.68	12.04	0.4	20.0	0.8	0.4
STRUCK	🟪	46.67	68.74	26.61	12.6	N/A	64.4	13.4
ACPF	🟦	27.55	90.38	35.27	12.6	N/A	31.0	1.4

[#] *cc – Color Code for the Tracker*

Figure 3.6 shows that an appropriate combination of features was crucial for successful tracking. A good example of these videos is illustrated in Figure 3.6, in which color, edge and shape features and their combination were not adequate and misled the tracker to frequently track the background or other similar objects. When the features covered different aspects of the object and teamed up with each other, the performance of the tracker was reasonably enhanced, see, e.g., E (AUC = 21%) versus CE (AUC = 37%).

For the set of five videos used in this experiment, the feature set CE (AUC = 37%) was worse than C (AUC = 53%), CDET (AUC = 63%) was worse than CD (AUC = 68%), and CDEST (AUC = 54%) was worse than CDET (AUC = 63%). These results suggest that the edge and 3D shape features (*SE* subset) were not effective in these scenarios. However, the performance of CDEGST (AUC = 72%) was significantly better than CDEST (AUC = 54%). Observing the trackers' behaviors, we found that in CDEGST, the CDG or CDT subset contributed substantially to the tracking, and the other features were effective in improving the tracking accuracy in the area dictated by the CDG or CDT subset.

Intuitively, we can characterize the features acting on a tracker into two groups, *(i)* those performing rough approximation of the target location, and *(ii)* those that fine-tune it once the rough target location has been identified. The features of the latter group are unable to localize the target in isolation, but once a certain ROI contains the target, their filter responses guide the tracker effectively. For instance, HOG feature (G) looks marginal by comparing the AUC of CD (68%) to that of CDG (69%). But by comparing the AUC of CDEST (54%) with that of CDEGST (72%), one can see the effective role of feature G in fine tuning the tracking (average CPE of CDEST was reduced by 22 pixels in CDEGST, and the SAE was reduced by 11 pixels).

These two groups of features, however, were not fixed or consistent over different videos. For instance, in videos with two crossing pedestrians in Figure 3.6, feature E was not effective in isolation, thus it dis-

Table 3.3: Comparison of the overall performance of the *OAPFT* with *PFT* using the same feature sets for the `child_no1` sequence, which has no occlusion.

Tracker	C	D	E	S	CD	CE	CG	CDE	CDG	CGT	CDET	CDGT	CDEST	CDEGST	All + 2D Confidence
PFT	**72.42**	**24.37**	**16.19**	10.21	**61.13**	**62.28**	**73.92**	**55.15**	**69.94**	**68.22**	**58.40**	**74.23**	46.81	**74.26**	**77.20**
Proposed	72.12	22.21	17.71	**11.57**	59.59	59.04	71.56	52.48	59.81	67.12	56.83	72.04	**46.90**	73.16	**77.20**

turbed the mediocre tracking of C as seen in the performance of CE. Another instance is the performance of feature T in two cases of this plot. While T has a little effect on CDET in comparison to its exclusion (CDE), it improves CGT over CG significantly. From Figure 3.6, it is seen that C and D are the features of the former group, and the rest of the features are from the latter group. Due to difference between videos, however, grouping of the features is different from the general grouping observed in Table 2 – for this sequence, G belongs to the second group, where in Table 3.2 it is a member of the first group.

Additionally, different features are less effective in some scenarios. For instance, in a low contrast scene with several same-color distractions the color feature is not adequate, or the HOG feature is not effective when tracking a pedestrian in a crowd. Still, the employed features cover different aspects of the target and can handle the vast majority of real-world videos. One or more feature may contribute substantially to tracking in a video, whereas the other features are not discriminative enough to emphasize the target location or may even distract the tracker. Benefiting from the normalization used in the likelihood definition, the former case was managed, whereas the results revealed the latter case still challenged the tracker, although it was largely undermined by the normalization. By introducing adaptive weights to the features (adaptive γ_i), as suggested by [147], the effect of different features are adjusted to applied videos and the results could be better for the individual videos.

Depth information, over all other features, plays an important role in resolving occlusion effects in our proposed tracker. The significance of the depth feature (D) is apparent by comparing CG (AUC = 54%) to CDG (AUC = 69%), CE (AUC = 37%) to CDE (AUC = 58%), and CGT (AUC = 55%) to CDGT (AUC = 74%) in Table 3.2.

Challenging a partial occlusion followed by a full occlusion, our binary-flag-enhanced trackers without the 2D projection confidence measure tended to increase the size of their bounding boxes, because they attempted to keep the boxes that covered the visible parts of the target(s), and larger boxes often had higher probability of having those included. After recovering from the occlusion, however, the size of the bounding boxes was quickly reduced to suit the target (Figure 3.6).

The proposed 2D projection confidence measure was effective in improving the tracking accuracy, as is evidenced by comparing the best feature set (CDEGST) without the confidence measure and our full tracker with the confidence measure (*OAPFT*) in Table 3.2. This scheme rendered the proposed tracker competitive with existing tracking-by-detection-type trackers such as *OI+SVM*. After careful investigation, we discovered that the background subtraction method used for calculating the 2D projection confidence measure assumed a static camera throughout tracking, and this underlying assumption degraded the performance of our full tracker in scenarios with moving camera such as `zcup_move_1`. Although our normalization of the likelihood seems to have eased this degradation, we need to equip the confidence measure with a more sophisticated background subtraction algorithm in the future.

In this experiment, the fastest processing time was achieved by the depth tracker (D), and the addition of other features, such as the 3D shape or HOG, required more time. The current implementation used Matlab, and the speed could be boosted by using GPU programming or low-level languages, such as C++, to promote the real-time tracking.

In summary, Table 3.2 and the plots in Figure 3.6 show that the advantages of the proposed tracker involve quick recovery from occlusion, better robustness against a tracker's failure (because the irrelevant particles are marked as occluded in a probabilistic manner), and a tighter grip on the target. The results indicate that *OAPF* with the optimal feature set outperformed the existing trackers, *OI+SVM*, *STRUCK*, and *ACPF*. Furthermore, the proper choice of features, with the help of the newly developed 2D projection confidence measure, improved the tracking performance in a variety of occlusion conditions.

3.4.3 Being Occlusion-Aware: the Trade-off

Next, we evaluated the switching mechanism based on occlusion awareness implemented in our *OAPF* by comparing it to the normal *PFT* that has no switching mechanism, using the same set of parameters (e.g., number of particles) and features. It is clear from Table 3.4 that *OAPF* has better overall performance given that the test videos include many occlusion situations. In our *OAPF* tracker, the population of particles is

Table 3.4: Comparison of the performance of the *OAPFT* with *PFT* using the same feature sets for the 5 annotated videos of the Princeton Tracking Dataset.

Tracker	C	D	E	S	CD	CE	CG	CDE	CDG	CGT	CDET	CDGT	CDEST	CDEGST	All + 2D Confidence
PFT	30.12	38.22	13.72	7.79	19.37	18.36	26.04	27.93	30.17	23.10	31.85	31.18	22.57	33.01	52.29
Proposed	**53.99**	**61.02**	**21.09**	**26.41**	**68.52**	**37.81**	**54.76**	**58.20**	**69.31**	**55.14**	**63.48**	**74.14**	**54.94**	**72.94**	**76.50**

divided into two, based on their binary flags. The tracking accuracy slightly decreased due to the decrease in the number of particles participating in the template matching, in particular when there was no external occlusion. The `child_no1` sequence in Table 3.3 was an instance of this description.

3.4.4 State-of-the-Art Evaluation

To examine the applicability of the proposed *OAPFT* tracker (i.e., with the optimal feature set CDEGST and the 2D projection confidence measure) to more general tracking scenarios, it was trained based on all five annotated videos, and the tracking results for the 95 unannotated videos were submitted to the evaluation website for the Princeton Tracking Dataset (`http://tracking.cs.princeton.edu/eval.php`). On the evaluation website, the tracking results were compared with state-of-the-art RGB and RGBD trackers, such as *OI+SVM* by [108] (whose variation constitutes the eight trackers indicated by letter (a) to (h) in Table 3.5) [3], *SAMF+Depth*, which is an advanced version of [148], *STRUCK* [27] and *TLD* [20] which showed the best tracking performance under occlusions according to the latest benchmarks [23,25], *large displacement optical flow tracker* [149] and other successful state-of-the-art RGB trackers such as *VTD* [60], *CT* [13], *MIL* [12], and *Semi-B* [65].

Table 3.5 shows the results, which are described under the classification of the target situations: target type, size, motion speed and motion type (which indicates whether the target actively moves by itself or is moved). Here, the total error was calculated by subtracting eq(3.14) from 1. Some of the trackers listed in this table were unable to detect full occlusions and provided bounding boxes to invisible targets. This is the reason why some trackers showed relatively high but constant *Type II* errors (≈ 6.94); due to the same reason, *Type III* errors were not applicable to these trackers. The trackers are ranked in each category (e.g., human target type, fast movement) and the most accurate tracker is marked in red, whereas the second and third best are marked in cyan and green, respectively. The average ranking over all of the categories is presented in the first column, upon which the table is sorted.

The results show that our proposed tracker performed well in most of the categories, and it was ranked first or second in almost all of them. Additionally, it exhibited the lowest *Type II* error, which was almost negligible ($< 1\%$), suggesting the algorithm could predict occlusions before their emergence and maintain the occlusion status for almost all of the frames that include occlusions. Having this conservative strategy, the tracker successfully protected the target template from getting contaminated during full occlusions, which are one of the weak points of normal *PFT*s. The reasonably low *Type III* error, moreover, indicates that our tracker recovered the target soon after occlusions. Our detailed observation indicates that this error mostly occurred when our tracker transitioned to an occlusion state in a premature manner.

*PFT*s are approximation-based trackers; thus, their tracking accuracy is inferior to that of tracking-by-detection-type trackers (e.g., *OI+SVM*). However, the *Type II* error rate reported for *OAPF* is comparable with other trackers, indicating that the proposed algorithm successfully enhanced the accuracy of the normal *PFT*. This improvement is mainly attributed to the use of the 2D projection confidence measure and the feature normalization procedure. Further investigations revealed that the 2D projection confidence measure enhanced scale adaptation and effectively preserved template in partial occlusions.

RGBDOcc+OF (called *OI+SVM* earlier in the manuscript) benefits from a direct occlusion detector using a histogram of depth, but the results suggest that it suffered from some false negatives (*Type II* errors) and even more false positives (*Type III* errors). In fact, *PCDet(f)* that used point cloud information instead of the histogram of depth, showed lower *Type II* error, implying there are other intelligent ways to incorporate the depth data into a feature to achieve lower *Type II* error.

When comparing our method as a general tracker and *RGBDOcc+OF* as a typical instance implementing the tracking-by-detection scheme, there was a large accuracy gap especially for the *animal*, *slow movement* and *no occlusion* categories, suggesting that the latter method was biased during training (e.g., more toward humans or fast motions). Furthermore, we discovered that our tracker tended to have errors

[3]Note that in the first experiment the version (a) of the trackers proposed by [108] was used (denoted by *OI+SVM*), because it had the leading performance on the dataset.

in scenarios with moving cameras (see section 3.4.2). However, the results obtained from the benchmark server did not include information about moving cameras, so they are not discussed here.

Table 3.5: The quantitative comparison of the state-of-the-art trackers for 95 test video sequences of the Princeton Tracking Dataset. The performance of trackers includes their tracking accuracy and occlusion handling as described in section 3.4.1. Best, second best, and third best results are illustrated with red, cyan, and green colors, and the table is sorted based on the overall tracker ranking over the different categories. These results were calculated by the online evaluation system provided by the dataset creators and available online: `http://tracking.cs.princeton.edu/eval.php`.

Tracker	Average Rank	Target Type			Targe Size		Movement		Occlusion		Motion Type		Error (%)			
		Human	Animal	Rigid	Large	Small	Slow	Fast	Yes	No	Passive	Active	Type I	Type II	Type III	Total
OAPFT (*proposed*)	**1.73**	**64.2**	**84.8**	**77.2**	**72.7**	**73.4**	**85.1**	**68.4**	**64.4**	**85.1**	**77.7**	**71.4**	**22.9**	**0.58**	**3.4**	**26.89**
RGBDOcc+OF(a)* [(d)]	**1.73**	**74.0**	**62.6**	**78.4**	**78.1**	**69.7**	**76.3**	**72.2**	**72.0**	75.2	**82.3**	**70.0**	**16.22**	2.28	**8.14**	**26.65**
DS-KCF [(d)] [150]	**3.09**	**67.0**	61.2	**76.4**	68.8	**69.7**	**75.4**	**66.9**	**63.3**	**77.6**	**78.8**	65.7	27.80	2.88	–[#]	**30.69**
RGBD+OF(b)* [(d)]	3.64	63.9	**65.3**	74.5	**71.5**	65.5	73.4	65.9	60.1	**79.0**	74.0	**65.8**	**24.65**	6.89	**0.37**	31.92
PCdet-flow(c)* [(d)]	5.36	50.5	51.6	72.7	63.4	55.5	73.9	53.0	55.0	64.4	75.5	52.7	29.53	1.48	10.06	41.08
SAMF+Depth [148] [(d)]	7.27	44.8	49.6	67.0	52.4	55.2	65.2	49.5	41.1	71.6	66.3	49.3	39.01	6.93	–[#]	46.04
LDP-SVT (unpublished)	7.91	46.8	58.8	55.8	52.9	52.3	56.3	51.1	41.0	68.4	58.2	50.4	40.52	6.93	–[#]	47.46
RGBOF(d)* [(d)]	8.27	47.1	47.0	63.6	47.4	57.5	56.7	51.8	46.9	61.9	63.4	49.3	39.88	6.93	–[#]	46.82
LDP-STRUCK (unpublished)	8.73	46.2	59.1	54.5	51.6	52.0	56.2	50.1	39.8	68.4	56.4	50.1	41.23	6.93	–[#]	48.17
PCdet(f)* [(d)]	10	40.6	42.1	61.7	55.4	43.6	58.5	44.8	46.3	52.0	64.9	42.6	32.50	**1.17**	17.62	51.31
Dhog(e)* [(d)]	10.64	43.3	48.3	55.9	47.2	50.3	52.7	47.5	38.4	63.5	54.3	46.9	44.10	6.93	–[#]	51.04
Struck [27]	12	35.4	47.0	53.4	45.0	43.9	58.0	39.0	30.4	63.5	54.4	40.6	48.69	6.93	–[#]	55.63
VTD [60]	12.55	30.9	48.8	53.9	38.6	46.2	57.3	37.2	28.3	63.1	54.9	38.5	50.10	6.93	–[#]	57.04
RGB(g)*	14.36	26.7	40.9	54.7	31.9	46.0	50.5	35.7	34.8	46.8	56.2	33.7	53.18	6.93	–[#]	60.12
CT [13]	15.73	31.1	46.7	36.9	39.0	34.4	48.6	31.5	23.3	54.3	42.1	34.2	56.69	6.94	–[#]	63.64
PCflow(h)* [(d)]	15.82	35.2	29.1	43.6	42.2	33.2	47.2	33.1	32.4	43.5	41.3	35.5	61.72	**1.2**	–[#]	62.92
TLD [20]	16.09	29.0	35.1	44.4	32.5	38.5	51.6	29.7	33.8	38.7	50.2	30.5	35.36	3.28	25.45	64.01
MIL [12]	16.45	32.2	37.2	38.3	36.6	34.6	45.5	31.5	25.6	49.0	40.4	33.6	57.59	6.94	–[#]	64.54
SemiB [65]	18.64	22.5	33.0	32.7	24.0	31.6	38.2	24.4	25.1	32.7	41.9	23.2	64.69	6.96	–[#]	71.66
OF [149]*	20	17.9	11.4	23.4	20.1	17.5	18.1	18.8	15.9	22.3	23.4	16.8	66.46	6.21	8.7	81.38

[(d)] *Trackers using depth information (RGB-D).*

[#] *These trackers are unable to detect full occlusions.*

* *These methods are benchmarked by [108]: —(a) SVM tracker with HOG and HOGD (i.e., HOG on depth image) features + Optical Flow + Occlusion Handling; —(b) Same combination w/o Occlusion handling; —(c) SVM Tracker with Point Cloud input + Point Cloud Optical Flow; —(d) SVM tracker with HOG and HOGD features; —(e) SVM tracker with HOGD feature; —(f) SVM Tracker with Point Cloud input; —(g) SVM tracker with HOG feature; —(h) Point Cloud Optical Flow;*

3.5 Conclusion

In this study, we presented a novel particle filter-based tracking algorithm with a particular interest in handling persistent and complex occlusions. The most important contributions of our method comprise managing persistent and complex occlusions by explicitly using an occlusion flag attached to each particle and treating occlusions in a probabilistic manner. The flag signaled whether the corresponding bounding box was occluded and then triggered the stochastic mechanism to expand the search area and to stop the template updating. Moreover, each particle in the proposed framework was evaluated by means of multiple features, so that its similarity to the target template was evaluated in terms of likelihood. Due to the newly developed 2D projection confidence measure in addition to the optimized feature subset selected from their pool, our tracker further exhibited high adaptability to changes in object scale and trajectory.

The experimental data revealed that our optimized set of seven features outperformed any other subset selected from the feature pool, and selecting effective features required keen attention to the aspects of the video sequence that the other features are monitoring. The performance of the tracker could be further improved by exploring high-dimensional features such as those provided by convolutional neural networks which are ground breaking in many computer vision domains [112] or introducing an adaptive weight to each feature channel [147]. Through intensive experiments using the Princeton RGBD Tracking Dataset, we found that our proposed tracker showed good tracking performance under variety of occlusions, outperforming the state-of-the-art RGB and RGBD trackers and the baseline *PFT*s. Further evaluation using the 95 blind test sequences registered in the same dataset also indicated that our tracker performed best among the 20 state-of-the-art RGB and RGBD trackers benchmarked on the dataset. These successful results are attributed to the good characteristics of our tracker: predicting emergent occlusion, preventing model drift, quick recovery from occlusions, compensating the impact of partial occlusion, and robustifying the feature fusion. The next steps toward a more robust and accurate tracker are to incorporate the confidence of each data channel into the tracking, to more explicitly use depth information, and to update the model adaptively with each observation.

4 Imitating any Tracker by a Unified Framework

"The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny...' "

— Isaac Asimov

4.1 Introduction

Every year many off-the-shelf trackers are introduced to address various visual tracking challenges. Typically, those studies intend to handle a single or a few of these challenges: appearance changes [9–14], abrupt/fast motion [15, 16], target/non-target confusion [19], long-term tracking [20], background clutter [21, 22], irregular scale changes [18], occlusions [128] and moving cameras. However, large-scale benchmarks [23, 25] suggested that *TLD* [20], *STRUCK* [27], *MIL* [12], *FBT* [28] and *SCM* [11] have overall superior performance dealing with different challenges. Many of these trackers have solved the issues by which other methods are troubled. While mastering all of these domains to construct the holy-grail of trackers is difficult, fusing these trackers to integrate their merits into a unified framework has not been very successful [151]. The biggest obstacles to construct such holy-grail trackers from currently realized ideas are the complexity of integrating mechanisms and contradictory objectives pursued by each of them.

In this study, we propose a framework, *MIMIC*, to imitate the behaviors of an arbitrary black-box tracker, with a relatively simple non-linear tracker benefiting from a pool of various features. This study is based on a premise that a linear combination of sufficiently expressive features along with a flexible but still simple non-linear observation model can roughly approximate the behaviors of many popular trackers.

Employing multiple features to track objects has been investigated considerably in visual tracking literature. Different frameworks like particle filters accommodate the feature fusion seamlessly [96]. Although automatic adjustment of the feature's weights has been focused in some studies [152], many studies opt to select an effective subset of implemented features to track with, yet the performance drastically depends on the way they approach feature extraction and selection. Different features may be obtained by applying various pre-defined templates on a single image patch [140, 153], be constructed by applying a particular function (e.g., scale) on a single image patch with different parameters [60], or a "feature pool" may consist of several heterogeneous features in [154].

Each employed feature lends its characteristics to the tracker, so that the collective behaviors of the tracker are affected directly by its active features and/or their corresponding weights. Some features have prominent effects on the tracker, which grants it certain degree of invariance against environmental changes. For instance, incremental PCA brings *IVT* [10] illumination invariance and sparse-coding-based features render the tracker robust against partial occlusions [128]. Thus, it is natural to attribute the behaviors of a specific tracker to its employed features. Moreover, complicated trackers behave in a way that seems to be closely emulated using several features fused together.

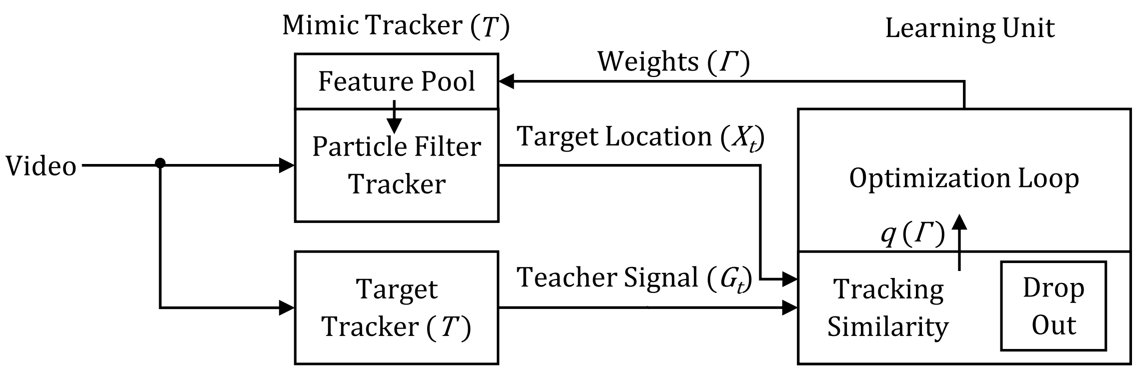


Figure 4.1: A scheme of the *MIMIC* tracker. The learning unit performs an optimization procedure based on an evolutionary algorithm enhanced with dropout algorithm to foster generalization.

The behaviors of many existing trackers can be imitated, by selecting the right set of features and/or proper weights for them. Without proper features, the observation model does not deal with the occlusion or sudden illumination changes for instance, hence fails to track the objects. In this study, we propose a unified framework to emulate the behaviors of these trackers by a proper mixture of feature responses. The features when combined in a tracking-by-detection tracker (such as *Ensemble tracker* [22]) or in an optimization framework (such as *Mean-shift tracker* [78]) lose the flexibility to mimic other trackers. Imitating trackers' behavior while they are dealing with different tracking challenges such as occlusions and illumination variations complicates the imitation problem even more. Such challenges are managed by a combination of specialized features and stochastic sampling in some trackers (e.g., in *IVT* [10], *MIL* [12] or *L1APG* [128]), so deterministic feature fusion methods do not work well on imitating their behaviors. Sophisticated trackers (e.g., *SCM* [11]), on the other hand, cannot be approximated with linear motion models or dense sampling. Linear observation models (e.g., in *Kalman Filter tracker* [69]) cannot emulate the non-stationarity of the scene observation, and moreover, a more complicated sampling and feature fusion are required to imitate highly adaptive trackers such as *STRUCK* [27]. Dense sampling in effect acts like the sliding window scheme in tracking-by-detection methods, hence cannot reproduce well the behaviors of probabilistic trackers. A non-linear non-Gaussian probabilistic framework such as the *multi-cue Particle Filter tracker* [96], is a natural choice to address these issues.

To validate our hypothesis – i.e., *a linear combination of proper features in a non-linear observation model emulates most of the complicated trackers* – a rich pool of features is provided to a particle filter tracker with a few hundreds of particles and a random walk motion model. The goal of this tracker is to imitate the behaviors of a specific target tracker closely, in the sense of tracking overlap on the unseen test videos. To achieve this, our tracker adjusts the weights of the features of the feature pool, based on the outputs of the target tracker while it operates in several videos used as the training data.

Our tracker, *MIMIC*, extracts a variety of features from the image frame, and weighs them in the observation model in a way to minimize the mismatch between its estimation of the object location and that estimated by the target tracker. These weights constitute the parameters to be tuned through supervised learning. Considering the high non-linearity of the objective function representing the mismatch to be minimized, this supervised learning was performed by an evolutionary algorithm which searches the high-dimensional parameter space for best combination of feature weights. To select a subset of effective features from the feature pool, the optimization should set the weights of many unnecessary features to zero, which resembles the feature selection in effect. This is done by imposing a L1 regularization term to the objective function.

To suppress over-fitting in this high-dimensional learning task, we employed "dropout", as the success of this regularization mechanism has been demonstrated in artificial neural networks [155]). By emphasizing on optimization over model ensemble, this algorithm investigates different combinations of the features and prevents over-fitting by approximating the training dataset by means of a virtual ensemble of the models employing variety of feature subsets. The proper combination of the L1 regularization and the dropout algorithm is expected to work well in the simultaneous task of feature selection and feature weighting, by avoiding over-fitting even when the number of available videos is limited (Figure 4.1).

In summary, our contributions in this study are two-fold:

- We proposed a unified framework to approximate the behaviors of an arbitrary tracker by employing a weighted subset of features selected from a feature pool in the observation model of our *MIMIC* tracker.

- We introduced several applications of this framework including "tracker identification" in which a black-box tracker is identified among other trackers based on its tracking results, and also a couple of novel approaches to perform tracker fusion.

Following this introduction, section 4.2 elaborates the proposed framework, and section 4.3 demonstrates the wide range of applications of this framework. After the proposed framework is discussed in section 4.4, our article is concluded in section 4.5 with the outcome of this study and some future directions to improve it.

4.2 Method

MIMIC is implemented as a *particle filter tracker* (hereafter, *PFT*) employing a linear combination of features in its observation model. Intuitively, *MIMIC* intends to mirror a given target tracker by maximizing the overlap between its tracking results and those by the target tracker, via adjusting the weights of the features selected from a predominantly given feature pool. Figure 4.1 depicts its basic scheme.

According to the *MIMIC* framework, a sequence of video frames I_t ($t \in \{1, 2, \ldots, \tau\}$) is provided to the system. A specific "target" tracker (i.e., the one that the *MIMIC* attempts to imitate) is assumed to track the target object along the sequence above, with its output (tracking result) at time t is denoted as G_t. Here, the target tracker is assumed to know the initial location of the object. The output of this tracking is a bounding box which is characterized by the object location $\langle G_{x,t}, G_{y,t}\rangle$ and its size $\langle G_{h,t}, G_{w,t}\rangle$. *MIMIC* also performs the tracking along the same frame sequence, that is, the object location is initialized in the first frame, and in the rest of image frames it tracks the object. More specifically, *MIMIC* estimates the current object location by employing an observation model based on the features listed in a feature pool $F = \{f_1, \ldots, f_n\}$ with their weights $\Gamma \equiv \{\gamma_1, \ldots, \gamma_n\}$. The object location in each frame t is estimated by *MIMIC* as a bounding box, denoted as X_t.

The major objective of *MIMIC* is by turning its weight vector Γ to enlarge the following "tracking similarity function" (or simply "overlap") that measures the similarity in the tracking results between the target tracker and *MIMIC*:

$$q_t(\Gamma) = \Psi(G_t, X_t(\Gamma)), \tag{4.1}$$

where $X_t(\Gamma)$ denotes the output of *MIMIC* which is dependent on the weight vector Γ, and $\Psi(.)$ is the normalized overlap between the two arguments:

$$\Psi(G_t, X_t) = \frac{|G_t \cap X_t|}{|G_t \cup X_t|}. \tag{4.2}$$

Here, $|.|$ denotes the area (in pixel number) of the defined region, and $\cap$ and $\cup$ are the pixel-wise intersection and union of given two image regions, respectively.

4.2.1 The Proposed Optimization Procedure

We employed "Evolution Strategy" [156] to perform supervised learning that tunes the feature weights in order to maximize the tracking imitation similarity function, eq(4.1). The non-linear and non-differentiable character of this objective function requires us to rely on derivation-free optimization techniques. Evolutionary algorithms provide a mature framework to optimize this type of objective functions; among those, Evolution Strategy (ES) is specialized in optimization in real-valued domains.

The main scheme of the ES optimization is presented in Algorithm 2. To *create initial population* (line 1), we create the first population or the first set of parents, consisting of μ solution vectors each generated from a Gaussian distribution around 1. Every solution vector is of n-dimensional and consists of real-valued weights that serve as the weights (γ_i) for the features in eq(3.3), and all features are initially equally active. *Cross over* combines two vectors randomly selected from the set of parents to generate two offspring (line 5) and repeats this process to produce λ offspring. The combination is essentially to make a new child by mixing the solution vectors of two parents; for example, by averaging them, by aligning components each copying from either parent, and so on. The randomly selected pair of parents do not necessarily succeed in producing children. The probability of cross-over success is governed by p_c; when two parents cannot reproduce, they are copied to the offspring population intact. This cross-over operation seeks to find better solutions by exploring the search space globally, and p_c balances the rate of exploration versus exploitation. After the cross over, the whole population are exposed to a *mutation* process (line 7). In this process, a solution vector in the population is randomly selected with probability p_m, then replaced with its mutated (i.e., slightly modified) version. This operator explores the search space locally and increases the diversity possessed by the population. In an operation called *survivor selection* (line 16), the next generation is prepared from the offspring only or from the union of parents and offspring. The former facilitates escaping from local minima, whereas the latter promotes the exploitation as well as preserving the diversity of the population. This process monitors the objective functions (in the context of ES optimization, it is called "fitness") of all the solution vectors to select μ solutions to construct a next population (line 17). A *fitter* solution with respect to the fitness is the one leading to a higher value of the tracking imitation similarity function, eq(4.1), but we also consider regularizing the ES optimization as discussed below. This procedure is iterated till it reaches the maximum number of generations, N_{gens}, when the final solution vector is obtained from fittest member of the final population. The implementation details of this ES optimization are presented in section 4.2.4.

4.2.2 Regularization to Obtain Sparse Weight Vector

We introduced two distinct regularization maneuvers, a direct inclusion of an L1 regularization term and a "dropout" algorithm. The L1 regularization term penalizes the number of selected features and keeps the weights in a reasonable bound. Accordingly, the objective function (eq(4.1)) is modified into

$$q_t(\Gamma) = \Psi(G_t, X_t(\Gamma)) - \alpha \sum_{i=1}^{n} |\gamma_i|, \tag{4.3}$$

where α is a regularization constant, determined by cross-validation in our case (see below). Without the regularization, the *MIMIC* is allowed to use any number of features, and tends to use all the available features as such high-dimensionality would provide more flexibility to the tracker for adapting the currently available data (in our case, the observed behaviors of the target tracker). This is a typical overfitting phenomenon, leading to degrade in generalization, often observed in supervised learning of artificial neural networks. The L1 regularization term encourages sparsity on the weights, hence is expected to maintain the generalization performance. It also emphasizes the role of more effective features, rather than approximating the target tracker by summing up small contributions of all of the features. However, this regularization is unable to explore the parameter space sufficiently, and often converges to local minima in which the selected weights for the features imitates the target tracker better than the other candidates, yet it differs from the best solution.

4.2.3 Incorporating Dropout

In addition to the well-established L1 regularization, we introduce another regularization, dropout, to our optimization. According to this regularization algorithm [155], a set of randomly selected features are masked in each iteration of optimization, and the weights corresponding to the masked features are ignored in every iteration of the optimization, i.e., they do not affect the fitness evaluation of a solution vector, and are kept intact during weight updates through cross over or mutation. More specifically, for each generation in our ES optimization (Algorithm 2: lines 2∼16), a mask $Z = \{z_1, \ldots, z_n\}$ is constructed by independently and randomly sampling all of its elements z_i (line 4). For every solution vector in the population, the masked features ($z_i = 0$) are preserved against ES operations (cross-over, mutation, and fitness evaluation) or the L1 regularization. Technically, the ES optimization tunes the weights of the unmasked features, while the masked features remain intact so that their unchanged weights are passed to the next generation.

By incorporating this dropout algorithm, the following objective function is maximized by the ES optimization

$$q_t(\Gamma, Z) = \Psi(G_t, X_t(Z^T\Gamma)) - \alpha \sum_{i=1}^{n} |z_i\gamma_i| \tag{4.4}$$

where $X_t(Z^T\Gamma)$ is the tracking output of our *PFT* with a feature weight vector Γ masked by Z.

The dropout algorithm is known to reduce overfitting by preventing complex adaptation to the training data [155]. In our implementation, dropout temporarily disable several features from the feature pool according to individual Bernoulli distribution $Ber(p)$ with the mean of 0.5, so that each feature is absent in the feature selection of *MIMIC* around 50% of the time, and different combinations of the features are explored in this way.

Dropout efficiently performs model averaging, which is a good way to reduce the generalization error by averaging the predictions virtually produced by a large number of different models. The standard way to do model averaging is to train many different *MIMIC*s and apply them to the validation data but this is computationally too expensive. On the other hand, the dropout algorithm would make it possible to train a wide variety of mimicking models in each generation of the ES optimization, within a reasonable computational time.

Applying an independent Bernoulli mask to every feature in the feature pool corresponds to applying uniform sampling distribution to the set of all feature subsets; this is much simpler than examining all the different feature combinations, whose number of constituents becomes 2^n.

Since the observation model is dependent on the summation of the feature-wise distance over the features (eq(4.6)), which is positive, the final feature weights when evaluating the performance of the *MIMIC* tracker after the ES optimization are all halved (Algorithm 2: line 18) in order to take an effective balance with the observation model in the fitness evaluation during the ES optimization (to compensate for the masking probability $p = 0.5$).

Since our *MIMIC* tracker, implemented as a *PFT*, behaves in a probabilistic manner, the weights of the features are evaluated as the average fitness when the *MIMIC* performs tracking for a single video N_{rerun} times (Algorithm 2: line 11~13); this averaging would make the tracker's fitness more reliable.

Algorithm 2: Mimic Tracker – Approximating a Target Tracker using a Particle Filter Tracker, a Pool of Features, and Drop-Out Feature Selection

```
   input  : Parameters $\theta$, Teacher signal $G$, Training videos $V$
   output: Feature Weights $\{\gamma_i\}_{i=1}^{F}$
1  $Parents \leftarrow$ Create Initial Population ($\mu$)
2  for $N_{gen}$ generations do
3      for $i \leftarrow 1$ to $n$ do
4          $z_i \sim Ber(\frac{1}{2})$                                  // Dropout mask
5      $Children \leftarrow$ Cross-Over ($Parents, \lambda, Z, p_c$)
6      $Population \leftarrow$ Survival Strategy ($Parents, Children$)
7      $Population \leftarrow$ Mutation ($Population, Z, p_m$)
8      for $j \leftarrow 1$ to $|Population|$ do
9          $W \leftarrow Population[j]$
10         for Video $v \in V$ do
11             for $N_{rerun}$ times do
12                 $X \leftarrow$ PFT ($W, \theta$)
13                 $E \leftarrow \sum_{t=1}^{T} q_t(\Gamma, Z)$
14             $Fitness\ (W, V) \leftarrow$ Average of $E$ in $N_{rerun}$ runs
15         $Fitness(W) \leftarrow \sum_{v=1}^{V} Fitness(W, V)$
16     $Population \leftarrow$ Survivor Select ($Population, Fitness, \mu$)
17 $b \leftarrow argmin(Fitness)$ , $W^* \leftarrow Population[b]$
18 $\Gamma \leftarrow \frac{1}{2} * W^*$
```

In Algorithm 2, V denotes the set of videos used for training the *MIMIC* tracker and θ encapsulates the tracker's parameters.

4.2.4 System Realization

The proposed method has several sets of parameters. Our *MIMIC* tracker is a type of *PFT*, and its dynamics are governed by noise variance of the motion model (Σ_{motion}). The number of particles used in *PFT* ($N_{particles}$) and some less important parameters are bundled into the parameter set θ. Finally, the number of reruns (see above) is determined by N_{runs}.

Another set of parameters determine the ES optimization procedure. In our implementation, we utilized averaging cross over and Gaussian noise mutation with adaptable variance (Rechenberg's one-fifth rule [156]) along with $\langle\mu + \lambda\rangle$ survival strategy (i.e., the population in the previous generation was kept in addition to the newly generated population to form the candidates of survival selection procedure). We also used a q-tournament survival selection scheme to improve the sampling fairness of the parameter space, avoid local minima, and increase the diversity of selected solutions. This selection involves running several tournaments among a few individuals chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected for the next generation. Selection pressure is easily adjusted by changing the tournament size, q. If the tournament size is larger, weak individuals – that might include good sub-solutions to the problem – have a smaller chance to be selected.

We also archived the best solution in the previous generation to maintain the stability of the evolution algorithm. These design details have been set by running the optimization procedure on a small yet rich training set with different settings of design, and observing the imitation performance of the population of the solution vectors as the generation proceeds. It should be mentioned that to implement the cross over operation with the dropout algorithm, two offspring are generated so as to share the value of each unmasked feature (i.e., the average of their parents for that feature) while inheriting the value of each masked one from either of the parents.

The population size (μ), the number of offspring (λ), the probability of operators (p_m and p_c), the parameter of the q-tournament survival selection (q), the number of total generations (N_{gens}) and the L1 regularization coefficient (α) govern the ES optimization procedure.

Among these parameters, μ, p_m, p_c, q, and α were determined by cross-validation. N (the number of particles), Σ_{motion} and λ were determined heuristically. The noise in the motion model, Σ_{motion}, which is described as a Gaussian random walk process, was set at three times of the largest change of the object to be tracked since the start of tracking, $N_{particles}$ was set to 1000, and $\lambda = 7\mu$ was used as suggested by [156]. To block the effects of the intrinsic parameters of our *PFT*, i.e., θ, they were set to appropriate constants. Additionally, N_{runs} was set to 5 based on the best practice introduced in VOT 2014 [125].

Another important point of our implementation is the design of the feature pool. Finding image features that are appropriate for reliably tracking the object has long been a hot topic in computer vision [140]. Specifically for *MIMIC*, which requires various effective features to shadow an even highly nonlinear target tracker, the construction of a rich feature pool is crucial. In our implementation, we provided 50 features for the system, all of which are supported by a wealth of literature, mostly inspired from object detection and image retrieval community. The list of these features are provided in the supplementary document (Appendix C). It should be mentioned here that the codebook of some features, such as BoVW (bag of visual words), was acquired solely on the training dataset without using the test dataset.

Our implementation was accelerated in two ways: *first*, it used a dictionary for efficiently calculating the fitness. The most computationally expensive part in the ES optimization is the calculation of the fitness. It involves the evaluation of all of the unmasked features for many particles of the *PFT* through all the frames of several training videos used for imitating the target tracker. To ease this computational expense, we utilized the fitness dictionary. When there was a pair in the dictionary whose entry closely matched the current query (i.e., the fitness of the *MIMIC* with its current feature weights on the set of training videos), the stored fitness corresponding to the matched entry was returned, instead of re-calculating the fitness for the query; otherwise a new pair of $\langle query, fitness\rangle$ was registered into the dictionary after calculating the fitness value. Monitoring the ES optimization by using a simple optimization problem, we realized that similar fitness values were often calculated several times during the ES iteration, so that a significant speed up could be expected by using this trick. A trained dictionary was used for every specific $\langle video, feature\ pool\rangle$ with the expense of a small amount of memory, so that the computational complexity was greatly discounted in practice.

The *second* speed up was achieved by a small tweak which was built upon the dropout algorithm. We skipped calculation of the masked features, and moreover, of the ones with very small weights. Since the mask was drawn with the probability of $p = 0.5$, half of the features were in average disabled in each operation in the ES optimization, which roughly granted a 2× speed up to the optimization. Since all the features are not equally expensive to calculate, we can add a penalty term to each feature based on its computational demand, as another regularization. This tweak is expected to reduce the computation time even more and will be explored in a future work.

4.3 Experiment

We first show the results in multiple experiments to verify the performance of *MIMIC* to shadow a target tracker. Later, we promote this scheme by demonstrating its applicability to different tasks.

In these experiments we employed two public datasets, TB-50 [24] and VOT14 [125]. The TB-50 dataset includes 50 video sequences, annotated with axis-aligned bounding boxes (AABB) and manually tagged with nine attributes, which represent the main challenges of visual tracking (e.g., illumination variations). TB-50 is used in the experiments that require training on videos with specific attributes. This dataset is also used in the experiments that demand more training data. The VOT14 dataset, originally prepared for *Visual Object Tracking Challenge 2014*, consists of 25 short video sequences showing various objects in cluttered backgrounds with different distractors. These videos were chosen from a large pool of sequences including the *ALOV++* dataset [28] based on clustering of visual features of objects and backgrounds in order to evenly sample the video pool. VOT14 is annotated with oriented bounding boxes (OBB) to precisely describe the target.

We evaluated the performance of our *MIMIC* using the sum of normalized overlaps between the *MIMIC* tracker and the target tracker, unless stated otherwise,

$$S(G, X) = \sum_{t=1}^{\tau} \Psi(G_t, X_t). \tag{4.5}$$

The rest of this section is divided into nine experiments summarized in Table 4.1 to answer the following research questions respectively:

1. *Is MIMIC capable of identifying underlying feature weights used in a target PFT, with the knowledge of the motion model and features used by the PFT?*
2. *Can MIMIC discover the employed features of an arbitrary tracker?*
3. *How closely can MIMIC approximate the behaviors of a target (black-box) tracker in various conditions?*
4. *Is it possible to identify a target tracker by observing its behaviors?*
5. *How accurately can MIMIC learn tracking from a human annotator?*
6. *How MIMIC handles different tracking challenges?*
7. *Can MIMIC surpass a set of target trackers by learning from their collective behaviors?*
8. *Can MIMIC learn from several target trackers at the same time?*
9. *Can several trained MIMICs collaborate?*

Hereafter, we denote a *MIMIC* trained to imitate a target tracker T as $\tilde{T}$.

Table 4.1: Summary of experiments.

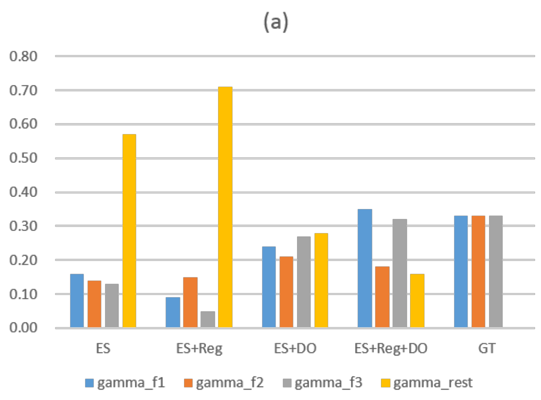
Goal	**Exp**	**Teacher**	**Features**	**Dataset**
Validation	1	Particle Filter	Known	VOT14
Validation	2	T	Known	VOT14
Discovery	3	T	Unknown	TB-50
Identification	4	$T_{1..m}$	–	TB-50
Learning	5	Human	–	VOT14
Discovery	6	Human	–	TB-50
Learning	7	Majority($T_{1..m}$)	–	TB-50
Learning	8	$T_{1..m}$	–	TB-50
Fusion	9	Human	–	VOT14

($T_1 \ldots T_m$ *are arbitrary trackers*)

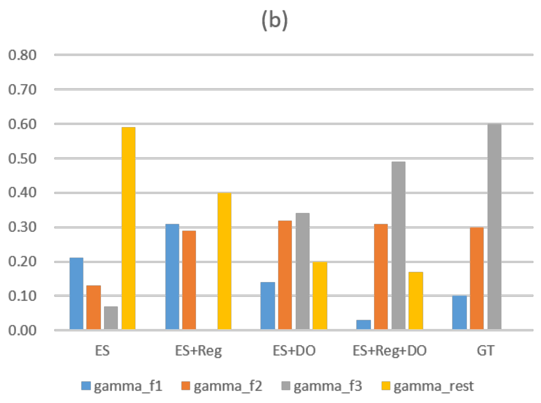
4.3.1 White-box Test

In this experiment, we demonstrate the capability of *MIMIC* to approximate a given *PFT* with known features by fixing all other factors of the tracker (e.g., motion model and parameters). For this experiment, we performed 5-fold cross validation on the VOT14 dataset and report the set of results with best overall training score.

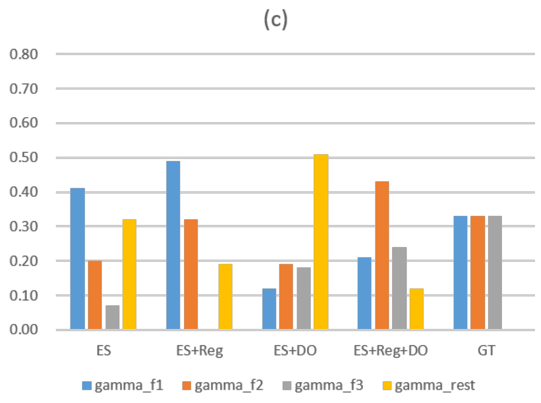
The target tracker is a *PFT* elaborated in section 3.2.1 employing three different features $f_i \in \{f_1, \ldots, f_n\}$ out of the feature pool and their weights w_{f_i} ($i = 1, 2, 3$). The weights of the *PFT* can be presented as a



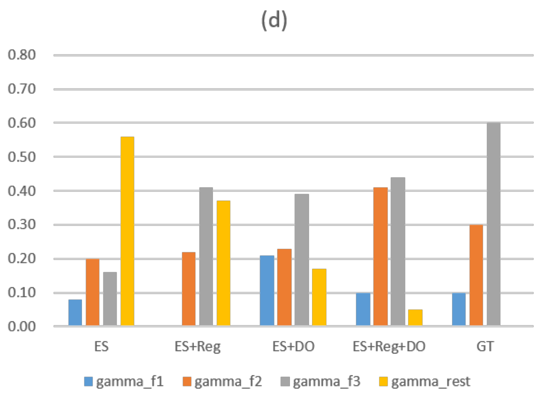
(a) equi-weighted independent features



(b) independent features with different weights



(c) equi-weighted correlated features



(d) correlated features with different weights

Figure 4.2: Experiment 1 – Feature weights with four kinds of optimization methods with different regularization, against the ground truth (*GT*). *ES* stands for Evolution Strategy, which is the basal optimization method to maximize the overlap between the *MIMIC* tracking results and those by the target tracker. *Reg* and *DO* stand for the L1 regularization and the dropout algorithm, respectively. In this experiment, three features were focused so that *GT* displays their values in feature setting. Each yellow bar depicts the contribution of the features other than the three features in the obtained weight vector. All feature weights are normalized. The *GT* weights for settings **(a)** and **(b)** were 0.33 and for **(c)** and **(d)** were 0.1, 0.3, and 0.6 for the three features. In settings **(c)** and **(d)**, the second and third features are correlated (see the main text).

vector $W \equiv \{w_i\}_{i=1}^{n}$ in which only w_{f_1},w_{f_2}, and w_{f_3} are not zero. The trained *MIMIC* is represented by its feature weight vector, $\Gamma \equiv \{\gamma\}_{i=1}^{n}$. We measured the dissimilarity Δ between normalized versions of these vectors based on the Chi-Square distance, which heavily punishes larger differences:

$$\Delta = \chi^2\left(\frac{\Gamma}{||\Gamma||}, \frac{W}{||W||}\right) = \sum_{i=1}^{n} \frac{(\tilde{\gamma}_i - \tilde{w}_i)^2}{\tilde{\gamma}_i + \tilde{w}_i} \ , \ (\frac{0}{0} \equiv 0) \tag{4.6}$$

where $\tilde{\gamma}$ and $\tilde{w}$ are normalized values of γ and w, respectively. We tested the following four scenarios in this experiment: (a) equi-weighted independent features, (b) independent features with different weights, (c) equi-weighted correlated features, and (d) correlated features with different weights. Table 4.2 shows the results of these experiments. The features employed by the target *PFT* for (a) and (b) were regular histogram of color (RGB; $8 \times 8 \times 8$ bins), gray-scale local binary pattern (256 bins) and bag of visual words of HOG (16 bins). These features are highly uncorrelated since they capture the target characteristics in color, texture and gradient domains. For (c) and (d), in addition to the regular histogram of color above, we used two highly correlated features, SIFT and SURF.

Figure 4.2 and Table 4.2 report the results of this experiment. Here, ES denotes the original evolutionary strategy employing the simple overlap as its fitness (eq(4.1)), ES+Reg uses the L1-regularized version of fitness (eq(4.3)), ES+DO denotes the use of dropout with the non-regularized fitness (in eq(4.1)), and ES+Reg+DO uses dropout with the L1-regularized fitness (eq(4.4)). The training score S_{train} and the test score S_{test} were evaluated in terms of the summed overlap eq(4.5). The overall results suggest that simultaneous usage of dropout and L1 regularization is not redundant, but beneficial for identifying the target tracker's feature weights. This is due to the fact that dropout disabled a set of features during the optimization to foster exploration over wide variety of feature combinations, whereas the L1 regularization encourages the optimization to use a minimal set within the unmask feature combination toward constructing the final solution.

In this basic experiment, we found that using both dropout and the L1 regularization is superior to the use of either of them (ES+DO or ES+Reg). The reason is that these two schemes regularize the ES optimization in different aspects. Dropout explores the space of discretized feature combinations globally while the L1 regularization encourages local sparseness over the real-valued feature weight space restricted by dropout. These different but cooperative role plays by the two methods correspond to the feature selection and feature weight tuning, required by our tracker's mimicking task.

The generalization ability of the sole ES optimization without regularization (ES) was poor, which is supported in Table 4.2 by the very low values of S_{test} for all of the feature settings (a-d) but the relatively high values of S_{train}; these values suggest the heavy overfitting to the training data. The ES optimization allowed the *PFT* to imitate the behaviors of the target tracker in the training set, in a way that is not generalizable to other videos, because the identified feature weights were apart from the ones used in the target tracker (signified by high Δ values). The L1 regularization was effective in increasing the overlap function both for the training and test datasets. However, the increased values of Δ in settings (a), (b), and (c) indicate that the features amplified by the L1 regularization were in fact different from those used in the target tracker. On the other hand, just adding the dropout algorithm to the ES optimization resulted in drastic increase in S_{test} as well as in decrease in Δ, showing this mechanism was quite effective in searching for the true set of features used in the target tracker. Since it might have not encouraged the weight vector to be further sparse, adding the L1 regularization worked better, as can be seen in the improvement in S_{test} and Δ by the ES+Reg+DO. Accordingly, from Table 4.2, we conclude that dropout significantly improved the ES optimization to seek a small subset of features used by the target tracker. Furthermore, the additional L1 regularization (ES+Reg+DO) was also effective in sparsifying the weight vector obtained by the ES+DO.

Closer look at the optimization curves (Figure 4.3) reveals that dropout promotes exploration by randomly masking so-far-dominant features and allows other features to join to imitate the behaviors of the target tracker. This exploration can be seen in large fluctuations of the best training fitness obtained by ES+Reg+DO. When the so-far-dominant features were masked, the training fitness decreased substantially as this loss could not be immediately compensated with other features. In long run, however, the training fitness consistently increased, though such exploration needed more time for the ES optimization to lead to good solutions.

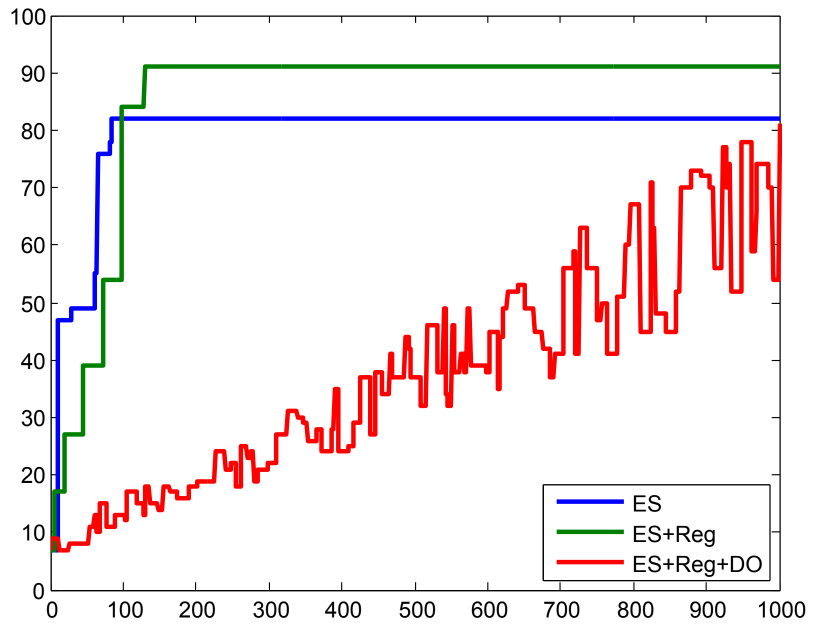


Figure 4.3: Experiment 1 – Training score versus optimization iteration. The quick fluctuations caused by dropout method indicates that the algorithm suppress one or more prominent features in the solution, to promote exploring for other alternative features.

Table 4.2: Experiment 1 -- Optimization curves, i.e., the progress of the training score (overlap) against the number of generations in the *ES* optimization. The obvious fluctuations caused by dropout indicate that this regularization suppresses one or more prominent features in the solution, to promote exploring into alternative feature sets.

Exp	Optimization	Δ^*	$S_{train}(\%)$	$S_{test}(\%)$
(a)	*ES*	0.222	81.9	27.7
	ES+Reg	0.411	92.6	34.3
	ES+DO	0.046	76.4	68.3
	ES+Reg+DO	**0.044**	81.2	**78.4**
(b)	*ES*	0.525	79.2	29.0
	ES+Reg	0.707	83.2	34.1
	ES+DO	0.079	71.3	77.6
	ES+Reg+DO	**0.049**	82.6	**81.2**
(c)	*ES*	0.209	86.0	42.1
	ES+Reg	0.361	90.4	51.8
	ES+DO	0.179	69.2	69.2
	ES+Reg+DO	**0.054**	83.2	**74.7**
(d)	*ES*	0.277	79.8	34.1
	ES+Reg	0.148	88.3	52.0
	ES+DO	0.092	65.2	64.2
	ES+Reg+DO	**0.041**	70.9	**79.1**

4.3.2 Verifying the Hypothesis

Popular trackers have employed different techniques and various features to accomplish good object tracking. Since our *MIMIC* tracker assumes that a wide variety of target trackers can be imitated, here we examined the hypothesis therein that *most of the trackers can be approximated closely by a PFT with an observation model that incorporates properly weighted features*. To this end, several popular trackers with high tracking accuracy were selected [23, 25]; they are *NCC* [157], *MST* [78], *IVT* [10], *MIL* [12], *STRUCK* [27], *CXT* [19], *CT* [13], *CSK* [14], *CSK* [14], *L1APG* [128], *TLD* [20], *SOAMST* [18], *FBT* [28] and *SCM* [11].

For each target tracker T, we trained a *MIMIC* $\tilde{T}$ to imitate it, so that the features with sufficiently large weights after the ES optimization, which are called significant features selected or SFSs in short, are reported. These features are compared with those described in the original article that presented the target tracker. Also, the trained *MIMIC* $\tilde{T}$ was tested against the target tracker T on a test dataset. For this experiment, we trained the *MIMIC*s on the VOT14 dataset by means of 5-fold cross-validation. Here we report the set of results from the fold with the best test score, S^*_{test}.

From Table 4.3, we see that when the training score was relatively high, at most three or four dominant features were found to satisfactorily approximate the target tracker, from which one or two had been included as the main features of the target tracker; the other features would have assisted our *MIMIC* tracker to compensate for the complexity of the target tracker. An interesting finding was, however, the features identified by our method were not necessarily similar to the ones deemed to be used by the target tracker. When mimicking the *MST*, a histogram of color (HOC) with $16\times16\times16$ bins (16 bins for each color channel) was selected, whereas in the original paper of the *MST* ([78]), different setting of $8\times8\times8$ bins was used. Note here that the HOC with the latter setting of a smaller number of bins also existed in our feature pool. This mismatch can be attributed to the difference in videos used for training between the two studies. The insufficient training and test scores for the $\widetilde{CXT}$ (Figure 4.4) occurred, because this target tracker used the context information of the non-target area whereas our *MIMIC* tracker could not exploit such information. The *MIMIC* failed to shadow the *TLD* tracker, indicated by the poor training and test scores for it. This failure can be attributed to the high degree of non-linearity possessed by this target tracker, while the *MIMIC* tracker was not able to reproduce such non-linearity only by combing the features. The *MIMIC* attempted to reproduce it by combining eight features, but the training score still remained low. On the other hand, the *MIMIC* was capable of overcoming the high non-linearity imposed by multi-instance learning

Table 4.3: Experiment 2 – Evaluation of the significant features selected (SFSs) by *MIMIC* to shadow various target trackers. The number of selected features are presented in the second column. When some of them are known to be used in the target tracker, they are listed in the third column (e.g., usage of local binary patterns (LBPs) and intensity correlations is explicitly mentioned in the article describing *NCC* [157]). The set of results from the fold with the best test score, S^*_{test} is presented for each *MIMIC*. Additionally, the training score (as the overlap) of these *MIMIC*s after 1000 ES generations are presented in the fourth column.

T	# SFS	Verified	$S_{train}(\%)$	$S^*_{test}(\%)$
NCC	2	LBP, Int. Corr.	94.8	92.4
MST	2	HOC	93.5	87.0
IVT	3	Intensity PCA	91.2	82.5
STRUCK	4	HOC, Haar-like	88.0	87.2
CT	2	Sparse SIFT	94.2	87.2
CSK	4	Color LBP	83.6	79.1
L1APG	3	Sparse SIFT	89.1	79.8
SOAMST	2	HOC	90.2	84.7
FBT	4	C-SURF	85.5	76.2
SCM	4	L1-Sparse	90.7	84.3
MIL	4	Haar-like	81.3	70.3
CXT	3	LBP	72.0	65.6
TLD	8	LBP	44.9	31.4

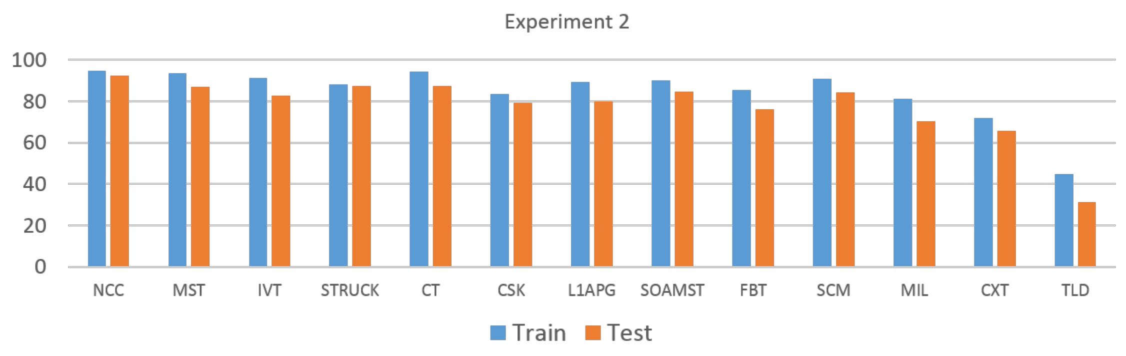


Figure 4.4: Experiment 2 – Training and test scores when imitating thirteen popular trackers, suggesting that the *MIMIC* tracker was able to approximate the behaviors of most of the popular trackers. However, the proposed tracker was not suitable to shadow highly non-linear trackers such as *TLD* and those exploiting information of non-target area, such as *CXT*. Interestingly, the *MIMIC* trackers performed 83.3 frame-per-second in average on a Pentium 4 Core i7 @ 3.50GHz with Matlab/C++ implementation, which was found to be one order-of-magnitude faster than some of the popular trackers listed in this table.

implemented in the *MIL* tracker to some extent.

During and after optimization, the *MIMIC* tracker followed the object in a quick pace, because it was implemented as a *PFT* with few hundreds of particles and 3∼4 features. In our Matlab/C++ implementation on a Pentium 4 Core-i7 @ 3.50GHz PC, it realized 83.3 frames per second (FPS) in average, which was found to be much faster than many of the target trackers. This significant speed up was achieved at the cost of simplifying the tracker (see section 4.3.9).

4.3.3 Black-box Test

In this experiment, we demonstrate whether *MIMIC* can imitate tracker's behaviors even when facing different tracking challenges: illumination variation (IV), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-plane rotation (OPR), out-of-

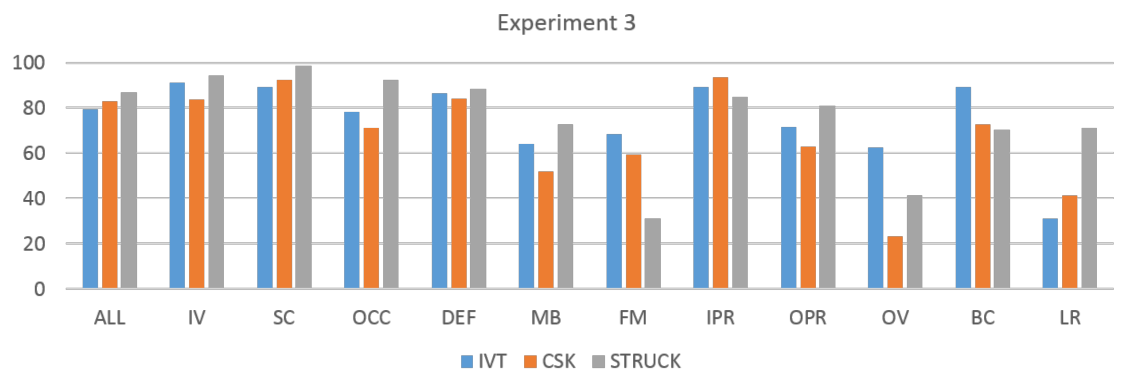


Figure 4.5: Experiment 3 – Results of tracking videos having each attribute from various challenges. Test scores are presented as the overlap in tracking results between *MIMIC* and one of the three target trackers in percent. The attributes are: illumination variation (IV), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-plane rotation (OPR), out-of-view (OV), background clutter (BC), and low resolution (LR).

view (OV), background clutter (BC), and low resolution (LR). Every tracker has its own specific internal mechanism to handle one or more of these challenges; for example, *IVT* [10] uses incremental PCA to handle illumination variations. Aiming to approximate an arbitrary target tracker, our *MIMIC* is expected to reproduce such an internal mechanism of the target tracker by well combining the features in hands. This reproducibility was examined by comparing with the behaviors of three trackers; *IVT* [10], *CSK* [14], and *STRUCK* [27], each of which is represented as T, when they tracked in the TB-50 dataset.

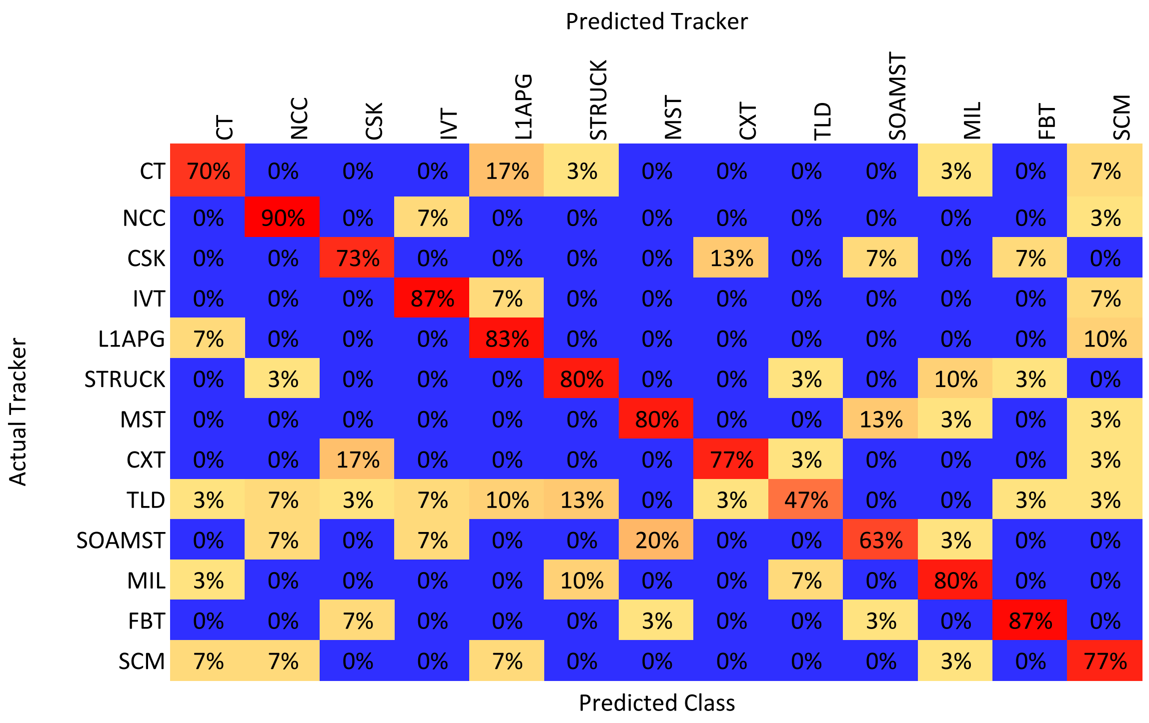
To examine a specific attribute A (A can be any of the above-mentioned challenges), we kept one video that contained attribute A as a test video, and trained the *MIMIC* on the remaining 49 videos. We repeated this procedure for all the videos containing A and averaged their test scores, as reported in Table 4.3. Figure 4.5 shows the test score, $\Psi(T, \tilde{T})$ (eq(4.5)), achieved by the *MIMIC*s, each of which was trained to shadow either of the above-mentioned three target trackers.

Figure 4.5 demonstrates that our *MIMIC* can handle specific challenges in a very similar manner to the target tracker. For instance, *IVT* was designed to effectively handle the illumination variation and our *MIMIC* trained to imitate this tracker showed good reproducibility on the videos attributed to the illumination variation. The test scores of *MIMIC* in low resolution (LR), fast motion (FM), and motion blur (MB) attributes were, however, low, because the training videos in the VOT14 dataset rarely include those labeled by these attributes. Furthermore, the target trackers themselves did not well address LR, FM, MB, and OV challenges, therefore there were few clues to learn from under those challenges. For example, the *CSK* relies on the target appearance and is not able to accommodate the cases where the object is not completely visible (OV) or the object's observation is of low resolution (LR); when imitating the *CSK* in either condition of OV or LR, the *MIMIC* suffered from a low test score.

4.3.4 Tracker Identification

Approximating different trackers with a unified framework enables us to perform tracker identification, i.e., to distinguish a particular tracker from many others in a tracker pool, by observing the way it tracks the objects. In this experiment, we intend to identify a tracker whose tracking results are given as query. More concretely, we trained a set of *MIMIC*s for each tracker in the tracker pool based on nine folds of a 10-fold segmentation of the TB-50 dataset. This procedure resulted in 9 feature weight vectors for each tracker, which are all stored in a tracker signature gallery. When we observed tracking results by an unidentified tracker, we trained another *MIMIC* to obtain a feature weight vector (hereafter called query signature) that well represents the unidentified tracker's behaviors. Finally, we identified the tracker applying a K-nearest neighbor classifier with $K = 4$ and Δ as dissimilarity measure (eq(4.6)) to the gallery of the feature weight vectors; the input to the classifier was the query signature. Due to the stochastic nature of our ES optimization, we repeated this classification task three times, which yielded 30 identification outcomes (3 times by 10 folds of the dataset) for each tracker to be identified. The confusion matrix of this tracker identification task is presented in Figure 4.6.

Figure 4.6 clearly shows that our *MIMIC* framework can address identification of most trackers. Less


Predicted Tracker
Actual Tracker
Predicted Class

	CT	NCC	CSK	IVT	L1APG	STRUCK	MST	CXT	TLD	SOAMST	MIL	FBT	SCM
CT	70%	0%	0%	0%	17%	3%	0%	0%	0%	0%	3%	0%	7%
NCC	0%	90%	0%	7%	0%	0%	0%	0%	0%	0%	0%	0%	3%
CSK	0%	0%	73%	0%	0%	0%	0%	13%	0%	7%	0%	7%	0%
IVT	0%	0%	0%	87%	7%	0%	0%	0%	0%	0%	0%	0%	7%
L1APG	7%	0%	0%	0%	83%	0%	0%	0%	0%	0%	0%	0%	10%
STRUCK	0%	3%	0%	0%	0%	80%	0%	0%	3%	0%	10%	3%	0%
MST	0%	0%	0%	0%	0%	0%	80%	0%	0%	13%	3%	0%	3%
CXT	0%	0%	17%	0%	0%	0%	0%	77%	3%	0%	0%	0%	3%
TLD	3%	7%	3%	7%	10%	13%	0%	3%	47%	0%	0%	3%	3%
SOAMST	0%	7%	0%	7%	0%	0%	20%	0%	0%	63%	3%	0%	0%
MIL	3%	0%	0%	0%	0%	10%	0%	0%	7%	0%	80%	0%	0%
FBT	0%	0%	7%	0%	0%	0%	3%	0%	0%	3%	0%	87%	0%
SCM	7%	7%	0%	0%	7%	0%	0%	0%	0%	0%	3%	0%	77%

Figure 4.6: Experiment 4 – Confusion matrix of the tracker identification task. The set of videos in the TB-50 dataset were divided into 10 folds, so that a leave-one-out procedure was used to obtain the gallery of tracker signatures. In each iteration of the cross-validation, a *MIMIC* tracker was trained based on each of the nine training folds, hence we obtained nine tracker signatures. By changing the target tracker in the tracker pool one by one, we constructed the gallery. Similarly, the test fold was converted into one query signature and this signature was used for identifying the tracker after regarding it as unidentified. The gallery forms the search space in which a 4-nearest neighbor classifier classifies the query to identify the tracker based on its query signature. Thus, the high classification accuracy indicates the ability of the proposed method to extract the tracker signature in terms of the feature weight vector to figure out the functionality of the tracker. To block the random effects of the cross-validation procedure and the proposed scheme to extract the tracker signature, the above process was repeated three times resulting in 30 identification outcomes for each tracker. In this confusion matrix, the query trackers index the rows of the matrix.

important features to reproduce each tracker were assigned zero weights by the corresponding *MIMIC* tracker as a result of regularization. Such a sparsified and hence exaggerated signature was effective in identifying an unidentified tracker even based on its behaviors in an unseen video. One shortcoming of this tracker identification method is that it's performance is upper-bounded by the imitation ability of our *MIMIC* tracker. When imitating some trackers such as the *TLD*, our *MIMIC* exhibited rather poor imitation ability, leading also to significant degradation of the identification performance. An interesting observation is that most of the trackers (7 of them) confuse the signature of *SCM* with their own signatures.

4.3.5 Learning from the Expert

Many studies have tried to understand the mechanisms underlying human object recognition (e.g., using psychological and psychophysical experiments like eye tracking [158]), and many features have been designed to imitate the human visual processing (e.g., edge detectors simulate L1 neurons in the early visual system [159]). Here, we try to allow our *MIMIC* to follow two different human annotators, present the resulting sets of features, and discuss the findings.

For this experiment, we employed the VOT14 dataset, and trained a *MIMIC* tracker based on its public annotation that is in the form of oriented bounding boxes (Part **a**). Additionally, we trained another *MIMIC* tracker based on our self-made annotation that is in the form of axis-aligned bounding boxes (Part **b**). We performed 5-fold cross-validation on the videos, and reported the intersection of *SFS*s (for definition, see section 4.3.2) obtained from the 5 folds.

For Part **a**, *MIMIC* achieved the test score of 39.52% (best score of all 5 folds), and the common set of

Table 4.4: Experiment 6 – Most occurring features in the *MIMIC*s trained with videos labeled by a specific attribute among variety of challenges. The test score is averaged over leave-one-video-out procedure for each attribute, and SFSs are reported when they appeared in more than 50% in the leave-one-video-out procedure.

A	IV	SC	OCC	DEF	MB	FM	IPR	OPR	OV	BC	LR
Avg. Test Score	53.5	61.2	12.4	52.1	23.2	39.2	32.8	42.8	17.2	38.2	19.2
Most Frequent SFSs	Intensity LCC	HOC, SIFT	-	-	HOC	HOC, Corr.	SIFT	Sparse-L1	-	-	-

*SFS*s included bag of HOGs, histogram of colors ($8 \times 8 \times 8$ in RGB) and bag of Gabor filters. For Part **b**, the best test score was 58.36% and the common set of *SFS*s included histogram of colors ($8 \times 8 \times 4$ in HSV), bag of SIFTs, and bag of visual words. Since the *PFT* employed in the *MIMIC* utilized axis-aligned bounding boxes as its observations, moderately low training and test scores were expected in Part **a**, still the features selected by our *MIMIC* consistently included histogram of colors (although from different color spaces). However, the *SFS*s of the trained *MIMIC* on human behaviors are not limited to the reported features, suggesting that the *SFS*s have substantially depended on the training videos and features in the pool, thus we cannot draw any direct conclusion about the features used by human annotators in general. From these results, we found that it is likely that human annotators cannot be imitated simply by linearly combining the features in our pool, which calls for designing better features and more advanced feature fusion methods.

4.3.6 How Experts Resolve Challenges

In this experiment, we examined which features are more effective in explaining human tracking behaviors when facing various tracking challenges. Here, we left out one video with attribute *A* from the TB-50 dataset as the test sequence, and trained our *MIMIC* to imitate the manual annotations attached to the other videos. The average test score for each attribute *A* is reported in the correspondent column of Table 4.4. The intersection of *SFS*s over the leave-one-video-out is also reported. This intersection was found to be empty for many attributes, because some features were indeed beneficial for imitating the human annotations but they were not consistent over the training videos. Besides, due to random nature of our ES optimization and dropout, this list also suffers from stochasticity. Having this in mind, Table 4.4 shows the *SFS*s which were obtained by *MIMIC* in more than 50% training tasks for each attribute.

The results seen in Table 4.4 are found hard to be interpreted in some cases. What is certain is that there is no dominant feature to reproduce human behaviors for most of the challenges. Sparse coding and SIFT feature are found among the reported *SFS*s; this is not surprising since these two features were inspired by architecture and functions of human visual systems – the former was built based on neural sparsity in the visual cortex and the latter was inspired by scale and rotation invariance found in higher layers in the visual cortex. On the other hand, the low test scores for some attributes such as occlusion, out-of-view, low resolution, etc. suggest that there are no sufficiently good feature in our feature pool (although it contains 50 of the most popular features presented so far in visual tracking literature) to imitate the ways a human tracks the object in videos while it is occluded, partly out of view, or have low resolution. Although this table presents the most occurring features when the *MIMIC* tried to shadow the human behaviors, we admit that for drawing a conclusion the size of the used dataset is barely enough. To this end, a large dataset of videos with an exhaustive occurrences of different challenges (e.g., all different types of occlusion) should be constructed. This experiment might have drawn the attention to the necessity of such a large-scale dataset to better understand the mechanisms underlying human visual processing.

4.3.7 Imitating the Majority Vote of Teachers

So far, *MIMIC* has been used to imitate behaviors of a teacher, either of a target tracker or a human annotator. Here, we take one step further and examine if *MIMIC* can surpass its teachers by being trained by all of them simultaneously. To this end, we should first define a single target bounding box $G_t^{\dagger}$ from the tracking results of k teacher trackers, $G_t^1, \dots, G_t^k$.

Each tracking algorithm has its own merits and demerits depending on its design principles and goals. So, tracker-level fusion is a plausible way of improving the tracking accuracy. Classical fusion (average, maximum, minimum, and median of bounding boxes) [160], combining target PDFs [161], weakly supervised learning from the outputs of different trackers [162], weighted tracker fusion [22, 161], using predictors to use different trackers [163], combining bounding boxes [164], and pixel-level tracker inter-

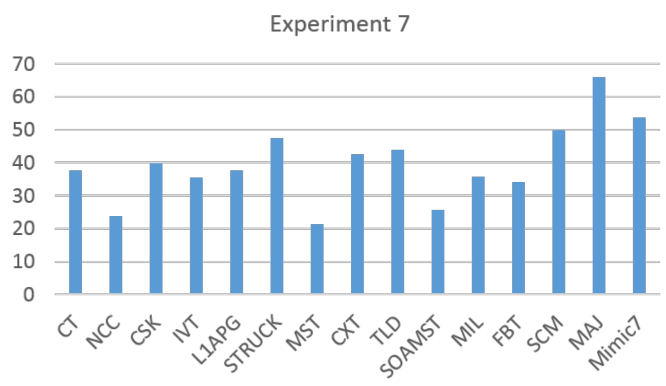


Figure 4.7: Experiment 7 – Training a *MIMIC* to imitate the majority voting of many trackers. The left thirteen bars show the performance of *MIMIC* when shadowing the thirteen target trackers, each measured in terms of area under the curve (AUC) of the success plot (for more details, see the main text). *MAJ* denotes the ≥ 5 majority voting of all the trackers, and *MIMIC7* denotes the *MIMIC* trained to imitate the *MAJ*.

sections [165] have been proposed to perform tracker-level fusion. Among them, "majority voting" has been known as one of the most robust techniques to fuse the results of several trackers [151], hence it was employed in this experiment.

According to the majority voting, the resulting bounding box is determined as the union of areas that a certain number of individual trackers have commonly detected as presenting the object. For a "$\geq N$" majority voting, the fusion bounding box corresponds to the rectangle $G_t^\dagger$ which contains all the areas in which at least N trackers agree upon the object's presence.

In this experiment, the video sequences were given to all of the thirteen trackers listed in Table 4.3 and their tracking results were fused by the ≥ 5 majority voting into the $G_t^\dagger$. This fused (virtual) tracker in turn served as the teacher tracker for our *MIMIC*. To measure the tracker's overall performance on all the frames in all the videos, we counted the number of frames in which the success degree S' was larger than a given threshold t_o. The area under the curve (AUC) of the resulting plot (S' vs. t_o) is reported as the performance of the *MIMIC* tracker here. The success degree $S' = \frac{1}{\tau}\sum_{t=1}^{\tau}\Psi(G_t^*, G_t^\dagger)$ was defined as the normalized overlap between the output of the fusion tracker $G_t^\dagger$ and the ground truth G_t^* for each video. We performed 5-fold cross-validation on the TB-50 dataset on all trackers, and the average AUC score over the 5 folds is reported in Figure 4.7.

By using *Mimic* to approximate the majority vote of the teacher trackers, the need to compute the result of all trackers is waived. The results of Figure 4.7 reveal that not only the majority vote technique achieves the best results among all the examined trackers, but also its approximation by *Mimic* obtained high robustness and notable score by learning from the best trackers in that area. The high non-linearity underlying the majority voting imposed the big score gap in the results between the majority voting and its *MIMIC*, yet the *MIMIC* seemed to have resolved this non-linearity partially by well combining the features in hands.

4.3.8 Satisfy All

In this experiment, we pursued another approach to learn from many teachers in an integrated manner. Here, we used the regularized fitness eq(4.4) in which the original tracker-wise similarity eq(4.1) was replaced with the following across-tracker similarity consistently for imitating all the trackers:

$$\Psi^{All}(\Gamma) = \sum_{a \in A} \Psi^{\tau}(G^{(a)}, X(\Gamma)) = \sum_{a \in All}\sum_{t=1}^{\tau} \Psi(G_t^{(a)}, X_t(\Gamma)) \tag{4.7}$$

where All is the list of all the teacher trackers and $G_t^{(a)}$ is the output of a tracker a at image frame t. This experiment was performed on the TB-50 dataset by means of attribute-based cross-validation described in section 4.3.3; average test scores are depicted in Figure 4.8.

This training of *MIMIC* based on the across-tracker similarity is essentially the task of finding a good balance between all of the aspects of different trackers. As it is demonstrated in Figure 4.8, actually,

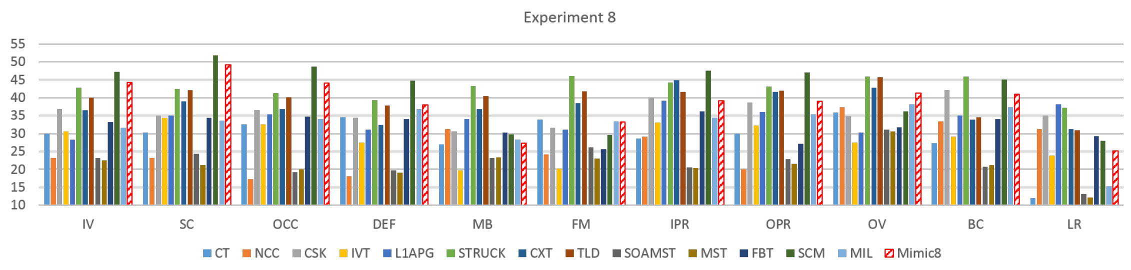


Figure 4.8: Experiment 8 – Training *MIMIC* based on the across-tracker similarity to allow it to learn from many teachers simultaneously (*MIMIC8*). The vertical axis shows the area under curve (in percent) of the success plot, which is different from Figure 4.4 that has shown the training and test scores of the dedicated *MIMIC*s.

the trained *MIMIC* never surpassed the tracker specialized for each attribute. However, it exhibited an acceptable performance on all attributes, hence its overall performance was superior to all of the teacher trackers. This is in contrast with such trackers as *TLD* which is generally a reliable tracker but behaves poor in the presence of background clutters (BC). It is also contrary to the tracker like *IVT* that has very good performance on a few categories (e.g., occlusions; OCC) but a slightly degraded one on other categories. Besides, the performance of the *MIMIC* trained based on the across-tracker similarity (i.e., *MIMIC8*) was rarely better than state-of-the-art trackers like *SCM*. This can be attributed to the inclusion of generally weak trackers such as *MST* into the calculation of the across-tracker similarity. However, this harmful effect can be avoided by carefully selecting the teacher trackers in the calculation of eq(4.7).

4.3.9 Agglomeration

In this last experiment, after training individual *MIMIC*s to follow each of different target trackers, we attempted to get a better tracker by integrating the results of those trained *MIMIC*s. We obtained thirteen *MIMIC* trackers each of which was trained to imitate one of the thirteen target trackers stated in section 4.3.7. A 5-fold cross-validation procedure on the TB-50 dataset was used to train the *MIMIC*s, and the one with the highest test score was selected as the representative of the trained trackers. Then, we applied a ≥ 5 majority voting on those representative trackers to obtain a collaborative result of all the thirteen *MIMIC*s. The test score of the representative *MIMIC* trained by each target tracker and the tracker based on majority voting by all the representative *MIMIC*s evaluated on all the videos in the TB-50 dataset are presented in Figure 4.9. In this figure, the ≥ 5 majority voting tracker (the blue bar in the *MAJ* column) is the same as in Experiment 7, and the shaded bar denotes the majority voting by the thirteen representative *MIMIC*s.

As can be seen in Figure 4.9, $\widetilde{NCC}$ and $\widetilde{MST}$ performed even better than their corresponding target trackers. The reason is the features selected to approximate the original trackers. Along with the *PFT*, these features were effective in tracking better than the original simple tracker in some scenarios. In Figure 4.4, the imperfect test scores for the trackers suggest that their *MIMIC*s cannot imitate the original trackers perfectly; however in some cases, the *MIMIC* may even get better than its teacher by using effective features from the pool. Another finding from Figure 4.9 is that the majority voting by *MIMIC*s worked significantly better than all of the constituents that had joined the vote. This is due to the fact that each *MIMIC* used 3~4 features that would have been specialized in imitating certain aspects of the corresponding target tracker. When integrated together in the level of trackers, they lent their advantages to the collective tracker, without affecting it severely in the scenarios in which those features were less effective.

4.4 Discussion

In this study, we proposed a new perspective of “unified tracker imitation framework”. Such framework is suitable to approximate the behaviors of trackers that are intelligent but demand heavy computations, by means of a computationally cheap online tracker.

Additionally, the proposed framework made it possible to imitate humans in various tracking scenarios. This can serve as a tool to monitor different features as they try to capture real-world. Furthermore, this framework is capable of combining several trackers (that perform well in some scenarios but poor in the others), leading to a super-tracker that performs well in all of the scenarios. As an implementation realizing such a framework, which is fast, intuitive, robust, and scalable, we presented in this study *MIMIC*.

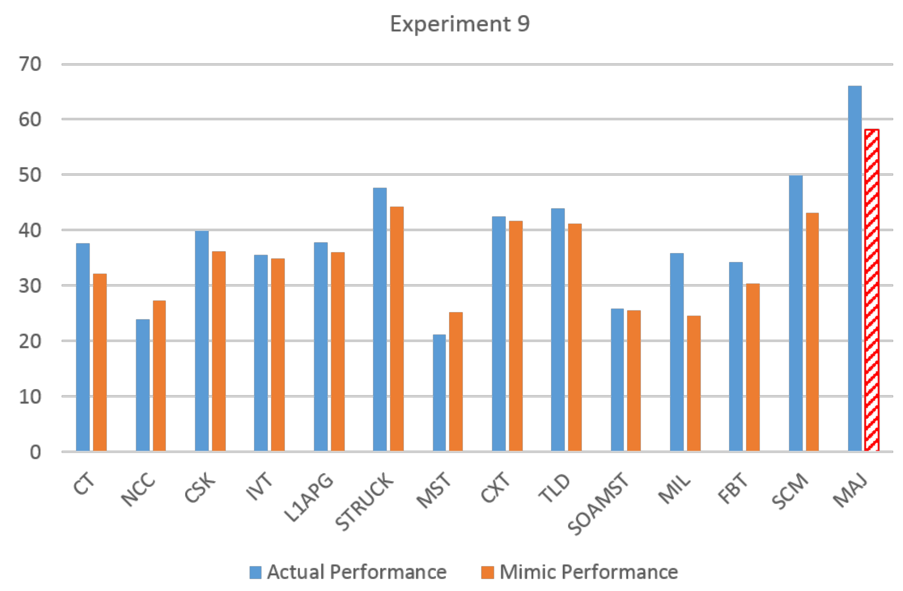


Figure 4.9: Experiment 9 – The majority voting by actual trackers (blue bar in the *MAJ* column) and by representative *MIMIC* trackers to each imitate each target tracker (red dashed bar in the *MAJ* column). In this plot, the tracking performance of the thirteen popular trackers and their *MIMIC*s were evaluated on the TB-50 dataset (the blue bars are identical to those in experiment 7). The performance was measured by the area under curve of the success plot, explained in section 4.3.7.

The selection of essential features from a given feature pool is the most important component of the framework, because it greatly affects the way of the tracker's behaviors in different scenarios, and moreover, the computational cost to use all the available features in an online manner is still too high for most computers. The feature selection in turn is a difficult task because it involves evaluation of 2^n different feature subsets from a pool of n features. This problem quickly becomes intractable as the number of features increases. In addition, the large cardinality of feature combination and the parameter space defined on it increases the risk of over-fitting to the dataset used for training.

There are two approaches to select the features automatically: *(i)* to treat the inclusion status of each feature in the final tracker as a hidden state and then to estimate it in a Bayesian manner, and *(ii)* to consider the feature selection problem as involved in the parameter estimation problem.

Online feature selection is the task of choosing an appropriate subset of features from a feature pool during tracking, in order to adaptively improve the tracking performance [11, 65, 166–169]. This is especially useful when the tracker wants to adapt to the very present video. However, such methods are greatly affected by abrupt changes in the scene (e.g., occlusion or illumination changes) or videos. As a result, the features selected by such online feature selection tend to over-fit to each video, which can be harmful for approximating the behaviors of the tracker in wide variety of scenarios. To reduce the over-flexibility of the online feature selection, it is possible to shrink the learning rate and to apply slow temporal dynamics to the learner, yet there is no general guideline for appropriately setting the learning rate.

Parameter learning, as a broader task of feature selection and feature weight tuning, offers several approaches that could be employed to tackle the "tracker imitation" problem [170, 171]. Our proposed approach differs from these methods in at least two points: *(i)* we avoid over-fitting in the parameter learning by using dropout whereas the framework of maximum likelihood estimation tends to use all the features, thus it is incapable of performing feature selection; *(ii)* treating the feature weights as the hidden states of the particle filter becomes to demand excessive computational power as the number of the features increases.

In general visual tracking, the relationship between the input (i.e., the observation) and the correct output (i.e., object size and location) is complicated, so that the tracker needs to keep enough features to model it accurately. There are many different settings of the feature weights that can imitate the behaviors of a target tracker when performing on a training dataset, especially if there is only a limited set of training videos available. Each of these weight vectors makes different prediction about the behaviors of the target tracker on the test dataset. Almost all of these weight vectors would perform worse on the test dataset than on the training dataset [155]. By utilizing dropout, we effectively overcame this difficulty as it was justified by the extensive experiments done in this study.

There are number of studies to apply dropout to various tasks, but to our knowledge, the current study is the first one to employ dropout in general visual tracking. Additionally, other studies have often used the likelihood as their objective function, while we followed a supervised learning scheme in this study and tried to reconstruct the desired outputs (here, the behaviors of the target tracker in a test dataset) even from a limited set of training data.

Imitating a tracker is more challenging than normal parameter learning. This difficulty arises because modelling a tracker is indeed a problem in the time domain. Thus we need to optimize our objective function, the fitness, whose input is a time-series. This character significantly constrains the number of training samples if we want to obtain the results in a reasonable time. Another related problem would be that the underlying large complexity of the objective function increases the chance of over-fitting even more, so inclusion of techniques to avoid the over-fitting deems to be crucial.

MIMIC, as an instance of the unified tracker imitating framework proposed in this study, is an online, scalable, and robust tracker. Built upon a particle filter tracker, this framework can fuse an arbitrary number of features registered in a feature pool. The effective feature selection has enabled this tracker to imitate most of the popular trackers closely while complying the real-time requirements. To achieve these preferable characters of *MIMIC*, the dropout algorithm [155] during the offline training phase played a crucial role. By promoting exploration over the model ensemble, dropout ensures the optimization procedure to find a generalizable *MIMIC* tracker for wide variety of target trackers even from a limited number of training videos. In our proposed framework, the feature selection and feature weight tuning were unified, so that an evolutionary algorithm optimized the non-convex fitness function. This optimization framework facilitated the integration of dropout as it was elaborated in Algorithm 2.

4.5 Conclusions

In this study, we introduced a novel approach to imitate the behaviors of an arbitrary target tracker, called *MIMIC*. We constructed *MIMIC* based on the assumption that most of the popular trackers can be imitated by a linear fusion of several features to be embedded in the observation model of a non-linear tracker. We employed a particle filter tracker and a rich pool of features to verify this hypothesis. The extensive experiments justified the overall validity of this hypothesis, although highly non-linear trackers and those using information of non-target areas were found not to be well approximated. This approach was effective in discovering the features underlying successful trackers, imitating the trackers under challenging scenarios (such as occlusion or illumination changes), tracker identification, imitating human annotators, and learning from multiple trackers.

To further evaluate our approach, we conducted a series of experiments and the findings suggested that *MIMIC* is capable of shadowing many of the state-of-the-art trackers, while it sheds light on the possible inner mechanisms of black-box trackers. The results showed that using dropout to prevent over-fitting was essential to appropriately imitating the given trackers. The data also revealed that usage of the L1 regularization in addition to the dropout is not redundant, as these two kinds of regularization worked in a cooperative manner for the feature weight tuning and feature selection. Using dropout was especially beneficial for the scenarios in which the target object underwent complicated transformations which were hardly explained by a set of simple features. Furthermore, when taught by several teachers, *MIMIC* outperformed its teacher trackers by collectively using their behaviors during its training. This opens a new research front of designing more advanced techniques to fuse variety of trackers.

This study can be further expanded into several directions in the future. Incorporating feature complexity into the fitness function would enable the system to find less computationally expensive solutions to imitate a tracker. Using a richer feature pool, and inclusion of features obtained from automatic feature extraction routines from the pool can be another way to enhance the system. Furthermore, in long tracking scenarios, the future tuning and selection can work along the target tracker to replace it with a fast interpretable *MIMIC* tracker with acceptable tracking accuracy.

5 Data-driven Occlusion Mask

"Once we accept our limits, we go beyond them."
— Albert Einstein

5.1 Introduction

Earlier, we review different techniques of compensating the occlusion effects on the trackers. Sophisticated motion models (section 2.3.2), robust representations (section 2.3.1), strong prior-knowledge (section 2.3.4), keen background suppressors (section 2.3.3) and redundant camera settings (section 2.3.6) were studied to manage the impact of the occlusions on the observations and tracking. Most of these techniques, however, have critical limitations for being deployed in real-world situations as they require strong models, environment understanding, or multiple cameras [61].

Despite of their importance, occlusion detection and reasoning are rarely addressed in the literature mainly because of vast variety of possible occlusion scenarios, the "shape variation-occlusion" dilemma, and the physical model shortcomings. Many techniques traverse the observations multiple times to infer the occlusion statuses of the objects [79]. In addition, many of such trackers do not meet the real-time requirements of online tracking because of heavy computation. In multi-target tracking scenarios, the information about other objects in the scene facilitates the occlusion reasoning process [32, 33, 86, 111] while many self-occlusions, background occlusions and non-target object occlusions are not handled by such methods. In single-target tracking, however, this information is not available and an online tracker is limited to the information obtained from previous and current observations in the tracking sequence. Monitoring abrupt deviations from appearance template in the main target candidates [58] or all of the candidates [109], histogram of depth in RGB-D sequences [108], the coefficient of trivial templates in *L1T* [129], and the ratio of foreground pixels to the number of pixels in the target's bounding box [34, 87] are some of the few known ways to detect occlusions.

Recently Kwak et al. [61] proposed an active occlusion detection and handling based on patch likelihoods. In their method (hereafter called OC, which stands for occlusion classifier) the target is divided into 4×4 regular grid cells, and the state of occlusion for each cell is determined using a trained classifier. The feature vector of this classifier is composed of the likelihood of 17 patches obtained for each cell, and a binary $\{occluded, not\ occluded\}$ state is obtained for each cell. The obtained mask, is then applied to a *L1-minimization tracker* and is proved to be successful in handling different occlusions including a challenging split-occlusion. This technique, however, heavily depends on the training data. For instance when one of the patches have a unique and rare appearance (e.g., purple soccer shoes of a player with a green uniform), it is useful to emphasize more on that distinctive patch as it is a good feature to track [140], but OC classifies such patch as occluded and discards such a useful feature. Besides, imagine a patch color pattern is given to the classifier as a positive sample. If the same color pattern is present in the occluder or background of a cell (which is very common when the scale of bounding box is not proper), a false negative (no-occlusion state when there is an actual occlusion) will be generated by the classifier. Furthermore, OC utilizes a single

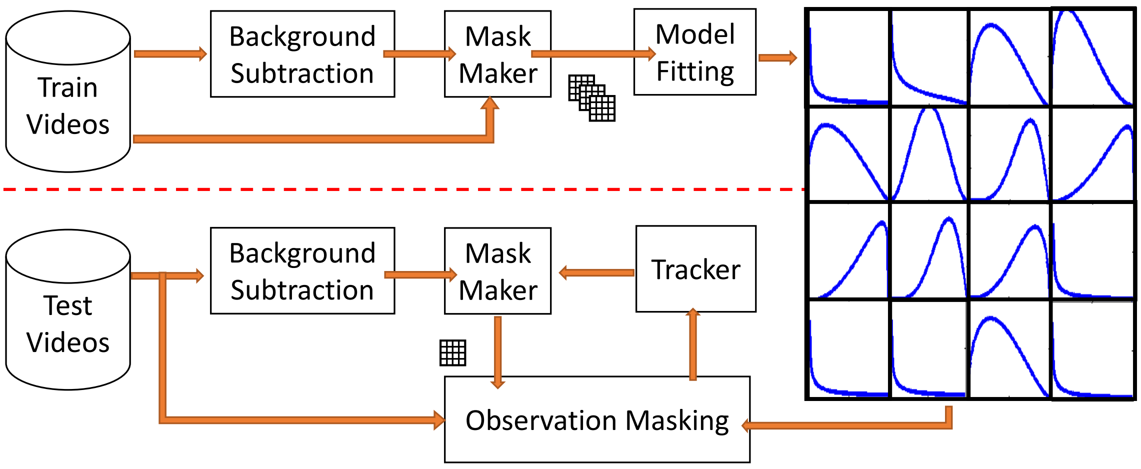


Figure 5.1: Schematic of the system **(top)** Training phase **(bottom)** Test phase

classifier to detect occlusions in the entire regions of an observation. This is not effective, since the patches closer to the boundary of the target usually involve more background pixels. In this case, several of the proposed patches in [61] are not applicable. This leads to an erroneous feature vector construction, which in turn results in an unreliable occlusion status for the cell. In another viewpoint, OC is also sensitive to its decision boundary as it is demonstrated in [61]. Above all, the crisp nature of binary status in OC, prevents the template from a smooth update, which in long term interrupts the function of the tracker. This is critical since the input features of the classifier are made from patches designed to overlap with their neighbor cells, and the non-smooth template pushes the obtained mask toward having more occluded cells.

To address these limitations, we propose an occlusion probability mask (OPM) for reliable tracking under occlusion. This technique is built upon the ratio of foreground pixels to all pixels in a bounding box initially proposed in [87]. We call this ratio as foreground ratio hereafter. To detect occlusions, the proposed method learns the patterns of foreground ratio based on the data collected during tracking with and without occlusions. These patterns are learned using parametric probability density functions (PDFs). Since the foreground ratio is lower around the boundaries of the target [44], we train different PDFs for different parts of the target. To this end, similar to [61], we decompose an observation into a regular 4×4 grid and train an independent PDF for each of them. This procedure yield an OPM that detects occlusions, assigns confidence to different parts of an observation, and guarantees reliable template update. As a byproduct of the method, the scale of the target bounding box is determined accurately, since the OPM punishes boxes with excessively large bounding boxes as they include more background pixels. Our occlusion probability mask can be integrated in many template-based or part-based tracking algorithms provided that the host algorithm shares the observation model with OPM.

In the next section, we incorporate the proposed algorithm in multi-cue particle filter tracker [96] as one of the general frameworks for possible host trackers. Section 5.3, explains the construction of the occlusion probability mask. Later Section 5.4 illustrates the experimental design and the embedding of the proposed algorithm in other trackers such as *L1T* [134], *IVT* [10], and *L1APG* [128]. This section also elaborated on the experiments of this study which were conducted on challenging occlusion scenarios and presents the results that highlight the superior performance of *OPM-enhanced L1APG* among the benchmarked occlusion-handling techniques. Finally, this *L1APG-OPM* pair is evaluated against popular visual trackers and the results of this pair are reported. The manuscript is concluded with a summary, a discussion and several directions for feature studies.

5.2 The Occlusion Mask

Our objective is to design a robust and general add-on algorithm to provide a base tracker with occlusion detection and effective template update. This algorithm should be embedded to the base tracker easily. To this end, we formulate the occlusion-state-dependent observation model. The proposed algorithm, the occlusion probability mask (OPM), models an observation as a weighted sum of its local sub-regions based on their degree of occlusion. In this section, we demonstrate how it can be embedded into the particle filter tracker (section 3.2.1) as an instance of appearance-based trackers.

Algorithm 3: Multi-cue PFT enhanced with occlusion Probability Mask

```
input  : Target initialization $X_1$
output: Target state $\hat{X}_t$
Create N particles
initialize target template $M_1^i$
for $t \leftarrow 2$ to $T$ do
    for $j \leftarrow 1$ to $N$ do
        Apply motion model
        Calculate occlusion mask for particle $j$ (eq(5.1))
        Calculate template matching likelihood (eq(3.3))
    Normalize particles' likelihoods
    Approximate new target state $\hat{X}_t$ (eq(3.4))
Update template (eq(5.2))
Resample particles
```

5.2.1 Occlusion Detection and Handling

Particle filter tracker [133] is conceptually robust against partial occlusions because of having several particles to sample the target object. This tracker is expanded to use kernels [44], fuse multiple cues [96], and achieve robustness against appearance changes [10]. Although having multiple particles mitigates occlusions to some extent, it suffers from template update with occluded observations that impair the tracking after occlusions. *L1-minimization tracker* [134] employs sparse minimization to handle the corruptions of the observation (due to noise, occlusion, etc.) but the performance of the occlusion handling is not satisfactory in practice [24, 61]. By considering the occlusion status, several studies improved the performance of *particle filters* [81, 109, 128, 129]. These methods either observe the sum of particle's likelihood [81, 109] or the energy of the trivial templates in sparse representation [128, 129] and stopped the template update once an occlusion is detected. This strategy is not sufficient to handle persistent partial occlusions as they corrupt a part of template gradually without alarming the tracker. Additionally stopping the template update completely in the case of an obvious but partial occlusion is not efficient, since the remaining observable parts of the target, sometimes involve critical information about the tracker location or appearance changes during the occlusion. Recently, [61] proposed an occlusion mask to undermine the impact of occluded parts of observation and published promising results about handling partial occlusions as well as other types of occlusions.

However, as discussed in Section 5.1, the occlusion mask proposed in [61] has four major issues: *(i)* sensitivity to training data, *(ii)* uniform treatment of different parts of the observation *(iii)* sensitive decision boundary in the occlusion classifier *(iv)* binary occlusion status of each cells which in turn results in *(a)* template grid cells inconsistency and *(b)* discarding critical information in partially occluded cells. We tackle these issues by using a robust mechanism for occlusion which is not target- or video-specific. A mechanism which generalizes the classifier to a probability distribution to avoid the problems caused by binary status of each cell, adopts a dedicated distribution for each part of the observation, and is not sensitive to variations in training data as an advantage of using a distribution.

The target region is decomposed into a regular $C \times C$ grid. We denote $Y_{t,c}^{(j)}$ as the image patch at cell c of the bounding box induced by particle $X_t^{(j)}$ from the observation Y_t. For each cell $c \in \{1, \cdots, C^2\}$, we define $\rho_{t,c}^{(j)}$ as the probability of the cell being occluded ($\rho \in [0, 1]$). To compensate the effect of occlusions, an observation should take the reliability of its subregions into account. Therefore, the features derived from this observation could be seen as a weighted sum of feature vectors of its subregions (here, the grid cells):

$$\phi_i(Y_t^{(j)}) = \sum_{c=1}^{C^2} \rho_{t,c}^{(j)} \phi_i\left(Y_{t,c}^{(j)}\right) \tag{5.1}$$

No smoothing function should be applied on the probability mask and its border should remain crisp. This is due to the fact that good features in the neighborhood of a occluded cell are getting punished by any kind of smoothing.

In the case of severe occlusion, i.e., when more than 30% of the cells in an observation have likelihoods less than threshold τ_1, we do not trust the information contained in the observation, and rely solely on the motion model to continue tracking to the next frame. This is due to the fact that heavy occlusion impedes an exact posterior estimation from the observation. The motion model is obtained from the history of target motions in the past τ_h frames.

5.2.2 Template Update and Feature Selection

Empowered by the occlusion probability map, the template can be updated intelligently since the feature vector in eq(3.13) is indirectly affected by embedding occlusion mask. In addition, to prevent a risky template update, the update is only performed if less than 30% of the cells in an observation have likelihoods less than threshold τ_2 ($\tau_2 \geq \tau_1$).

$$M_{t+1}^i = \begin{cases} M_t^i & , \mathcal{K}(\tau_2) > 0.3C^2 \\ (1-\xi_i)M_t^i + \xi_i \phi_i(\hat{Y}_t) & , otherwise \end{cases} \tag{5.2}$$

where $\mathcal{K}(\tau_2) = \sum_{c=1}^{C^2} u(\rho_{t,c}\phi_i(\hat{Y}_{t,c}) - \tau_2)$ is called as the occlusion ratio and $u(x)$ is 1 when $x > 1$ and 0 otherwise. In this equation, the value of template is obtained by eq(5.1) and the corresponding $\rho_{t,c}$ is induced by the estimated target location at frame t.

Degeneracy is a phenomenon in which a template contains no good features to track [140]. Decomposing a bounding box into a $C \times C$ grid, increases the risk of having degenerated cells. Such cells are indicated by a high ratio of degenerated pixels, and are not useful for tracking. Discarding degenerated cell in calculations of observation likelihood, similar to [61, 140], improves the localization performance and is implemented in our OPM-enhanced trackers. Algorithm 3 elaborates the mechanism of multiple-cue particle filter tracker with occlusion probability mask.

5.2.3 Generalization

Kwak et al. [61] described the procedure of embedding an occlusion mask into *L1T* [134] and *IVT* [10] trackers. Similar procedure can be used for *L1APG* tracker [128] that also deals only with intensity values. Here, we generalized this procedure to embed occlusion mask into a particle filter tracker with arbitrary features. Note again that this approach is applicable on any other template-matching-based tracker. The trick is to break the observation into subregions, apply the confidence probability on the local observation model, and agglomerate those local observations to reconstruct a robust observation.

5.3 Occlusion Probability Distributions

In this section, we choose an indicator to measure the reliability of a particular region of the observation, fit a parametric probability distribution on the empirical measurements of that indicator, and efficiently calculate the confidence for query observations during the test phase.

5.3.1 Observation Reliability Measure

Zhao and Nevatia [87] proposed to use foreground ratio (defined in Section 5.1) as an indicator for occlusion. We denote the foreground ratio (i.e., the ratio of foreground pixels to all) of cell c with $\zeta_c \in [0, 1]$. This value is calculated for every particle j at time t, $\zeta_{t,c}^{(j)}$. When dealing with foreground ratio, high values indicate that the image patch is almost filled with foreground and it is likely to be in the center of a target bounding box. In contrary, low values are probably from the the boundaries of the object where it is adjacent to background pixels or occluded partially [44]. These observations root from the inspection of the 65 training video (to be introduced in section 5.4) and many other real-world video sequences. Of course, tubular, halo, and elongated objects are not included in this observation, yet we achieved acceptable results using this assumption to build our tracker.

To obtain the foreground ratio we have employed the background subtraction method described in [172], which exploits motion coherence to handle moving cameras. This method combines the appearance modeling with motion field using belief propagation, and is robust against partial occlusions. However, heavy occlusions interrupt the tracking of this algorithm. If less than 30% of the C^2 cells in an observation have likelihoods less than threshold τ_3 (i.e., $\mathcal{K}(\tau_3) < 0.3C^2$), we translate the background mask obtained in previous frame to a new position using the motion history explained in section 5.2.1. Once the occlusion ratio went back to not-occluded zone, the background subtraction algorithm is re-initialized.

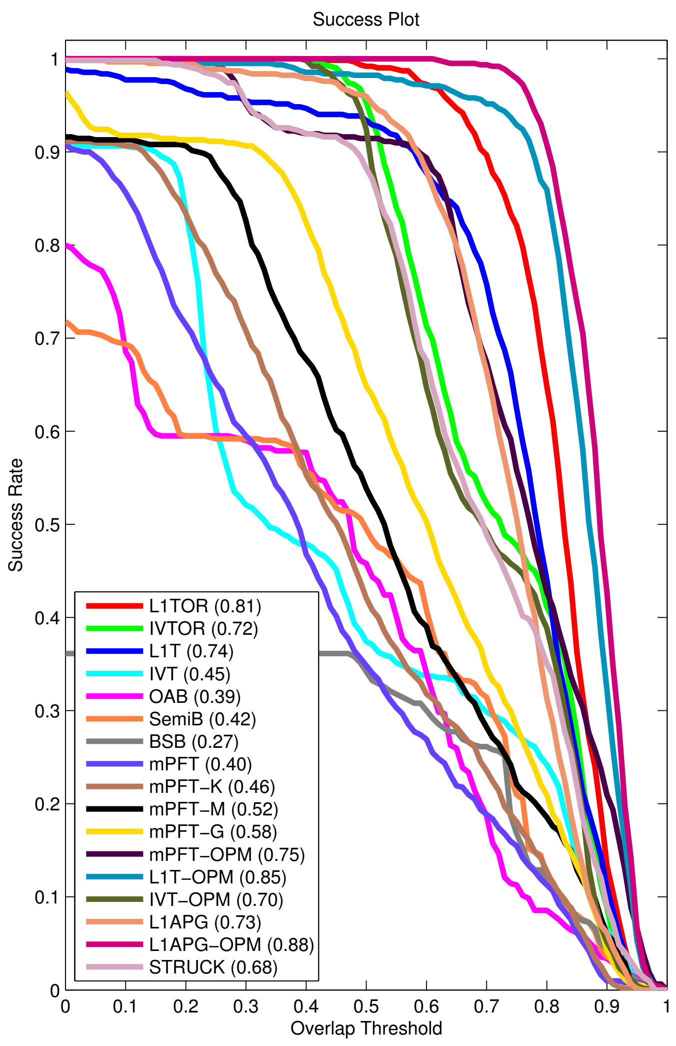


Figure 5.2: The success plot of trackers in Experiment 1. The AUC value of different trackers are presented below the plot (trackers ordered by ascending AUC value).

5.3.2 Obtaining Distributions

In the training phase, the image sequence and its annotated target position is given to the training algorithm. This algorithm then subtracts the background from the frames, and revised by a human annotator to achieve satisfactory precision. Then the target bounding box is decomposed into the regular grid $C \times C$, and the foreground ratio is calculated for all the cells and all the frames. The foreground ration of a specific cell across all training frames is then agglomerated into an array upon which, a parametric distribution is fit. The obtained distributions for all cells, are later used to calculate the occlusion probability distribution $\mathcal{B}_{C \times C}$. This distribution is then used to obtain the occlusion probability mask of a particle given the foreground rations of its cells, $\rho_{t,c}^{(j)} \propto \mathcal{B}_c\left(\zeta_{t,c}^{(j)}\right)$. This is in contrast to the approach in [61], where a occlusion mask is calculated on the estimated target position and acts on the next observation. Note that the likelihood obtained from this distribution would be normalized against the likelihood of other particles before being used in eq(5.1).

Training on the 65 training videos, we observed that the distribution foreground ratios across all frames can be well-approximated with a Beta distribution (i.e., $\mathcal{B}_c$ is a Beta distribution fit on the data obtained from 65 videos). This distribution is governed by only two parameters, and acts in the domain of $[0, 1]$. It takes various shapes to explain the variations of the distribution in cells (as depicted in Figure 5.1). While in the central cells, the distribution is almost bell-shaped, the distribution of marginal cells are skewed.

As discussed in Section 5.1, the occlusion mask proposed in [61] has four major issues: *(i)* sensitivity to training data, *(ii)* uniform treatment of different parts of the observation *(iii)* sensitive decision boundary in the occlusion classifier *(iv)* binary occlusion status of each cells which in turn results in *(a)* template grid

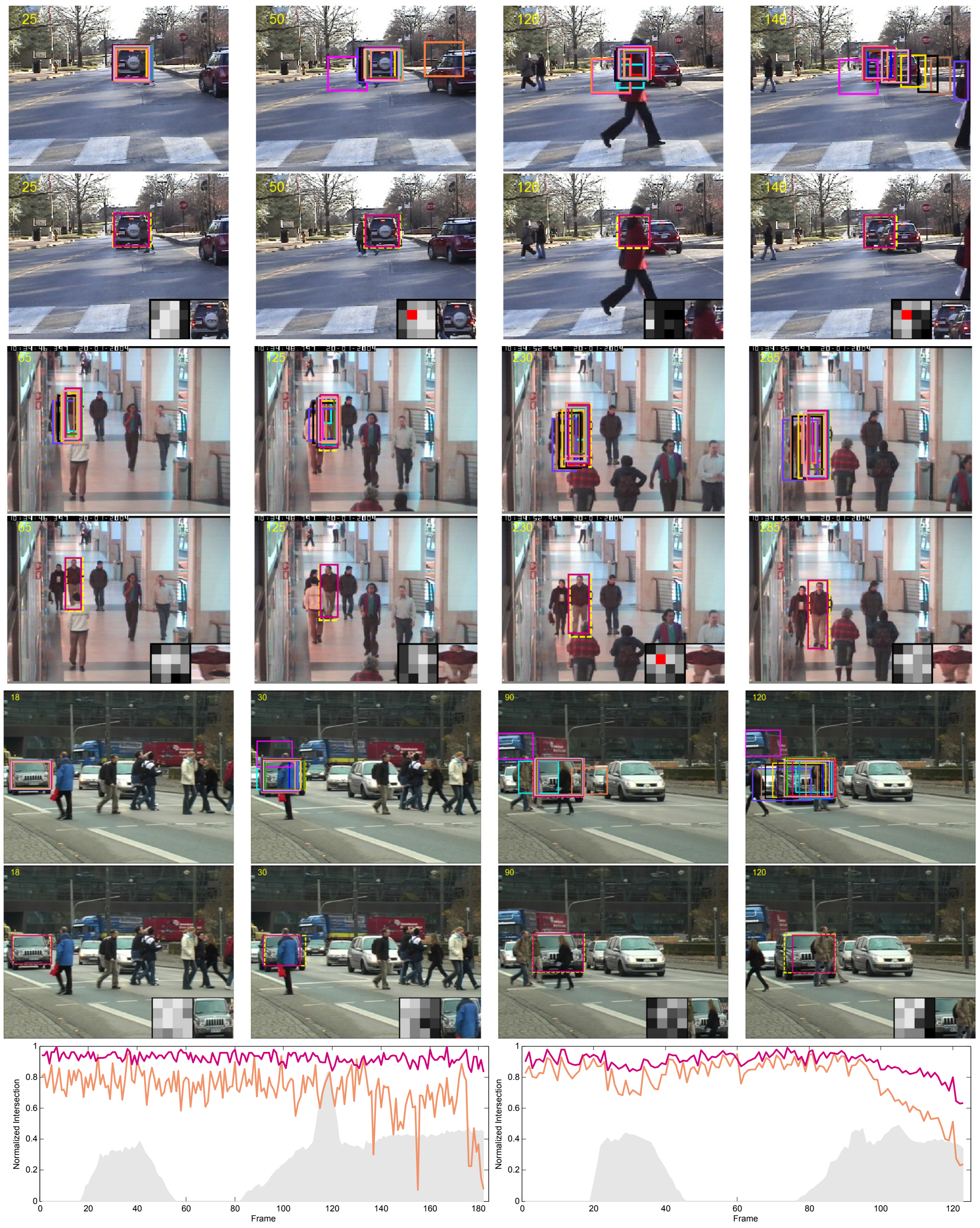


Figure 5.3: The tracking results of our algorithm on the `campus` **(top)**, `CAVIAR` **(middle)** and `TUD-Crossing` **(bottom)** sequences. Ground truth is illustrated by yellow dotted line, while three best trackers (*L1APG-OPM*, *L1T-OPM*, and *L1T-OR* and two worst trackers (*OAB* and *BSB*) for these scenarios are depicted on the frames. Isolated and noisy occlusion regions have been successfully detected as it is illustrated by the occlusion masks. The localization error of *L1APG* and *L1APG-OPM* is illustrated on the lower panel, while the shaded area indicates the extent of the occlusion.

cells inconsistency and *(b)* discarding critical information in partially occluded cells. We tackle these issues by using a robust mechanism for occlusion which is not target- or video-specific. A mechanism which generalizes the classifier to a probability distribution to avoid the problems caused by binary status of each cell, adopts a dedicated distribution for each part of the observation, and is not sensitive to variations in training data as an advantage of using a distribution. We benefit from the inherent property of the particle filter in the training process. Particle filter create a myriad umber of particles just around the target when

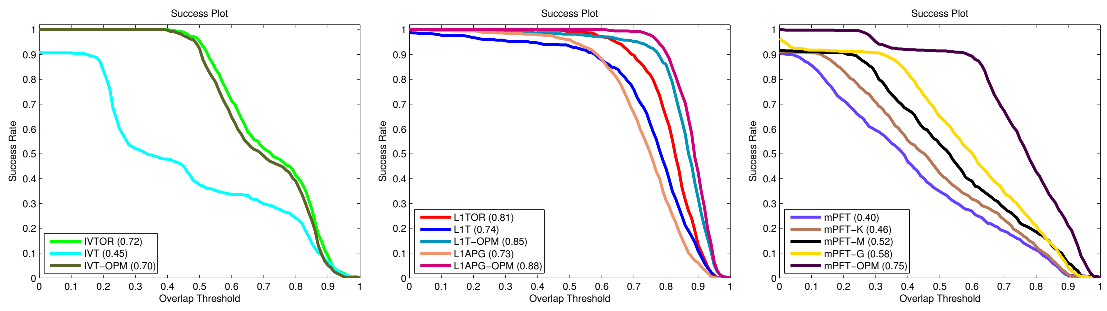


Figure 5.4: Success plot of the derivations of base trackers **(left)** *IVT*, **(middle)** *L1T* and **(right)** *mPFT*. They indicates the significance of the enhancement which OPM brings to the base algorithms, whereas it is effective on more advanced version of the base algorithms (e.g., *L1APG* tracker is an advanced version of *L1T tracker*).

most of the include some parts of the background or the occluder. While the classifier of [61] learns different patterns of targets and non-targets, our OPM learns how to deals with the cases where the target doesn't dominate a cell, but still it is present and may contain good features to track. This also renders the proposed method less sensitive to the training data given that the videos contains real-world scenarios and contain background clutter, occlusions and other tracking challenges.

5.4 Evaluation

To evaluate the effectiveness of proposed method, we first combine it with different base trackers and examine it with several challenging occlusion sequences. This experiment illustrates the advantages of proposed occlusion detection mechanism. The best $\langle base, OPM\rangle$ combination is then compared with six trackers from state-of-the-art in visual tracking on eight challenging public video sequences. The tracking results of presented methods were obtained using the codes provided by the authors with default parameters with the default parameters and using the same initial position in the first frame. Quantitative and qualitative analysis indicate that the proposed technique OPM in combination with *L1APG* [128] tracker surpasses other trackers in terms of tracking accuracy.

5.4.1 Data

To train our systems we use 65 videos, the colored sequences from TB-100 [24] (except `Singer1`, `Walking`, `Walking2` and `Deer`). The target location annotation of these 65 videos are publicly available in the form of axis-aligned bounding box. The selected videos include sports, humans, animals, cars, toys, faces, movies, live performances and theater-grade animations. The dataset is annotated with nine attributes and videos are selected to ensure that different video attributes (e.g. background clutter) are adequately present in the dataset.

Note that in contrast to [61], no manual occlusion annotation is required to train the occlusion mask, just a few human interventions are required to rectify the mistakes of background subtraction algorithm. The training procedure subtracts the background of every target bounding box, divides it into a 4×4 regular grid, and calculates the foreground ratio in all of them. Finally the parameterized distribution is tuned to fit the accumulated data for each cell. Three publicly available videos (from TB-100 [24], TUD-Crossing [173], and CAVIAR [115]) are used as test videos in the experiment 1. Furthermore in Experiment 2, nine public sequences `Jumping`, `Car4`, `Singer1`, `Walking`, `Walking2`, `Sylvester`, `Deer`, and `FaceOcc2` from TB-100 [24] are used as test sequences. All sequences involve various types of partial and/or full occlusions, and most of them involve multiple occlusions.

5.4.2 Implementation

In our implementation, we have limited our base trackers to use axis-aligned bounding boxes when they are using OPM (occlusion probability mask) for numerical stability reasons. Additionally, we decrease their tendency to change the bounding box size to satisfy the requirements of the employed background subtraction method.

The proposed OPM integrates successfully with the base trackers as it is demonstrated in the experiments. The extra time complexity imposed by the method is also reasonable, hence it can meet the requirements of online tracking. For instance, adding the OPM reduces the calculating speed from 24.1 fps in *L1APG* to 21.3 fps in *L1APG-OPM* on a Pentium IV 3.50 GHz.

5.4.3 Different Base Trackers

This experiment strives to demonstrate the effect of proposed occlusion mask in combination with different base trackers (*mPFT* [96], *IVT* [10], *L1T* [134] and *L1APG* [128]). Furthermore, our results are compared with 13 other tracking algorithms such as *mPFT* [96], *mPFT-K*, *mPFT-M*, *mPFT-G* [38], *IVT* [10], *L1T* [134], *L1APG* [128], *OAB* [64], *SemiB* [65], *BSB* [130], *STRUCK* [27], *L1T-OR* [61] and *IVT-OR* [61].

In this experiment we used *mPFT* with 8×8×8 histogram of colors and 256-bin LBP texture descriptor [145]. The observation model of this tracker can be enhanced using different observation masks. We implemented radial kernel in *mPFT-K* and data-driven pixel probability map in *mPFT-M* inspired by [44] and [34]. Additionally we extended the observation model introduced in [38] to use multiple-features in *mPFT-G*. The observation model in [38] utilizes raw foreground ratio to construct the weight-map for the observation. The combination of *mPFT* and our proposed occlusion probability map (elaborated in Section 5.2) is denoted by *mPFT-OPM* in this experiment.

In addition, *L1T-OPM* and *IVT-OPM* are obtained by combining *L1T* and *IVT* with the proposed occlusion mask using a procedure similar to [61]. However, *L1T-OR* and *IVT-OR* which are proposed in [61] discard partially occluded cells due to the binary state of each cell (either occluded or not occluded), but our proposed occlusion mask uses a more moderate approach by assigning lower weights to the cells with lower calculated probability rather than discarding them completely. This approach enables a smooth template change and effective use of unoccluded parts of the cell. Installing the proposed OPM on *L1APG* tracker follows the same procedure as *L1T*. *L1APG* uses an internal occlusion detection method that monitors the energy of the trivial templates and in the case of occlusion, allow them to have more energy [1]. It turns out that this scheme does not interfere with our proposed algorithm and these two mechanisms work together perfectly.

In Figure 5.3 the state of the ground truth, three best and two worst trackers for three test sequences are demonstrated — to prevent the clutter in the graph by drawing all 17 bounding boxes. The occlusion probability map and the estimated target in each sequence is overlaid in the right corner of the corresponding frame. The occlusion probability map is illustrated with a color-coding scheme in which brighter cells have higher likelihood. Degenerate cells are also marked in red.

Being tested on `TUD-Crossing` sequence (Figure 5.3 top), multiple occlusions by pedestrians is introduced. *L1T* and its derivatives, *mPFT-G*, *STRUCK* and *SemiB* tracked the target successfully and maintained target templates accurately. The moving car in the `campus` sequence (Figure 5.3 bottom) is severely occluded by pedestrian and another car. The severe occlusion in frame 120 was the tipping point for many of trackers when they start to drift away from the target. By observing the occlusion mask, it is evident that very low probability is assigned to the occluded portions of the template, while the boundaries of the target are not completely discarded by the proposed mask. Furthermore, the persistent occlusion in the following frames troubled *SemiB* that previously survived the occlusion around frame 120. The `CAVIAR` sequence (Figure 5.3 mid) involves several partial occlusions and troubled *mPFT*, *mPFT-K* and *mPFT-M*.

Being tested on `TUD-Crossing` sequence (Figure 5.3 top), multiple occlusions by pedestrians is introduced. *L1T* and its derivatives, *mPFT-G*, *STRUCK* and *SemiB* tracked the target successfully and maintained target templates accurately. The moving car in the `campus` sequence (Figure 5.3 bottom) is severely occluded by pedestrian and another car. The severe occlusion in frame 120 was the tipping point for many of trackers when they start to drift away from the target. By observing the occlusion mask, it is evident that very low probability is assigned to the occluded portions of the template, while the boundaries of the target are not completely discarded by the proposed mask. Furthermore, the persistent occlusion in the following frames troubled *SemiB* that previously survived the occlusion around frame 120. The `CAVIAR` sequence (Figure 5.3 mid) involves several partial occlusions and troubled *mPFT*, *mPFT-K* and *mPFT-M*.

The performance of the three best and two worst algorithms on these sequences are compared quantitatively in Figure 5.3 for each video. The overlap ratio is the normalized bounding box intersection ratio between ground truth and tracking result for a frame in a video sequence. To compare the overall performance of trackers, a “success plot” is presented in Figure 5.2. A tracker is successful to track the target in each frame, when it has an overlap ratio greater than a threshold τ_o. For a specific threshold, the performance of a tracker is calculated as the sum off all successful frames over the number of frames. Higher

[1]Trivial templates explains occlusion and noise in the reconstruction of the original image.

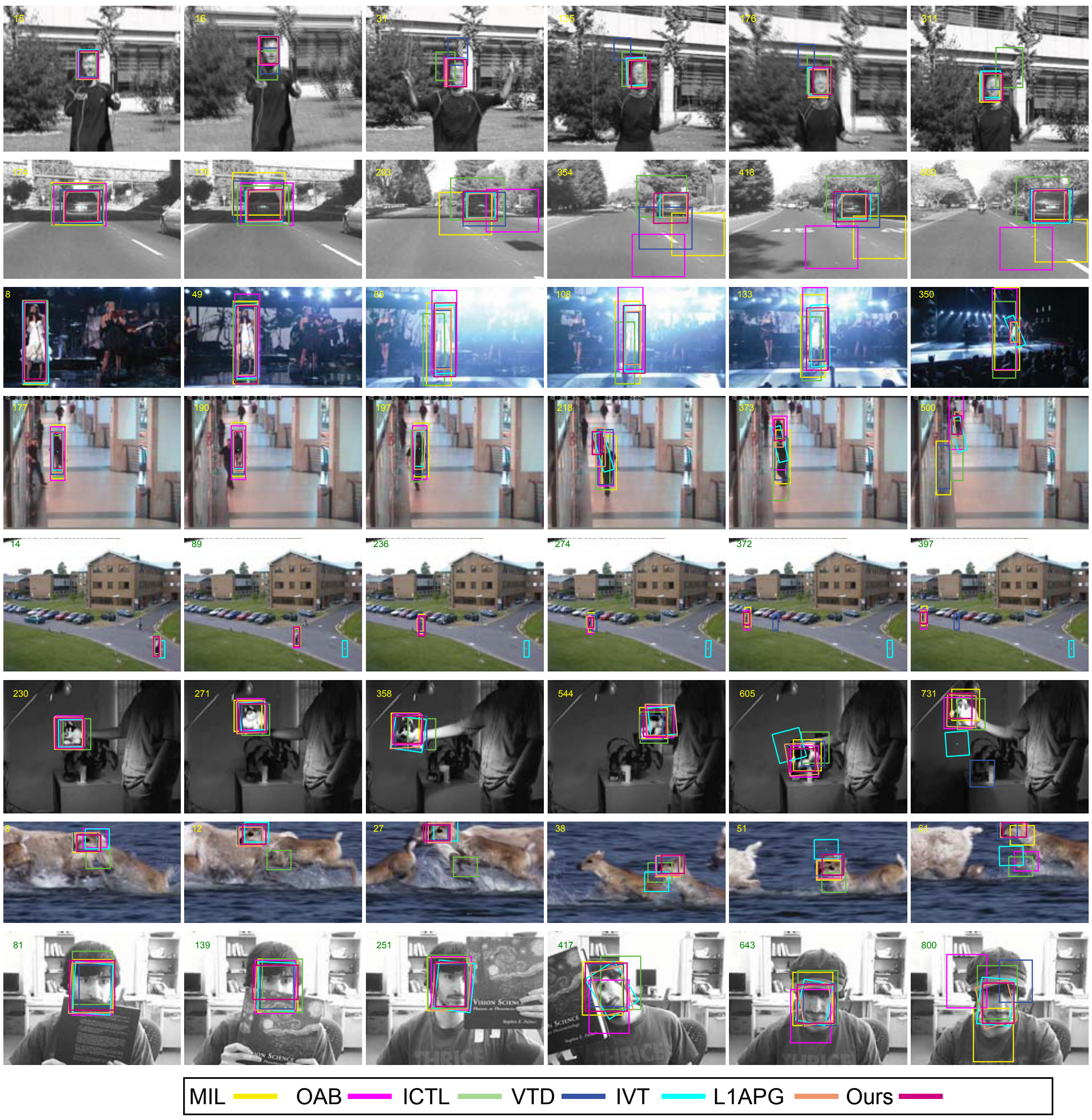


Figure 5.5: Tracking results of different algorithms for sequences `Jump`, `Car4`, `Singer1`, `Walking1`, `Walking2`, `Sylvester`, `Deer`, and `FaceOcc2`.

thresholds naturally result in lower number for performance, and no particular value for threshold is expressive enough to compare trackers. Thus, success plot presents the performance of a tracker given different values for the threshold. The area under curve (AUC) of the trackers serves as the fair measure to compare the results (Figure 5.2).

The results demonstrated a great improvement from *mPFT* to *mPFT-OPM* (AUC jumps from 0.40 to 0.74) . Additionally, the best tracking result is obtained by *L1APG-OPM* (AUC = 0.87). However, the proposed occlusion mask is not perfect as observed in this experiment. Since the occlusion indicator is based on the foreground ratio, our proposed method is unable to distinguish the occlusions that are perfectly aligned with the central parts of the target. The worst result goes to *BSB* (AUC = 0.26) that failed to track the target to the end in all three sequences. Furthermore, *STRUCK* (which is one of the most successful trackers to handle occlusions in a large benchmark according to [24]) was surpassed by the methods that utilize explicit occlusion mask. The results reveals that the size of the bounding boxes obtained by *L1APG-OPM* are generally smaller than those of *L1APG*. In the former, the occlusion mask punishes the marginal cells with very small foreground ratios (in excessively large bounding boxes), and automatically adjust the scale of the resulting bounding box.

	MIL	OAB	ICTL	VTD	IVT	L1APG	Ours
Jumping	0.030	0.030	0.198	0.221	**0.020**	0.025	0.027
Car4	0.749	0.786	0.326	0.313	0.049	0.048	**0.041**
Singer1	0.299	0.466	0.503	0.056	0.155	0.069	**0.039**
Walking1	0.361	0.179	0.323	0.339	0.148	0.032	**0.026**
Walking2	0.007	0.010	0.008	0.049	0.572	**0.003**	0.005
Sylvester	0.069	0.058	0.096	0.203	0.197	0.032	**0.024**
Deer	0.022	0.060	0.306	0.027	0.110	0.017	**0.009**
FaceOcc2	0.120	0.144	0.137	0.209	0.053	0.062	**0.047**
Average	0.207	0.216	0.237	0.207	0.163	0.36	**0.027**

Table 5.1: The average tracking error. The error is measured using the Euclidean distance of two center points, which has been normalized by the size of target from ground truth. In this table, the lowest tracking error is shown with bold font. Furthermore, the highlighted entries in red indicates than the corresponding tracker was unable to accomplish tracking on that particular sequence. The last row of the table presents the average of the accuracy errors of trackers for all sequences.

5.4.4 Comparison with other Trackers

The findings of the previous section revealed that *L1APG-OPM* tracking pair is significantly effective in managing occlusions and outperforms similar algorithms. The second experiment is meant to investigate the localization accuracy of this tracker pair in a set of eight videos which involves several tracking challenges (e.g., illumination change, background clutter, abrupt motions) on top of various occlusions. To this end, six state-of-the-art trackers (*Incremental Visual Tracking* (*IVT*) [10], *Mutiple Instance Learning* (*MIL*) [12], *Visual Tracking Decomposition* (*VTD*) [53], *Incremental Covariance Tensor Learning* (*ICTL*) [174], *Online AdaBoosting* (*OAB*) [64] and *Accelerated Proximal Gradient L1 Minimization tracker*(*L1APG*) [128]) are evaluated and their localization accuracy is listed in Table 5.1. To evaluate the localization accuracy of these trackers, the localization error is measured as the Euclidean distance between the center of ground truth and tracker bounding boxes and normalized by the size of the ground truth bounding box. Additionally, highlighted entries (with red font color) in Table 5.1, indicates that the corresponding tracker failed to track the target throughout that sequence.

Abrupt motion and the resulting blur in sequence `Jumping` impedes *ICTL* and *VTD*. Fast motion and clutter in Deer sequence drifts away many trackers from the target. In `Car4` sequence, cast shadows trouble most of the trackers. Likewise in `Singer1` sequence, severe illumination changes impede many trackers. *L1APG-based trackers* were the only ones to handle the challenging confusion case in `Walking1`, while in `Walking2` sequence the *IVT* and *VTD* were unable to keep the target due to scale changes and low resolution of target. These two tracker also fail to keep track of the target in Sylvester sequence that contains rapid pose and illumination changes. Various occlusions in `FaceOcc2` overwhelm most of the trackers while *L1APG* based trackers and *IVT* successfully performed the tracking on them. The tracking results of this experiment are illustrated in supplementary materiel.

Inherited from its base tracker, *L1APG-OPM* achieves high localization accuracy in various situations, but the relatively large improvement in `Sylvester` and `Singer1` sequences indicates the effectiveness of the proposed occlusion handling mechanism. As stated in the last row of Table 5.1, *L1APG-OPM* achieves the best overall performance among the evaluated trackers over these challenging set of videos.

we concluded that the best combination of OPM with base trackers is achieved using *L1APG*. This can be attributed to the optimization framework in *L1APG*, which efficiently use the information embedded in the masked observation. Besides, the occlusion detection mechanism in *L1APG* acts as a compliment for the proposed OPM, and make the tracker more robust against occlusions. Figure 5.5 demonstrates the qualitative results for the performance of *L1APG-OPM* (porposed) against current popular trackers: *IVT* [10], *MIL* [12], *VTD* [53], *ICTL* [174], *OAB* [64] and *L1APG* [128].

5.5 Conclusion

In this study, we proposed an explicit occlusion detection mechanism for visual tracking. Our algorithm utilizes the foreground ratio as the random variable in different regions of an observation, and empirically estimates the probability distribution of this variable. Using these distributions, an occlusion probability mask is constructed to evaluate the reliability of various regions of an observation. This mask was proved to be successful in managing occlusions and safely updating the target template throughout the tracking sequence. The next step would be to find better occlusion indicators to serve as the main occlusion detection indicator.

6 Improving the Tracking Performance

"Simplicity is the ultimate sophistication."
— Leonardo da Vinci

6.1 Gridding

Histogram-based local descriptors are pervasive tools in many computer vision tasks, such as image retrieval, object tracking, and object recognition. Such descriptors compress the original distribution, which save storage and computation time and grant a certain perceptual robustness to the matching [175]. Being both compact and invariant to specific changes in the image, color histograms gained popularity in object tracking, e.g., the famous *mean-shift tracker* [176] and color-based particle filter tracker is built upon this feature.

Histogram of color (hereafter HOC), however, suffers from lack of spatial information and cannot differentiate patterns of colors and just considers the portion of them in an image. For that, several other cues are used along with them in systems to improve tracking accuracy such as texture, motion, shape etc. Furthermore, the quantization effect and perceptual difference of colors inaugurate more challenges in using this feature. Nevertheless, HOC is usually in charge of making most out of the color information. To alleviate the shortcomings of this feature, researchers use various color spaces with special characteristics, employ different similarity measures to compare color histograms, and try to embed necessary spatial information into the feature whenever possible.

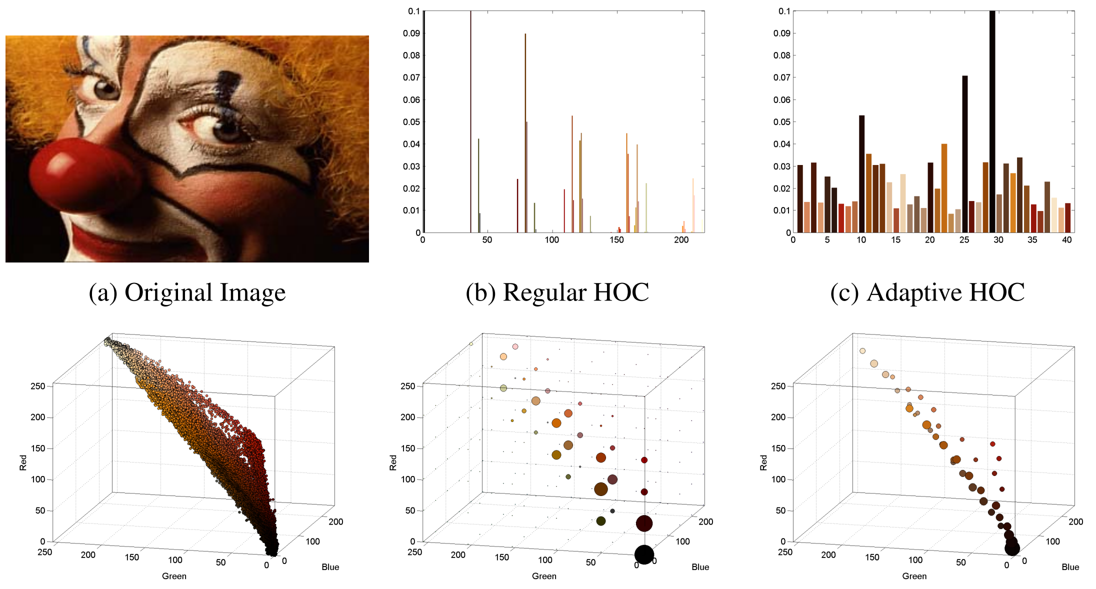
This article focuses on RGB color histograms and scrutinizes the use of gridding as a simple mathematical trick to boost the performance of the feature while preserving all of the advantages of a HOC. Presenting a gist of numerous ways in which HOC is used in RGB space, the manuscript then compare these ideas and enhance them by proposed scheme, to show the effectiveness of the gridding idea.

6.2 Literature Review

6.2.1 Histogramming

Histograms, by default are a partition of the space into equal bins. Applied on RGB data, each channel is partitioned into same-size bins. The result is a regular histogram of color or R-HOC in which n governs the coarseness of the bins. Being simple and interpretable, this type of HOC is the very popular. However, regular binning for high dimensional data often yields poor performance: fine binning leads to fluctuations due to statistically insignificant sample sizes for most bins, while coarse binning diminishes resolving power [175].

A solution to this issue is to employ adaptive binning whereby the bins are adapted to the color distribution. Clustering methods such as k-means provides such binning, which are then used to make an adaptive



(a) Original Image (b) Regular HOC (c) Adaptive HOC

(d) Color Point Cloud (e) Regular Binning (6×6×6 bins) (f) Adaptive Binning (40 bins)

Figure 6.1: Regular and adaptive binning to construct HOCs. While the former, assign color points to predefined bins, the latter cluster colors to find the best bin centers for histogramming. The Regular HOC, is sparse and is sensitive to quantization level of color channels. The Adaptive HOC employs a more compact representation of the color distribution in the image, and is more intuitive. In this figure the color histograms are normalized, and the scale of color distributions are altered for better visualization.

histogram of colors (or A-HOC). The number of bins in this method is parameterized by color clusters. Figure 6.1 illustrates the difference between regular and adaptive binning to construct color histograms.

6.2.2 Similarity Functions

In order to match histograms, a similarity function must be coupled with the color feature. Being an empirical estimate of a probability distribution of feature, histograms can be treated similar to distributions, thus a similarity function, enjoys a wealth of techniques to compare two distributions. Following [175], we distinguish four categories for dissimilarity measures.

First category involves *heuristic histogram distances* that are mostly proposed in the image retrieval context, but is also popular in other computer vision tasks. The Minkowski distance family (L_p) is arguably the most used heuristic for dissimilarity. L_1 computes the sum of absolute dissimilarity, L_2 penalizes the larger errors more severely, and L_∞ as the limiting case measures the maximal difference. Having the normalized histograms in hand, Cosine distance (CO) reduces to inner product of two histograms and indicates the distance between them as the cosine of angle in between. The Pearson correlation coefficient (CR) indicates the linear dependence between random variables.

The second category is derived from *non-parametric test statistics*, which examine the hypothesis that two empirical distributions are generated from the same true distribution. The Kolmogorov-Smirnov distance (KS) is defined as maximal discrepancy between two cumulative distributions, Match distance (MA) equals the sum of absolute distance between their cumulative distances. Cramer-von Mises statistics (CM) penalize the discrepancy of two cumulative histograms quadratically as it sums them. The measures based on cumulative histograms are all sensitive to bin ordering. The χ^2 statistics (CS) is adopted from Pearson's chi-sqaure test statistics and was introduced with an asymmetric formulation in literature. Later a slight modification addressed the issue and improved the performance of the metric. Zweng et al. [177] reported that a bounded version of this function has best performance in a tracking task in several color spaces. However, this function is sensitive to quantization effect and shape deformation [178]. Another distance in this category is Bhattacharyya distance (BH), which uses Bhattacharyya coefficient to measure the dissimilarity of two distributions.

Third category of distances is inspired from *information-theoretic divergences*. The Kullback-Leibler divergence (KL) measures "how inefficient on average it would be to code one histogram using the other as the true distribution for coding". This measure is not bounded and is sensitive to empty bins in the histogram. A symmetric and numerically stable version of KL-divergence is presented in Jeffrey Divergence (JD).

The final category is composed of distance measures empowered with *cross-bin information*. In applying the bin-to-bin functions presented in the last three categories, it is often assumed that the domain of the histograms is aligned. However, in practice, such an assumption can be violated due to various factors, such as shape deformation, non-linear lighting change, and heavy noise. The ground distance, is defined as the amount of difference between elements of the feature vector. With the aggregation of all ground distances in a positive-definite symmetric similarity matrix $\mathbf{M}$, the weighted L_2 distance will be transformed into Quadratic distance for color histograms (QD). An extension to Quadratic distance, introduced in [178], reduces the effect of large bins. In this hybrid, Quadratic-Chi distance (QC), the normalization power of chi-square is utilized along with cross-bin relationship presented by quadratic distance. Another member of this category is Diffusion distance (DF), which models the problem as a temperature field and considers the diffusion process on it [179]. Earth Movers Distance [180] is another family member that models the histogram distance with different bin sizes and bin centroids as a transportation problem (EM). Many approaches try to improve the speed of this method. For example $EMD - L_1$ exploits the structure of L_1 distance and $\widehat{EMD}$ [181] used a graph theoretic approach with thresholded ground distance to linearize the computational complexity.

6.2.3 Model Update

The tracking task, involves lots of dynamics in the subject ranging from rigid and non-rigid transformations to abrupt changes in pose, motion, illumination, camera parameters, temporal clutter, noise and occlusion. Robust trackers compensate for these changes by updating their model across the time, to keep the model close to the designated subject. Several model update (MU) approaches are popular in the literature.

The simplest approach is to average all of the observations from the beginning of the sequence. Although updates the model for new changes, this method lacks enough flexibility to accommodate drastic changes in the model. In the second method a leaky memory scheme is employed, i.e., all of the observations contribute to the template model based on their recentness. The result of this approach is over-smoothed and may lose the clarity after a while. The third approach uses an extra buffer to update the model. Essentially an extension of the leaky memory model, this method updates the model with small forgetting rate for every observation. Then after a certain number of intervals, the model is updated with a larger forgetting factor that helps the removal of the remainders of initial observations if not useful anymore. Finally, the forth approach is to use only the recent observations. For this a queue of a fixed size is utilized and each new observation is enqueued in it. The template in each frame is the average of all models in the queue.

6.3 Gridding for HOC: Divide and Conquer

This section elaborates the proposed feature.

6.3.1 Expansions of Color Histogram

It has been stated in the literature that tracking using region based descriptors take two extremes: blob-based trackers and pixel-wise template trackers. Generally a blob-based tracker accumulates less information about the target, since it discards all its spatial information. A middle level to this spectrum has been established by the use of kernel histogram methods. While the initial method based on mean shift formulation [176] share more similarities to blob trackers, some modifications (such as those in [182]) drift them more toward the template tracker end of spectrum.

However, even with adaptive binning based on the overall distribution of all images in a tracking sequence, often for specific patches of image only a small fraction of the bins in a histogram contain significant information. Fine quantization for the histogram as a solution is highly inefficient. On the other hand for an image characterized by many color details, a coarse binning for histogram is not adequate. Traditional histograms as fixed-size structures can hardly balance efficiency and expressiveness. Several methods have been proposed to enhance the color representation, including variable-size histograms called signatures [175], embedding color spatial relations into color corellograms [183] and expansion of histogram with higher order spatial moment called spatiograms [184]. These approaches are changing the nature of color histograms thus reducing its robustness to rotation, deformations and illumination changes while improving the performance in the normal condition.

6.3.2 Proposed Method

Here we introduce the idea of gridding, which while preserving the very nature of color histograms, addresses the above-mentioned issues. The intuition behind this method is that with breaking a big block of data into smaller chunks, processing them and aggregating them again, the data become more tractable and is augmented with the underlying process. The agglomeration scheme in turn grants desired properties to the result. Specifically in this case, we break the input image, into spatially-regular grid segments, calculate the histogram of color to each of sub-images and combine obtained histograms to make one final histogram. We propose two agglomeration techniques: *(i)* averaging in which the final HOC is smoothed to avoid a few bins overload the total histogram if placed only in a part of image; *(ii)* linear combination in which each grid cell is analyzed and given an importance coefficient and the resulting histogram is the linear combination of grid histograms weighted by these coefficients. We call the former method grid-driven average HOC (G_a) which requires no additional information. In case of the latter, grid-driven weighted HOC (G_w), we define the importance of the grid cell proportional as the ratio of foreground pixels to all pixels in the grid cell. This requires background subtraction, which is not feasible in many tasks, and the importance criteria can be arbitrary function which assigns a weight to the grid cells (Figure 6.2). The procedure of constructing G_a and G_w is elaborated in Algorithm 4. The binning strategy is arbitrary in line 4 of this algorithm.

It is important to contrast our method with spatial pyramid matching [185]. This method performs pyramid matching [186] in image space and use clustering in feature space. It concatenates the results of different levels of pyramid, which yield a large sparse feature vector. This holistic method is suitable for scene classification. It is not suitable for tracking task in which background clutter challenges the tracker and object transformations should be handled. Also exponentially shrinking cells of this method, increase the chance of outlier problem in the histogram. In contrast, our proposed feature manages object transformations by averaging and accounts for background.

Algorithm 4: Constructing Grid-driven HOC

```
input  : Gridding granularity $n$, Image Patch $I$
output: Grid-driven HOCs $G_a$ and $G_w$
1 $B \leftarrow$ Calculate Image Background Mask
2 $I_{i,j} \leftarrow$ Divide $I$ into $n \times n$ cells
3 $B_{i,j} \leftarrow$ Divide $B$ into $n \times n$ cells
4 $C \leftarrow$ Calculate Histogram Centers
5 $\alpha_{i,j} \leftarrow$ Calculate ratio of foreground pixels to all in $B_{i,j}$
6 $h_{i,j} \leftarrow$ Calculate HOC for $I_{i,j}$ using $C$
7 $G_a \leftarrow \frac{1}{n^2} \sum_{(i,j)=(1,1)}^{(n,n)} h_{i,j}$
8 $G_w \leftarrow \frac{1}{n^2} \sum_{(i,j)=(1,1)}^{(n,n)} \alpha_{i,j} h_{i,j}$
```

6.4 Evaluation

In this paper we investigate the performance of different similarity measures under various histogramming paradigms, and show the efficiency of the proposed gridding method to enhance almost all of them. Another experiment is dedicated to examine the compatibility of histogramming method with similarity measure and model update. All of the sequences are in RGB color space and for all experiment related to a histogramming schemes, the bins are preserved the same. Following the style of [177], we used 50 consecutive frames from `S2.L1.12-34.View001` sequence of PETS 2009 dataset to simulate a tracking task. The three pedestrians present in the sequence are represented with a bounding box, manually pre-segmented and a background mask is prepared to block the effect of other factors on the evaluations [177].

6.4.1 Experiment I

In the first experiment, different combinations of histogramming method and similarity measures are examined on the sequence. The similarity of each person in each frame to that of last frame, *intra-similarity*, is calculated throughout all of the sequence. Furthermore, the mutual similarity of each person to other subjects in each frame, the *inter-similarity*, is calculated for each frame as well. The inter- and intra-similarity

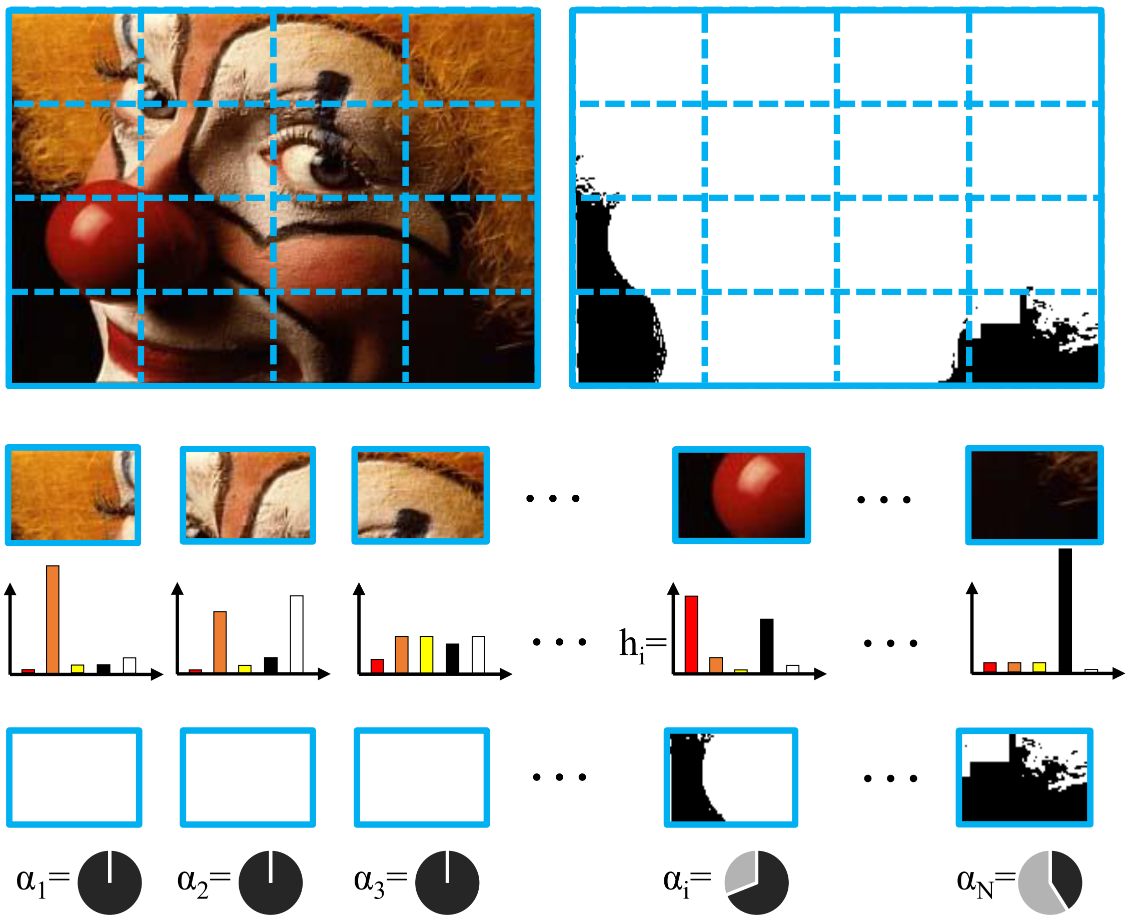


Figure 6.2: Gridding to enhance HOC. The Image and its background mask (if available), is divided into spatially-regular grid. Then, a HOC is constructed for each cell, and the weight of cell (i.e., ratio of foreground to all pixels) is calculated. Finally, these histograms are agglomerated to produce a HOC with the similar length of each of cell HOCs.

values are obtained from $a_{ij} = 1 - (d_{ij} \div d_{max})$ in which d_{ij} is the distance between bounding boxes i and j, and d_{max} is the maximum of all distances measured for this dissimilarity function (e.g. CS). Intuitively, a similarity function should maximize the intra-similarity, while minimizing inter-similarity. To facilitate comparison between different combinations of $\langle HOC, DIST \rangle$, we introduce a score $S = \sqrt{\alpha(1-\beta)}$ in which α is the average of all intra similarity values and β is the average of all inter-similarity values. Higher S values (values closer to one), indicates a better combination in distinguishing target and non-target bounding boxes.

Table 6.1 gathers a comprehensive list of all combinations of histogramming methods and similarity measure combinations [1]. In this table, two types of HOCs, Regular with 5×5×5 bins and Adaptive with 40 bins, two types of gridding with different grid sizes ($G_a n{\times}n$ and $G_w n{\times}n$), and 16 similarity measures are compared. For the cross-bin similarities, a matrix $\mathbf{M}$ is constructed for both regular and adaptive histogramming methods based on CIEDE2000 color distance and is shared between all measures. The project code can be found at `https://github.com/meshgi/Histogram_of_Color_Advancements` and EM distance is based on $\widehat{EMD}$ [181]. From Table 6.1 it is apparent that KL-divergence provides the best score for tracking which means the object has higher intra-similarity throughout the sequence, while the rest of the objects have large distance to that. It is also evident that gridding-enhanced histograms have better performance than the normal condition. In this particular example the 3×3 grid-driven weighted color histogram outperforms other schemes for most of the possible similarity measures, especially with KL-divergence that achieves the best performance of $S = 90.83\%$.

Many other patterns are evident in this table, interesting for further investigation. For instance, it is

[1]Due to space limitation only 10 of the measures are presented, the rest are available in the author webpage: `http://ishiilab.jp/member/meshgi-k/r-tracking.html`

Table 6.1: Combinations of $\langle HOC, DIST\rangle$ for tracking robustness, row best is in **bold**, column best is underlined, and best value is shaded (A: adaptive, R: regular))

HOC	Similarity Measures (%)									
	L2	CR	CS	BH	KL	DF	KS	CM	QC	EM
R	69.7	62.6	70.4	72.1	**86.2**	71.0	69.6	48.8	64.2	67.8
A	72.6	72.0	74.5	72.9	**88.2**	72.4	66.8	54.5	64.0	66.4
R G_a(2×2)	70.3	63.5	71.0	72.4	**86.4**	71.4	70.1	49.8	65.9	68.0
A G_a(2×2)	73.0	72.9	74.5	73.1	**88.9**	73.3	71.1	54.5	65.6	66.6
R G_a(3×3)	70.0	63.3	71.0	71.9	**86.2**	70.9	69.9	49.6	65.7	67.5
A G_a(3×3)	72.7	72.0	75.4	72.5	**90.4**	73.4	69.9	51.9	65.3	66.2
R G_a(5×5)	69.5	62.8	70.9	71.7	**86.0**	70.7	69.3	48.4	64.9	66.9
A G_a(5×5)	72.9	72.1	73.4	71.8	**88.7**	72.4	68.4	54.2	64.6	66.4
R G_w(2×2)	69.9	62.7	70.5	70.9	**86.5**	69.8	69.3	51.1	63.8	65.0
A G_w(2×2)	72.6	73.2	75.4	72.1	**88.8**	71.9	64.9	40.3	64.0	63.7
R G_w(3×3)	70.0	63.3	71.0	71.9	**86.2**	70.9	69.9	49.6	65.7	67.5
A G_w(3×3)	73.6	74.6	76.2	73.0	**90.8**	73.6	72.7	60.9	65.4	66.2
R G_w(5×5)	69.5	62.8	70.9	71.7	**86.0**	70.7	69.3	48.4	64.9	66.9
A G_w(5×5)	72.8	72.1	74.5	72.5	**88.9**	73.4	69.4	61.1	64.6	66.1

apparent that the performance of the grid-based average HOC along with cross-bin similarity measures is better than the other distances. On the other hand the statistic test distances get along with weighted version of gridding. This is intuitive as the weighted gridding emphasizes the statistics in each sub-frame proportional to their significance in the total result. On the other hand, cross-bin measures emphasize on the distance of the colors and the amount of correspondent bin-to-bin difference which is smoothed by the process of averaging. Interestingly, *QC* as a bridge between these two families bends in favor of averaging. Moreover, there is an optimized grid size for the sequence in which most of the similarity measures showed the best performance. This size is object-specific and in our case, for the pedestrians a 3×3 grid works well (left/middle/right + head/torso/legs) despite the out-of-plane rotation of the subjects in the sequence.

In addition to our proposed features, we examined the performance of pyramid matching [186] and spatial pyramid matching [185] in this experiment. The former achieved $S = 79.7\%$ while the latter, which is designed as a holistic feature for scene understanding, obtained only $S = 51.3\%$.

6.4.2 Experiment II

In a strive to find the best $\langle HOC, DIST, MU\rangle$ combination, we attempt another experiment in which all new bounding boxes are matched to a template. This experiment is particularly useful to evaluate the performance of this combination in top-down trackers. In this experiment the target in the first frame acts as an initialization for the template, which is updated in the successive frames using surveyed methods. All of the similarity values for each tuple of $\langle HOC, DIST, MU\rangle$ are averaged to make a final score. In this implementation, the forgetting factor for leaky memory is set to 0.1, the queue size for storing the latest templates are set to 5, and the forgetting factor for the second layer of buffer is set to 0.4. Table 6.2 contains the best result of model update schemes.

Interestingly, all of similarity measures, except *CO*, took their maximum value from last 5 frame MU scheme. This indicates the role of incorporating recent changes into the model. It also suggests that higher forgetting rates for the leaky memory schemes may be beneficial in this scenario. It is also noteworthy that all of the test cases prefer a model update to no update case, which clarifies the role of model update in dynamics of tracking. In Table 6.2 it is clear that *CM* has the best template matching performance, but as it is inferred from Table 6.1 it does not have adequate inter-object similarity rejection and thus is not a good candidate for the histogram-based template matching. KL-divergence, on the other hand, has on average 4% less template matching performance under model update case. Yet, it enjoys an exponential punishment method for other objects, which makes it as an excellent choice among all the others.

Table 6.2: Combinations of $\langle HOC, DIST, MU\rangle$ for best template matching performance, row best is in **bold**, column best is underlined, and best value is shaded (A: adaptive, R: regular))

HOC	Similarity Measures (%)									
	L2	CR	CS	BH	KL	DF	KS	CM	QC	EM
R	86.0	95.5	93.4	82.8	93.2	84.5	86.4	**96.8**	90.8	80.4
A	86.5	94.5	92.9	83.1	92.8	83.7	88.8	**96.4**	90.4	80.5
R G_a(2×2)	85.9	95.4	93.2	82.7	93.1	84.3	86.3	**96.7**	90.6	80.4
A G_a(2×2)	86.0	94.4	93.0	83.2	92.8	83.8	84.1	**95.2**	90.2	80.7
R G_a(3×3)	86.0	95.4	93.2	82.8	93.1	84.5	86.4	**96.8**	90.7	80.6
A G_a(3×3)	86.7	94.4	92.6	82.9	92.4	83.5	88.6	**96.8**	90.3	80.2
R G_a(5×5)	86.1	95.5	93.2	82.8	93.0	84.5	86.7	**96.8**	90.9	80.7
A G_a(5×5)	87.1	94.6	93.0	83.1	92.8	84.1	88.0	**96.9**	90.3	80.4
R G_w(2×2)	84.1	95.0	92.5	81.8	92.3	83.4	84.4	**96.2**	89.8	80.6
A G_w(2×2)	84.0	93.4	91.9	82.2	91.7	82.6	85.8	**96.2**	89.4	81.2
R G_w(3×3)	86.0	95.4	93.2	82.8	93.1	84.5	86.4	**96.8**	90.7	80.6
A G_w(3×3)	85.6	94.3	92.9	83.6	92.8	83.4	88.4	**96.8**	90.4	80.8
R G_w(5×5)	86.1	95.5	93.2	82.8	93.0	84.5	86.7	**96.8**	90.9	80.7
A G_w(5×5)	86.5	94.4	92.9	83.3	92.8	83.6	88.5	**96.1**	90.5	80.6

6.5 Conclusion

In this paper, A new tool to improve visual tracking accuracy using color cues is introduced. The idea involves decomposing a bounding box into a regular grid, in order to incorporate more spatial information and embed local salient color details into the histogram. Two gridding schemes are also proposed to accommodate different requirements, when the background data is/is not available. Then a comprehensive study over the existing (dis)similarity measures is conducted and many ideas from other computer vision realms such as image retrieval and texture analysis are brought into object tracking.

Using two simulation experiments, we first study the performance of (dis)similarity functions in finding relevant target using color histogram, while discarding other objects to successfully track the designated object. The results signify the usefulness of gridding to improve the tracking performance. It showed that almost any combination of histograms and similarity measures are boosted by gridding. Based on the results it is advisable to use grid-based weighted HOC when the background data is available, but the other version is also superior to normal HOC in most of the cases. To employ these features in other trackers, the degree of gridding, n can be calculated using cross-validation.

In the second experiment we emulate a tracking scenario and illustrate the compatibility of different template update schemes for various dissimilarity measures. It has been found that while all model updates surveyed in this paper improves the performance of template matching, none of them is completely suitable for proposed gridding schemes. Also the results revealed that, calculating template as the average of the last 5 observations, outperformed other methods.

In future, we plan to extend the scope of study to other color spaces, irregular grids, optimal griddings, and uncontrolled tracking scenarios.

7
Conclusions & Future Directions

"A movement is accomplished in six stages, and the seventh brings return."
— Syd Barret

This study provided a comprehensive overview of the occlusion problem in online visual tracking. Based on the new categorization, occlusions are defined based on their three intrinsic attributes: duration, spatial extent, and complexity. Many studies tackled partial temporal occlusions, and significant progress has been made so that temporal partial occlusions with the spatial extent of 30% are considered as solved [23]. On the other hand, few attempts have been made to handle persistent or complex full occlusions, and even temporal full simple occlusions are still challenges for many trackers. This study collected the main problems caused by occlusion and provided the best practices in order to facilitate designing more robust trackers. By categorizing the state-of-the-art solutions to the occlusion problem, this study discussed the merits and demerits of each solution and illuminated the landscape for future research.

The thorough analysis in the survey highlighted effective approaches for robust tracking and provided potential future research directions in this field. Better foreground segmentation schemes, considering the split and merge events, dealing with varying number of objects, and incorporating other visual clues (such as context and motion patterns) into formulation of the association problem are recommended approaches to advance foreground trackers. By providing more robust detectors using latest breakthroughs in object categorization and fine-level object detection, the tracking-by-detection approaches would increase the accuracy of trackers. Feature detector and tracker fusion are trending solutions in this area while simultaneous localization and detection proved to be a successful strategy initiated by [70]. Moreover, part- and patch-based solutions and robust appearance models (locally sparse, discriminative, and occlusion-invariant) are the essence of recent successful trackers. Advanced motion models such as parametric models, stochastic sampling and adaptive motion models by the guidance of, e.g., optical flows or context information, seem to be another successful strategy. Cheap access to depth information provides the opportunity to use this rich source of information to disambiguate occlusion situations, to keep track of targets, to design powerful features, and to partially compensate the 3D-2D projection data loss. Robust detection of occlusions improves the tracker performance significantly and there are lots of work to do in this track. Utilizing hybrid models and switching mechanisms in order to compensate the demerits of different trackers with one another sounds promising. Occlusion reasoning is yet to be formalized, and by doing so, many ideas can be applied to this field. Formulating it as a competitive phenomenon between objects, decompositional approach and using context are a few directions in which preliminary studies gained success.

Inspired by the capabilities of mode-switching trackers, we proposed a novel particle filter-based tracking algorithm with a particular interest in handling persistent and complex occlusions. The most important contributions of our method comprise managing persistent and complex occlusions by explicitly using an occlusion flag attached to each particle and treating occlusions in a probabilistic manner. The flag (i.e., the switch to change the sampling approach and motion model) signaled whether the corresponding bounding box was occluded and then triggered the stochastic mechanism to expand the search area and to stop the template updating. Moreover, each particle in the proposed framework was evaluated by means of multiple features, so that its similarity to the target template was evaluated in terms of likelihood. Due to the newly developed 2D projection confidence feature involved in the set of multiple features, our tracker further exhibited high adaptability to changes in object scale and trajectory. The experimental data revealed that our set of seven features outperformed any subset of this feature set, and selecting effective features required keen attention to the aspects of the video sequence that the other features are monitoring. Through intensive experiments using the Princeton RGBD Tracking Dataset, we found that our proposed tracker showed good tracking performance under variety of occlusions, outperforming the state-of-the-art RGB and RGBD trackers and the baseline *PFT*s. Further evaluation using the 95 blind test sequences registered in the same dataset also indicated that our tracker performed best among the 17 state-of-the-art RGB and RGBD trackers benchmarked on the dataset. These successful results are attributed to the good characteristics of our tracker: preventing model drift, quick recovery from occlusions, and facilitating the fusion of features.

However, the feature selection and tuning has been a daunting part of designing a tracker, and researchers stick to the few known best practices in the literature. However, by changing the features and their weights it is observed that the tracker take on various types of personalities and behaves similar to certain trackers. Following this lead, we introduced a novel framework to imitate the behavior of an arbitrary target tracker, called *MIMIC*. *MIMIC* is made possible based on the assumption that most of the trackers can be imitated by a linear fusion of several features embedded in a non-linear tracker. We employed particle filter tracker and a rich pool of features to verify this hypothesis. Extensive experiments verifies the hypothesis, although heavily non-linear trackers and those using information from non-target areas can not be approximated well. This method discovered the underlying features of a successful tracker, imitate trackers under challenging scenarios (such as occlusion or illumination changes), tracker identification, shadowing a human annotator, and learn from multiple trackers. To evaluate *MIMIC*, we conduct a series of experiments and the finding suggests that it is capable of shadowing many of the state-of-the-art trackers, while it shed light on the possible inner mechanism of black-box trackers. Additionally, the results showed that using dropout to prevent over-fitting is essential to obtain a proper *MIMIC*. The data also revealed that using dropout and regularization is not always redundant, as they can be employed in different procedures of the learning task. Using dropout is especially beneficial for the scenarios in which target object undergoes complicated transformations which is hardly explained by a set of simple features. Furthermore, *MIMIC* outperforms its teachers by using their collective tracking knowledge during its training. This opens a new research front to design more advanced tracker fusion techniques.

Handling partial and full occlusions in the *OAPFT* have been addressed by integrating two different mechanisms, occlusion flags for full occlusions and occlusion masks for partial occlusions. However, occlusion masks can be used in an arbitrary appearance-based tracker to improve its observation and gain some robustness against partial occlusion. In the next part of the study, we elaborate this explicit occlusion detection mechanism for visual tracking. Our algorithm utilizes the foreground ratio as the random variable in different regions of an observation, and empirically estimates the probability distribution of this variable. Using these distributions, an occlusion probability mask is constructed to evaluate the reliability of various regions of an observation. This mask was proved to be successful in managing occlusions and safely updating the target template throughout the tracking sequence. This algorithm is tested intensively with different occlusion scenarios and showed significant tracking improvement.

As we examined the effectiveness of gridding in the construction of occlusion mask, we wonder if it is possible to enhance other features using this mechanism and we select color histogram as a ubiquitous feature in computer vision. The idea involves decomposing a bounding box into a regular grid, in order to incorporate more spatial information and embed local salient color details into the histogram. Two gridding schemes are proposed to accommodate different requirements, when the background data is/is not available. Then a comprehensive study over the existing (dis)similarity measures is conducted and many ideas from other computer vision realms such as image retrieval and texture analysis are brought into object tracking. Using two simulation experiments, we first study the performance of (dis)similarity functions in finding relevant target using color histogram, while discarding other objects to successfully track the designated object. The results signify the usefulness of gridding to improve the tracking performance. It showed that almost any combination of histograms and similarity measures are boosted by gridding. Based on the results it is advisable to use grid-based weighted HOC when the background data is available, but the other version is also superior to normal HOC in most of the cases. In the second experiment we emulate a tracking scenario and illustrate the compatibility of different template update schemes for various dissimilarity measures. It has been found that while all model updates surveyed in this paper improves the performance of template matching, none of them is completely suitable for proposed gridding schemes.

Also the results revealed that, calculating template as the average of the last 5 observations, outperformed other methods.

In summary, standing on the shoulder of a thorough literature review (Chapter 2), we proposed a mode switching tracker, *OAPFT*, to handle various types of occlusion (Chapter 3). It employs a recursive Bayesian inference to predict the eminent full occlusions and prepare the tracker to deal with it by expanding its search area and freezing the template to prevent its corruption. Feature selection and tuning for this tracker has also been automated by a framework that strives to imitate teacher trackers (Chapter 4). This scheme requires a few tracking data samples and select the most effective feature set out of a feature pool. *OAPFT* also utilizes an occlusion mask to improve the observation. This data-driven occlusion mask decompose the observation into regular grid, and provide a occlusion probability for each cell to weigh the observation in the corresponding region. This technique is designed in an add-on fashion to work with an arbitrary appearance-based tracker (Chapter 5). The idea of gridding is then applied on color histograms to provide a more robust feature (Chapter 6).

There are several directions to enhance various parts of this study. The performance of the *OAPFT* could be further improved by exploring better features such as those provided by convolutional neural networks which are ground breaking in many computer vision domains [112] or introducing an adaptive weight to each feature channel [147]. Incorporate the confidence of each data channel (i.e., each extracted feature) into the tracking is another way to enhance the feature-based observation o this tracker.

Combining the proposed occlusion mask and proposed gridded color histogram (and other gridded features) can enhance the observation quality even more. Other color spaces, irregular grids, optimal griddings, and uncontrolled tracking scenarios can further enhance this proposed feature.

In this study, the depth information is not actively used for occlusion handling. If one uses depth feature smartly, occlusion problem may be solved more easily. To this end, we promote using depth information to enhance the observation, sampling region, and occlusion reasoning as another research direction.

The feature selection and tuning part of the study can also be expanded from several directions. Incorporating feature complexity into fitness function of the *MIMIC* would enable the system to find the most computational inexpensive solution to imitate a tracker. Using a richer feature pool, and inclusion of features obtained from automatic feature extraction routines in that can be another way to augment the system. Furthermore, in long tracking scenarios, the future tuning and selection can work along the target tracker to replace it with a faster, more intuitive *MIMIC* tracker with acceptable tracking accuracy.

Finally, occlusion in visual tracking demands dedicated evaluation frameworks and benchmarks to study off-the-shelf tracker under various occlusion cases. Further investigations and surveys on occlusion are required and databases covering all aspects and circumstances of occlusions are yet to be made. Additionally more detailed criteria provide more insights into the dynamics of the tracker during and after occlusion and help designing better trackers.

A
Experimental Details of OAPFT

"If you can't explain it simply, you don't understand it well enough."

— Albert Einstein

A.1 Detailed Analysis of Videos

In this section, the results of all trackers over all five videos used in the paper are presented. Following a brief overview of the dataset configuration and rival algorithms, the videos are presented one-by-one with a quick review on their characteristics. Then the result of the trackers are presented along with a discussion over the performance of them.

As mentioned before, this study uses five videos from Princeton Tracking Dataset [108]. Authors of this paper aimed to standardize a uniform evaluation criteria for tracker comparison, having occlusion evaluation in mind. The dataset contains 100 sequences, acquired with Microsoft Kinect, with 640×480 pixels 8-bit RGB color images and same size 8-bit normalized depth map, and tried to cover various types of occlusion with real-world indoor setup (office, room, library, shop, sports, concourse, fields). The targets are comprised of humans, animals or relatively rigid objects and their initial position is provided for the first frame. This dataset is publicly available at: `http://tracking.cs.princeton.edu/`. The five manually annotated videos is employed in this paper are: `new_ex_occ4`, `face_occ_5`, `bear_front`, `child_no1` and `zcup_move_1`. Based on the definition of video challenges presented in a recent benchmark over online object trackers [24, Table 2], we attributed each video as follows:

IV	illumination variation;
SV	scale variation - when the target grows or shrinks by the factor of 2 comparing to the initial size;
DEF	non-rigid deformation or articulation;
MB	motion blur;
FM	fast motion - when the target displaces more than 20 pixels between two consecutive frames;
IPR	in-plane rotation;
OPR	out-of-plane rotation;
OV	out-of-view;
BC	background clutter - the background near the target has similar color or texture;
LR	low resolution - target bounding box has less than 400 pixels.

Inspired by a pepaer by [34], the occlusions of the video are divided into following categories:

PTO	partial occlusion;
SAO	self- or articulation occlusion;
TFO	temporal full occlusion - shorter than 3 frames;
PFO	persistent full occlusion;

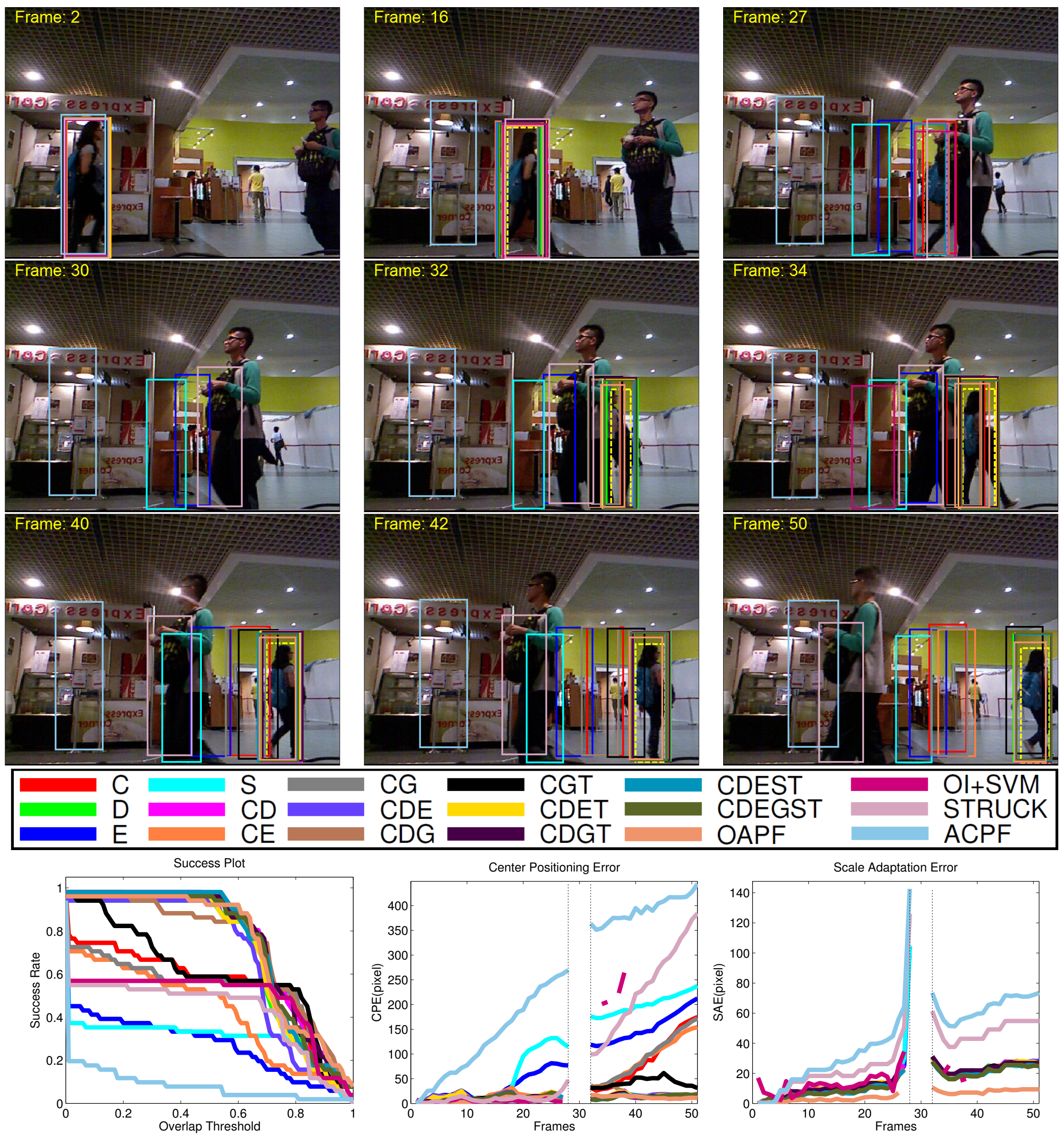


Figure A.1: Qualitative Analysis of sequence `new_ex_occ4`, the ground truth is marked with yellow dashed line.

CPO	complex partial occlusion - including "split and merge" and permanent changes in a key attribute of a part of target;
CFO	complex full occlusion.

As mentioned in the main paper, we compare *OAPFT*, our proposed method using different feature sets with *OI+SVM* [108], *STRUCK* [27] and *ACPF* [44].

A.1.1 Sequence 1: `new_ex_occ4`

Properties

This sequence contains 51 frames, in which a pedestrian walks along a corridor in which she is occluded by another pedestrian for several frames. The provided ground truth of this file is not homogeneous because in some frames it shrinks suddenly to cover the small unoccluded part of the target. A ground truth file is called homogeneous if the size of the target is kept consistent during the tracking and do not change drastically between consecutive frames. Furthermore, the box scales are erroneous and in some cases stretches beyond the size of the image (the corrected version is utilized in this experiment). Additionally the initial bounding

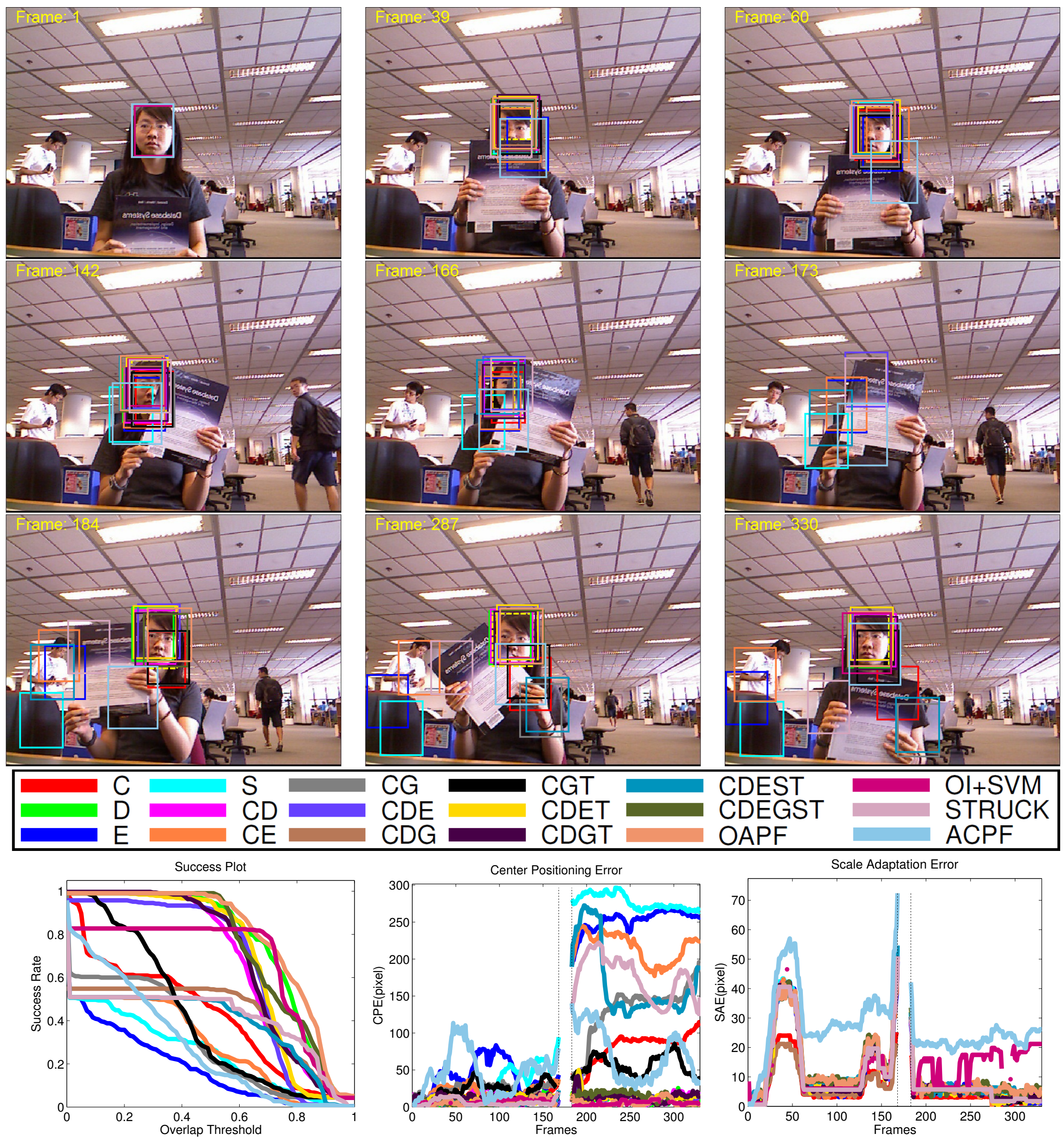


Figure A.2: Qualitative Analysis of sequence `face_occ_5`, the ground truth is marked with yellow dashed line.

box is different from the one mentioned in the ground truth file, thus the values from ground truth is used for the initialization of the trackers throughout this experiment – the case holds for the other videos. This is the most challenging video of these five because of the complex full occlusion in which the target and the occluder are similar in color patterns and shape (and as a result similar HOG), and cross each other so close that the noisy depth values almost have similar values. The sequence can be attributed by DEF, MB, FM and BC.

Analysis of Trackers

The implementation of *ACPF*, the particle filter tracker, although is similar to our C *tracker*, but it also involves geometric weighting of the pixels in the color histogram, that works well in some scenarios, but suffers severely in the presence of background clutter. Another difference is the use of adaptive binning in our implementation of color histogram. We used k-means clustering of input image in first frame with 40 clusters and used the centroid centers as histogram bins in color histogram, where *ACPF* used $8 \times 8 \times 8$ bins for RGB channels which leads to many zero entries in the resulting feature vector. Furthermore, KL-divergence performs better for discriminating color histograms and guides the particle filter far better than Bhattacharyya distance used in *ACPF*.

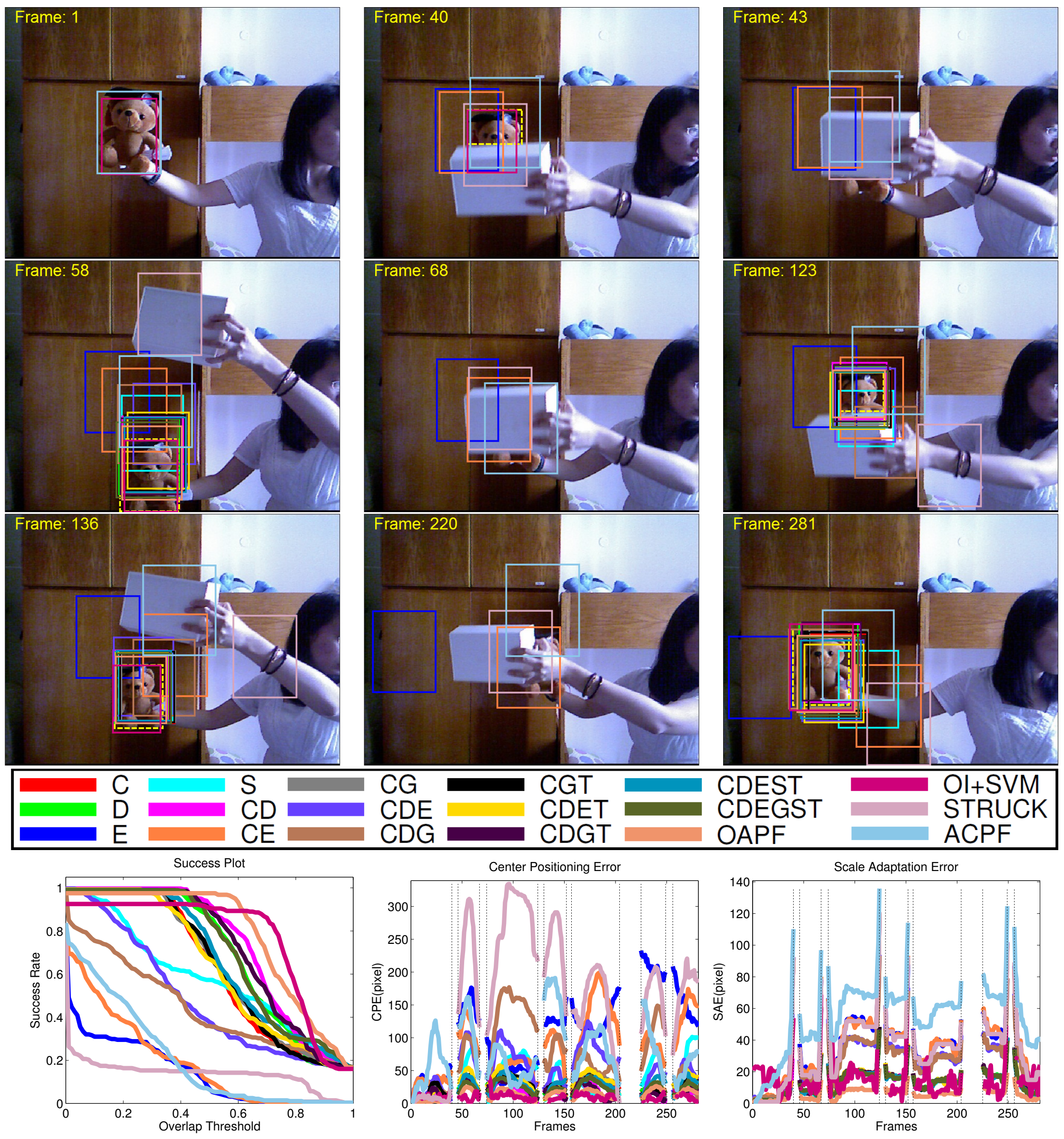


Figure A.3: Qualitative Analysis of sequence `bear_front`, the ground truth is marked with yellow dashed line.

By having a close look on Figure A.1 it is obvious that in Frame 30 with the emergent occlusion and sudden shrinkage of target bounding box, both our proposed method, *OAPFT*, and *OI+SVM* transit into occlusion mode. The occlusion indicator of *OI+SVM* detects a sudden drop in the size of bins around the target nominal depth and declares occlusion. Looking into *OAPFT* internal dynamics reveals that many particles couldn't find a probable candidate for following and occlusion mark particles dominates the population, resulting in occlusion declaration for the tracker. A few frames before occlusion, in frame 27, some of trackers based on the corresponding features still respond to the object or background. For instance tracker E (edge-only) converged to a non-target part of the scene due to background edge clutter, which is solved later with more particles going to occlusion state in the succeeding frames.

With the reappearance of the target in frame 32, the occlusion state is resolved because the number of occluded particles and their probability don't exceed the fixed threshold, δ_{occ} and some of the trackers are evoked. Due to the expanded search zone, the estimation of the target is generated from a wide area, where some of them maybe not relevant in the case of cluttered features (e.g., edge) that cause the immediate estimate of the tracker to be less accurate (refer to trackers E and S in the frame 32). Tracker *OI+SVM* could not recover from this complex occlusion and maintain the occlusion state for most of the following frames.

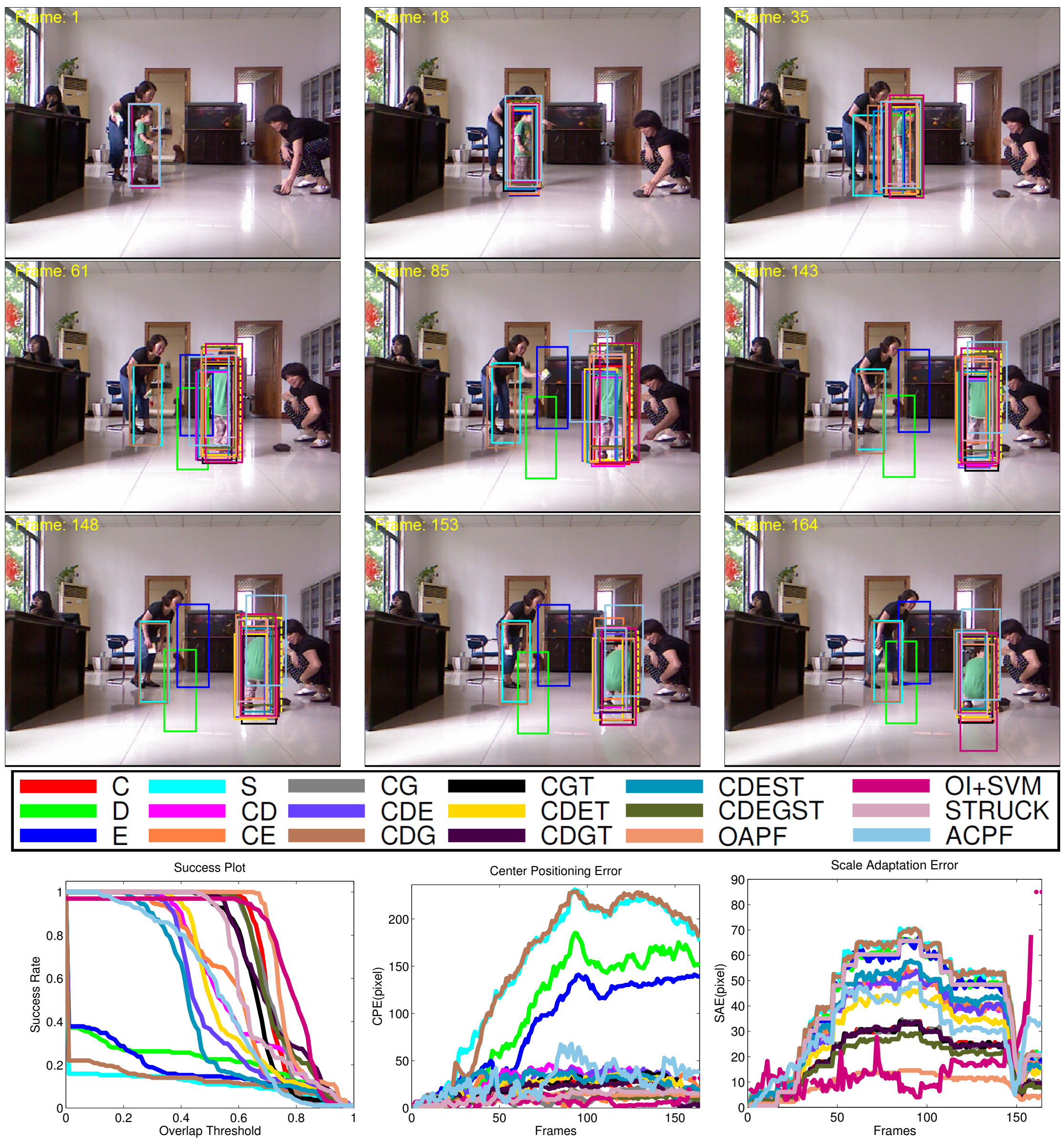


Figure A.4: Qualitative Analysis of sequence `child_no1`, the ground truth is marked with yellow dashed line.

If the feature is not expressive enough for the scenario, their corresponding tracker would lose the target, as it is seen for single-featured trackers or a combination of them (e.g. trackers C, E, S, CE and CG in frame 42). Anyway, the quick recovery from occlusion for well described trackers is evident from frame 40 where trackers start to follow the target again. As mentioned earlier, having enough descriptive power in the form of features, is an essential factor for a successful tracking. For example in this sequence, color, edges and shape features and the pure combinations are not adequate to describe the target and the corresponding tracker chase the background or similar object. But when features with covering different aspects of the object are teamed up with each other, the weakness of each is covered e.g., CGS. Also there is a high probability that one of the features can solve the tracking video almost well, and in combination with other features, its accuracy could be improved. This is the case observed for depth feature in this videos. Finally it should be mentioned that including more features is not always a good solution as it might weaken the positive effect of successful features in the video. For instance tracker CDEST becomes unstable and is unable to find the target quickly after a partial occlusion. This emphasizes the fact that feature selection still has many depths to dive in. Table A.1 compares all the algorithms and provide more insight in this regards by comparing the AUC and CPE values.

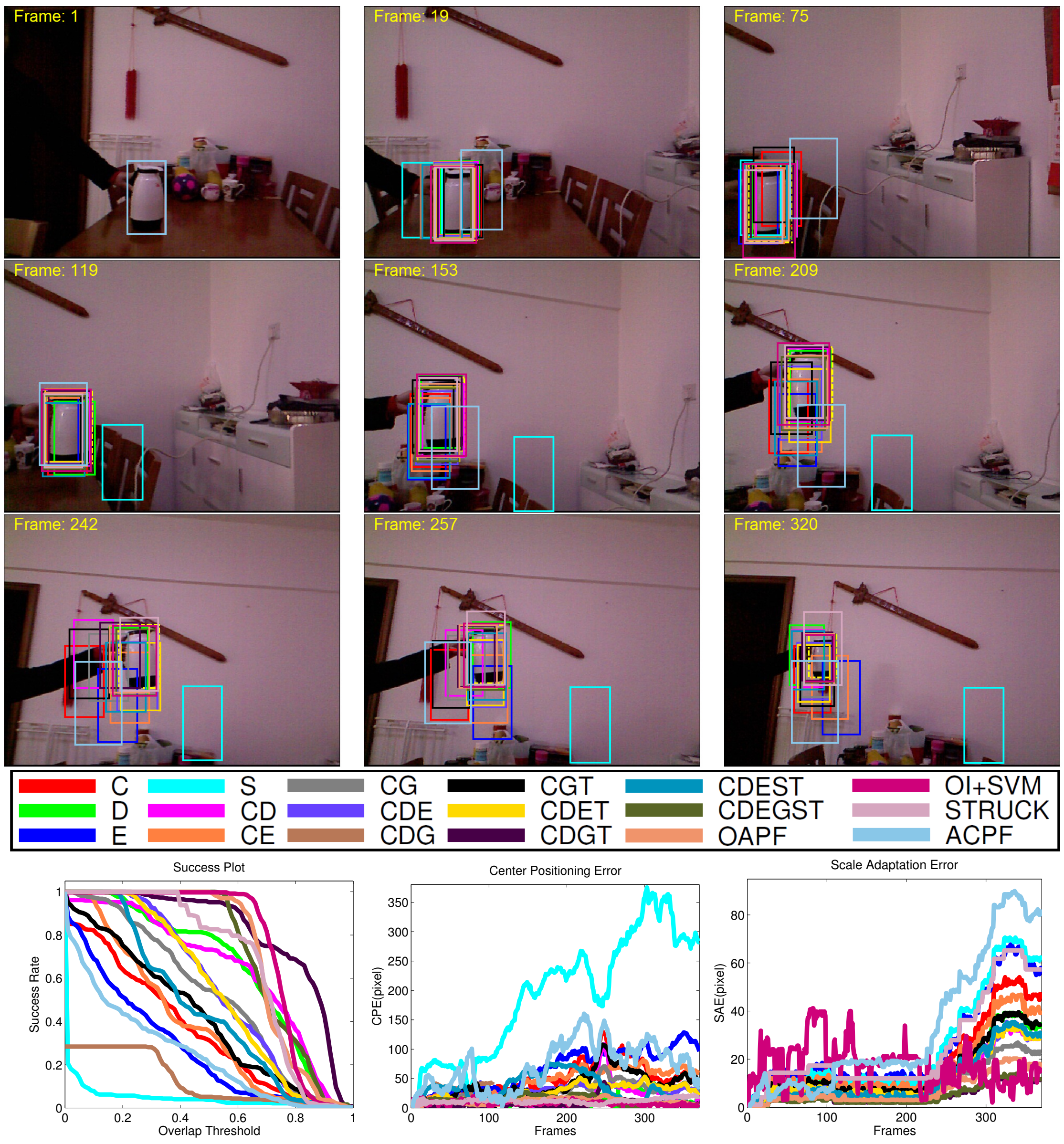


Figure A.5: Qualitative Analysis of sequence `zcup_move_1`, the ground truth is marked with yellow dashed line.

A.1.2 Sequence 2: `face_occ_5`

Properties

This sequence involves 330 frames, having human face as the target, and a book is moved in front of the target to cover it partially from different angles as well as causing a full occlusion. The background is an office with moving people, but the target remains still throughout the video. The provided ground truth of this video is not homogeneous because its size changes to accommodate the partial occlusions and tries to embody only the visible portion of the target. This sequence contains persistent occlusion, and several partial occlusions and the background has clutter and other distractions (BC).

Analysis of Trackers

The result of the experiment is illustrated in Figure A.2. In this video, the geometric distribution of color pixels used in *ACPF* caused this tracker to deviate from the target as it is evident in frame 39. Additionally due to presence of many weak edges in the background, tracker E was not successful in tracking the target as it is observed that the tracker lost the target in frame 39 and although found it later in frame 60, did not

Table A.1: Tracker evaluation for sequence `new_ex_occ4`.

Tracker	cc[#]	*AUC*	*CPE*	*SAE*	*MI*	*FT*	*MT*	*FPS*
C	🟥	55.71	39.75	14.43	**0.0**	1.0	10.0	7.1
D	🟩	71.29	11.09	13.67	**0.0**	3.0	**0.0**	13.4
E	🟦	27.20	78.36	16.16	3.0	**0.0**	25.0	4.4
S	🟦	31.20	103.36	15.17	3.0	**0.0**	29.0	0.6
CD	🟪	**76.00**	10.63	14.81	**0.0**	1.0	**0.0**	6.7
CE	🟧	42.98	42.80	13.45	**0.0**	3.0	12.0	3.2
CG	⬜	53.27	41.21	13.91	**0.0**	2.0	12.0	4.8
CDE	🟪	68.41	13.21	13.34	**0.0**	3.0	**0.0**	3.0
CGS	🟫	74.67	12.16	12.89	**0.0**	1.0	**0.0**	0.6
CGT	⬛	60.90	20.77	13.94	**0.0**	2.0	**0.0**	2.6
CDET	🟨	71.57	12.14	13.78	**0.0**	2.0	**0.0**	2.0
CDGT	🟪	75.22	10.51	14.10	**0.0**	2.0	**0.0**	2.6
CDEST	🟦	74.71	10.83	12.70	**0.0**	1.0	**0.0**	0.5
CDEGST	🟩	73.92	10.72	12.17	**0.0**	2.0	**0.0**	0.5
OAPFT	🟧	74.57	**8.71**	**4.55**	**0.0**	2.0	**0.0**	0.5
OI+SVM	🟥	48.76	20.28	8.29	1.0	17.0	4.0	<0.1
STRUCK	🟪	42.92	96.66	31.37	3.0	**0.0**	20.0	**17.2**
ACPF	🟦	8.73	228.31	41.57	3.0	**0.0**	38.0	0.9

[#] *cc – Color Code for the Tracker*

track the target well due to the template corruption. In frame 142 with another partial occlusion, *ACPF* and S lost the target completely and were not able to recover it again.

Approaching to full occlusion in frame 166, most of the versions of our proposed method along with *OI+SVM* start to transit to occlusion state. In frame 173, in the middle of persistent occlusion however, it is observed that trackers S, E, CE, CDE, CDEST, and *ACPF* were not in occlusion state. Looking at the dynamics of these trackers, it is evident that edges and 3D shape fails to describe the target clearly and histogram of color is mainly attracted to the color of skin and doesn't provide high quality description of the target. This is why these trackers fails to track the target and the fixed l_{occ} used for all videos, was not completely suitable for this video. However, with more descriptive features added to each of them, the tracker works well and detect occlusion successfully (e.g. CDE → CDET and CDEST → CDEGST).

With the target reappearance in frame 183, the occlusion recovery process starts. As it is illustrated in frame 184 most of the trackers that detected occlusion earlier, recovered from the occlusion. However the histogram of color suffered from corrupted template thus trackers C and CG were unable to track the target successfully afterwards. Moreover due to the same problem, tracker CGT fails later in frame 280. Note that tracker CGS although detects the occlusion correctly, but stays in occlusion status till the end of scenario.

Table A.2 supports the qualitative observations. By a close look at *CPE* and *MT* values, it is evident that trackers with high errors lost the target during the tracking. Also tracker CGS with a large *FT* value maintains the occlusion status for many frames during the video. In this video, *OI+SVM* suffers from false tracking state and *STRUCK* lost the target during the persistent full occlusion.

A.1.3 Sequence 3: `bear_front`

Properties

This sequence has 281 frames, in which a teddy bear is being moved around the screen over a same color background. During the course of this sequence a white box is also moved in a random to cover a part or whole body of the bear several times. The provided ground truth of this file is not homogeneous because the ground truth box shrinks size in the case of partial occlusion. This video contains 6 full occlusions from the same occluder with various lengths and directions. The sequence can be attributed by IV, IPR and BC.

Analysis of Trackers

Figure A.3 depicts a few snapshots of the tracker performances handling occlusions. In frame 43, during the full occlusion it is observed that trackers E and CE fails to detect the occlusion. This failure is rooted in

Table A.2: Tracker evaluation for sequence `face_occ_5`.

Tracker	cc[#]	*AUC*	*CPE*	*SAE*	*MI*	*FT*	*MT*	*FPS*
C	🟥	42.37	43.58	5.46	**0.0**	1.0	6.0	9.7
D	🟩	76.23	8.62	9.26	**0.0**	2.0	**0.0**	**14.6**
E	🟦	18.35	133.76	9.32	14.0	**0.0**	148.0	10.1
S	🟦	25.48	139.25	11.16	14.0	**0.0**	148.0	1.7
CD	🟪	67.90	12.26	9.25	**0.0**	2.0	**0.0**	11.3
CE	🟧	29.20	107.65	9.45	14.0	**0.0**	148.0	7.2
CG	⬜	31.75	70.97	9.19	**0.0**	3.0	120.0	7.8
CDE	🟪	65.38	11.93	9.60	14.0	**0.0**	**0.0**	7.8
CGS	🟫	42.94	**4.27**	**4.10**	**0.0**	149.0	**0.0**	1.5
CGT	⬛	41.34	36.61	9.19	**0.0**	3.0	**0.0**	5.0
CDET	🟨	69.45	11.81	9.11	**0.0**	3.0	**0.0**	5.1
CDGT	🟪	69.25	11.59	9.39	**0.0**	1.0	**0.0**	5.1
CDEST	🟦	39.20	78.77	10.97	14.0	**0.0**	148.0	1.4
CDEGST	🟩	74.19	9.05	10.72	**0.0**	3.0	**0.0**	1.3
OAPFT	🟧	**79.16**	6.30	10.85	**0.0**	3.0	**0.0**	1.3
OI+SVM	🟪	66.90	5.00	8.94	**0.0**	57.0	**0.0**	<0.1
STRUCK	🟪	41.55	81.33	9.33	14.0	**0.0**	148.0	11.3
ACPF	🟦	28.93	55.53	25.84	14.0	**0.0**	40.0	1.9

[#] *cc – Color Code for the Tracker*

failure of edge detector to find strong persistent edges throughout the sequence, also due to background color clutter, the color feature is not strong enough to prevent tracker CE from losing the target. This argument hols for *ACPF* that solely relies on color cues to track the target and get absorbed to the background in this scenario. Additionally *STRUCK* lost the target permanently during this occlusion and tracks the occluder afterwards as it is seen in frame 58. Due to corrupted template, trackers C and CE in frame 58 is seen to stick to the background, and fail to recover the target.

Furthermore the trackers CGS and CDE faced problems in tracking as it is seen in frame 68, while the former recovers the target slowly due to lack of enough expressive power of its features, the latter has a corrupted template. Moreover tracker S fails to recover from the occlusion completely as shown in frame 220 and lose the target later in this scenario.

Investigating evaluation Table A.3, the above mentioned observations are proved as tracker E has a high *CPE* and *MT* values, CE almost has the same situation, late recovery of CDE impacts its *SAE* and *CPE* vlues and CGS has a high *MT* value. In this scenario *OI+SVM* performs well thanks to its object detector scheme.

A.1.4 Sequence 4: `child_no1`

Properties

This sequence has 164 frames, in which the subject moves across a furnished room and crouch in the end. The provided ground truth of this file is homogeneous. This video involves articulated body motions and deformations, and self-occlusions of the target. While the target is easily distinguishable using color features, other features such as depth suffers from heavy noise and clutter. The sequence can be attributed as by DEF, OPR, and BC.

Analysis of Trackers

As it is seen in Figure A.4, the trackers generally perform well over this sequence. This sequence has a depth clutter with many outlier values and noise in the measurements and same depth objects in the field of view, hence tracker D with only histogram of depth as the leading cue lost the target early in the scenario. Additionally the background contains various complicated structures and details which introduces many unwanted elements into edge and gradient feature-space that distract trackers using such kind of cues. For instance tracker E and CGS suffers from this clutter and lost the target in early frames on the sequence. As an advanced feature relying on depth information, 3D shape also fails to provide robust information to

Table A.3: Tracker evaluation for sequence `bear_front`.

Tracker	cc[#]	*AUC*	*CPE*	*SAE*	*MI*	*FT*	*MT*	*FPS*
C	🟥	63.85	22.81	10.84	**0.0**	2.0	**0.0**	6.1
D	🟩	70.02	18.43	10.78	**0.0**	3.0	**0.0**	**13.4**
E	🟦	14.58	90.48	28.48	46.0	**0.0**	94.0	7.8
S	🟦	58.55	33.13	10.68	**0.0**	3.0	**0.0**	0.8
CD	🟪	74.14	14.98	10.95	**0.0**	1.0	**0.0**	6.0
CE	🟧	19.20	72.98	29.02	46.0	**0.0**	37.0	4.3
CG	⬜	64.56	22.28	10.78	**0.0**	3.0	**0.0**	4.6
CDE	🟪	48.70	45.47	22.28	**0.0**	1.0	**0.0**	4.2
CGS	🟫	44.84	61.16	21.80	**0.0**	3.0	33.0	0.7
CGT	⬛	64.39	22.11	10.70	**0.0**	3.0	**0.0**	2.7
CDET	🟨	64.13	24.54	10.85	**0.0**	2.0	**0.0**	2.7
CDGT	🟪	72.53	15.46	10.76	**0.0**	3.0	**0.0**	2.7
CDEST	🟦	67.40	21.10	10.65	**0.0**	3.0	**0.0**	0.7
CDEGST	🟩	70.85	17.63	10.75	**0.0**	3.0	**0.0**	0.6
OAPFT	🟧	**78.90**	10.86	**4.76**	**0.0**	7.0	**0.0**	0.6
OI+SVM	🟪	76.99	**7.84**	11.14	1.0	20.0	**0.0**	2.1
STRUCK	🟪	14.12	142.03	28.07	46.0	**0.0**	154.0	9.4
ACPF	🟦	21.01	66.43	45.52	46.0	**0.0**	23.0	0.9

[#] *cc – Color Code for the Tracker*

tracker thus tracker S was not successful to track the targets. Tracker CDEST with three ineffective features, struggles to track the target but the poor performance of this tracker is observed throughout the sequence. Although the rival algorithms track the target successfully, they don't maintain an optimal size for their

Table A.4: Tracker evaluation for sequence `child_no1`.

Tracker	cc[#]	*AUC*	*CPE*	*SAE*	*MI*	*FT*	*MT*	*FPS*
C	🟥	72.12	18.36	20.07	**0.0**	**0.0**	**0.0**	9.5
D	🟩	22.22	104.67	39.64	**0.0**	**0.0**	103.0	**16.1**
E	🟦	17.72	86.55	39.57	**0.0**	**0.0**	101.0	9.0
S	🟦	11.58	150.53	43.53	**0.0**	**0.0**	138.0	1.6
CD	🟪	59.59	26.68	32.11	**0.0**	**0.0**	**0.0**	9.5
CE	🟧	59.05	15.61	32.68	**0.0**	**0.0**	**0.0**	6.4
CG	⬜	71.56	12.09	19.91	**0.0**	**0.0**	**0.0**	7.1
CDE	🟪	52.48	26.75	32.15	**0.0**	**0.0**	**0.0**	6.6
CGS	🟫	13.20	151.24	43.40	**0.0**	**0.0**	128.0	1.5
CGT	⬛	67.12	22.73	19.91	**0.0**	**0.0**	**0.0**	4.5
CDET	🟨	56.84	22.52	28.09	**0.0**	**0.0**	**0.0**	4.4
CDGT	🟪	72.04	16.98	19.95	**0.0**	**0.0**	**0.0**	4.4
CDEST	🟦	46.90	27.13	34.98	**0.0**	**0.0**	**0.0**	1.2
CDEGST	🟩	73.16	10.52	17.77	**0.0**	**0.0**	**0.0**	1.2
OAPFT	🟧	77.21	10.92	**9.15**	**0.0**	**0.0**	**0.0**	1.2
OI+SVM	🟪	**77.30**	**6.81**	13.97	**0.0**	5.0	**0.0**	<0.1
STRUCK	🟪	66.41	11.73	39.55	**0.0**	**0.0**	**0.0**	11.8
ACPF	🟦	53.83	29.86	30.31	**0.0**	**0.0**	**0.0**	1.6

[#] *cc – Color Code for the Tracker*

target boxes. *OI+SVM* and *ACPF* without explicit size adaptation mechanism and *STRUCK* with no such mechanism, are unable to the scale change of the target during the scenario. This is especially evident in frame 143. This can be inferred from Table A.4 easily. On the other hand tracker BCDEGST benefits from "2D Projection Confidence" feature and performs well regarding scale adaptation.

A.1.5 Sequence 5: `zcup_move_1`

Properties

This sequence has 370 frames, in which a pot is moved in different directions first and then away from the camera. The camera in this scenario is moving, and the lighting condition is poor. The provided ground truth of this file is homogeneous. Moving camera, same color background and out-of-plane rotation which causes self-occlusions are the most prominent attributes of this sequence (OPR, BC).

Analysis of Trackers

A glance at Figure A.5 reveals that the target is hard to distinguish from the background only using color feature. Due to this reason *ACPF* tracker fails to track the object from the early frames of the video (i.e. frame 19). Although tracker C benefits from color histogram with adaptive binning, but it also suffers from this problem and is unable to keep the main focus on the target (r.t frames 209, 242, 257). Additionally due to hard edges introduced by the background and soft edges of the target, trackers E and CE show poor tracking performance during the video. Additionally tracker CGT performs not very well for this sequence (e.g. frame 257).

Evaluation Table A.5 supports the claim that color, edge and 3D shape features fails in this scenario as it is inferred from trackers C, E and S respectively. On the other hand histogram of depth works very well for the scenarios (r.t tracker D) and is effectively improved with the combination of other features in trackers CD, CDE and CDET. Also the HOG feature was effective not only in localizing the target, but also in enforcing the scale adaptation to the system as it is observed in trackers CG and CDGT. However the poor performance of 3D shape tracker enforce tracker CGS to stop tracking (indicated by high *FT* value) as the probability of all non-occluded particles become very small. Moreover this feature reduce the performance of the trackers including it as it is seen from tracker CDEST against CDET.

The most important observation in this table is the failure of 2D confidence projection feature to improve the scale adaptation of tracker BCDEGST. This is due to the fact that the background subtraction employed here (inspired from [141]), assumes a static background, the condition which is is violated by the camera movement in this scenario. However using the proposed feature normalization procedure, the impact of this failure is significantly reduced on the tracker scale adaptation.

Detector based tracker such as *STRUCK* and *OI+SVM* performs well in this scenario, while the color-based particle filter tracker, *ACPF*, suffer from low contrast. This low contrast cause non-adaptive bin color histograms to experience value concentration on a few number of bins that make it difficult to separate resembling bounding box from irrelevant ones.

Table A.5: Tracker evaluation for sequence `zcup_move_1`.

Tracker	cc[#]	*AUC*	*CPE*	*SAE*	*MI*	*FT*	*MT*	*FPS*
C	🟥	35.89	47.70	19.42	**0.0**	**0.0**	41.0	8.4
D	🟩	65.35	14.84	14.50	**0.0**	**0.0**	**0.0**	14.4
E	🟦	27.59	65.61	24.61	**0.0**	**0.0**	40.0	9.9
S	🟦	5.22	194.09	24.70	**0.0**	**0.0**	291.0	1.4
CD	🟪	64.96	16.75	12.04	**0.0**	**0.0**	12.0	8.0
CE	🟧	38.62	44.26	17.24	**0.0**	**0.0**	**0.0**	6.7
CG	⬜	52.68	26.49	9.79	**0.0**	**0.0**	**0.0**	6.2
CDE	🟪	56.04	25.78	12.29	**0.0**	**0.0**	**0.0**	6.7
CGS	🟫	12.82	8.96	**1.39**	**0.0**	265.0	**0.0**	1.3
CGT	⬛	41.93	38.08	14.54	**0.0**	**0.0**	19.0	4.0
CDET	🟨	55.40	26.41	12.24	**0.0**	**0.0**	**0.0**	4.2
CDGT	🟪	**81.67**	**6.95**	4.85	**0.0**	**0.0**	**0.0**	4.0
CDEST	🟦	46.50	33.20	11.93	**0.0**	**0.0**	**0.0**	1.1
CDEGST	🟩	72.57	12.21	4.86	**0.0**	**0.0**	**0.0**	1.1
OAPFT	🟧	72.67	11.16	7.27	**0.0**	**0.0**	**0.0**	1.0
OI+SVM	🟥	75.79	8.46	17.88	**0.0**	1.0	**0.0**	<0.1
STRUCK	🟪	68.36	11.96	24.70	**0.0**	**0.0**	**0.0**	**17.3**
ACPF	🟦	25.23	71.76	33.12	**0.0**	**0.0**	54.0	1.6

[#] *cc – Color Code for the Tracker*

A.2 Video Demonstration

In order to better illustrate the comparison between trackers, supplementary videos are available in the first author webpage: http://ishiilab.jp/member/meshgi-k/oapft.html. These videos compare the performance of different feature sets for our proposed tracker, visualization of different features extracted from sequence, and comparison between our algorithm and *ACPF*, *OI+SVM* and *STRUCK*.

B
Visual Tracking Challenges

"All sorts of computer errors are now turning up. You'd be surprised to know the number of doctors who claim they are treating pregnant men."
— Isaac Asimov

Occlusion happens when a portion (or all) of the object disappears from the observed scene due to obstruction of the camera line-of-sight to the target. This phenomena appears due to numerous reasons, as it's apparent from Table B.1 and is frequently seen in real-world video sequences. Complex interactions between objects, large displacement, cast shadows and dense crowds are instances of the cases where temporally and spatially significant occlusions may occur. Appearance change during occlusion, persistent occlusions, re-emergence of occluded object, emergence of the initially-occluded object in the middle of the scene, temporal silhouette merge of multiple objects, singular targets which are observed as spitted, and fragmented trajectory of a targets are from many difficult problems that trackers should address to handle occlusion completely.

Table B.1: Main challenges of visual tracking [23, 24, 187]

Source	Cause	Details
Camera	Motion	Fast Motion
		Motion Blur [24]
		Shake
	Distortion	Color
		Lens (fish eye, perspective)
	Zoom	Focus Blur
		Scale Change
	Sensor	Low Frame Rate [3]
		Low bit-depth
		Low Resolution
	Multi Camera	Overlapping views
		Viewpoint
		Tracking across cameras with non-overlapping views [107]
	Illumination	Dark Scenes
		Low contrast
		Glare
Illumination	Ambient	Abrupt Ambient Light Change
	Directed Light	Intensity Change
		Abrupt Number of Sources Change
		Direction Change
	Backlight	
Object	Illumination Artifacts	Blind Spots (IR absorbent, etc)
		Reflection
		Spectularities
		Shadows (Motion Pattern, Shadow-to-Object resemblence, Feature Detection Errors)
		Transparency
	Transformations	Rigid Body Transformations (Translation, In-plane Rotation)
		Non Rigid Shape Deformation and Articulations
		Out-of-Plane Rotations
		Affine Transformation
	Size	Small Size (Low resolution problem)
		Scale Variations
		Perspective Effect
	Multiple Target	Varying number of objects
		Similar objects (Confusion problem)
		Different classes of object [43]
		Identification Problem
		Inter-object interactions
		Inter-object occlusions
	Motion	Fast Motion
		Motion Blur
		Sudden Velocity Change
		Sudden Direction Change
	Model	Albedo change [123]
		Physical model shortcomings [187]
		Appearance pattern change
	Pose	Complicated pose
		Pose Variation
		Self-Occlusion
Scene	Appearance	Appearance pattern Change
		Background Clutter
		Unconstrained
	Non-target Objects	Motion
		Far-field (Low resolution problem)
		Scene-to-target occlusions
	Illumination Artifacts	Reflections
		Specularity
		Shadows
		Transparency
		Atmospheric Turbulence
Constraints	Time	Real-time requirements
	Field of View	Object partially out-of-view (Shear problem) [123]
		Target appearing in the middle of scene
	Sensor Data Fusion	Resolution incompatibility
		Channel reliability
		Correspondence problem

C
Features

"The solution often turns out more beautiful than the puzzle."

— Richard Dawkins

Finding image features to reliably track an object has always been a hot topic in computer vision [140]. Specially for *MIMIC*, which requires various successful features to shadow a target tracker, the construction of a rich feature pool is crucial. In our implementation, we provide more than 50 features to the system, all of which are supported by a wealth of literature.

Features described in the well-cited benchmark by [188] including shape context [189], steerable filters [190], PCA-SIFT [191], differential invariants [192], spin images [193], SIFT [194], complex filters [195], moment invariants [196] were the first ones to populate this pool. Other features such as variations of color histogram benchmarked in [38], Gabor filters and Haar wavelets with different orientations and receptor field, various edge detectors, affine invariant feature detectors [197], various texture detectors (using FFT, LBP [145], Q-LBP [198], etc.), various gradient-based features such as GLAC [199], HOG [143] and its variations (e.g., the implementation in the VL-feat package) are among other implemented features.

Modern features optimized for object detection are further included to this pool: Bag of Visual Words [200, 201], Bag-of-X [202], kernel codebooks [203], grid-enhanced color histograms [38], Fisher vectors and Advanced Fisher Vectors [204], reconstruction based approaches like LLC [17].

A list of these faetures are presented as follows:

1. Histogram of Color in RGB colorspace with Regular binning of size $8 \times 8 \times 8$ and $64 \times 64 \times 64$ bins (2 features)
2. Histogram of Color in HSV colorspace with Regular binning of size $8 \times 8 \times 4$ and $64 \times 64 \times 64$ bins (2 features)
3. Enhanced HOC based on the gridding scheme described in [38] (13 Features)
4. Haar-Like Wavelets (17 Features)
5. FFT Filters (20 Features)
6. R-SIFT
7. A-SIFT
8. RLBP
9. Difference of Gaussians
10. SFOP
11. LIOP

12. DAISY
13. SURF
14. Histogram of Oriented Optical Flow
15. Poselets
16. Spatial Envelope
17. Object bank
18. ORB
19. Differential invariants [192]
20. Spin images [193]
21. Complex filters [195]
22. Moment invariants [196]
23. Affine invariant feature detectors [197]
24. Bag of Visual Words [200, 201]
25. Bag-of-X [202]
26. Local Energy based Shape Histogram (LESH) [205]
27. GLOH [188]
28. Bag-of-Words of SIFT and HOG with different Parameters (16 Features)
29. L1-Sparse Coding
30. LCC
31. LLC
32. LLC+Codebook Derivatives
33. LCC+Codebook Derivatives
34. Hierarchical Spatial Coding
35. Local Coordinate Coding
36. Extended Local Coordinate Coding
37. Differential Sparse Coding
38. Discriminative Sparse Coding
39. BOW + Spatial Pyramid Model (SPM)
40. Deformable Part Model (DPM)
41. SPM + DPM
42. ScSPM
43. Reconfigurable Models (RBoW)
44. GIST
45. Hierarchical Matching Pursuit
46. Visual Concept (VC)
47. Super Vector Coding
48. Fisher Vector (of SIFT)
49. Extended Fisher Vector
50. Fisher Vector (of LCS)
51. Fisher Vector (SIFT + LCS)

52. Q-LBP
53. Bag-of-X
54. PCS-SIFT [191]
55. Kernel codebooks [203]
56. Gray-Level Co-occurrence Matrix (GLCM)
57. BRISK
58. Spatio-Temporal Auto-Correlation of Gradients (STACOG) [206]
59. Gradient Local Auto-Correlation (GLAC) [199]
60. Histogram of Oriented PDF Gradients [207]

Bibliography

[1] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” *Acm Computing Surveys*, vol. 38, no. 4, p. 13, 2006.

[2] K. Cannons, “A review of visual tracking,” *York Univ., Toronto, Canada, Tech. Rep. CSE-2008-07*, 2008.

[3] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, “A survey of appearance models in visual object tracking,” *TIST*, vol. 4, no. 4, p. 58, 2013.

[4] G. Pulford, “Taxonomy of multiple target tracking methods,” in *Radar, Sonar and Navigation, IEE Proc.*, vol. 152, no. 5. IET, 2005, pp. 291–304.

[5] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, “Survey of pedestrian detection for advanced driver assistance systems,” *PAMI*, vol. 32, no. 7, pp. 1239–1258, 2010.

[6] J. Candamo, M. Shreve, D. B. Goldgof, D. B. Sapper, and R. Kasturi, “Understanding transit scenes: A survey on human behavior-recognition algorithms,” *TIST*, vol. 11, no. 1, pp. 206–224, 2010.

[7] B. Zhan, D. N. Monekosso, P. Remagnino, S. A. Velastin, and L.-Q. Xu, “Crowd analysis: a survey,” *MVA*, vol. 19, no. 5-6, pp. 345–357, 2008.

[8] C. B. Santiago, A. Sousa, M. L. Estriga, L. P. Reis, and M. Lames, “Survey on team tracking techniques applied to sports,” in *AIS’10*. IEEE, 2010, pp. 1–6.

[9] B. Han and L. Davis, “On-line density-based appearance modeling for object tracking,” in *ICCV’05*, vol. 2. IEEE, 2005, pp. 1492–1499.

[10] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, “Incremental learning for robust visual tracking,” *IJCV*, vol. 77, no. 1-3, pp. 125–141, 2008.

[11] W. Zhong, H. Lu, and M.-H. Yang, “Robust object tracking via sparsity-based collaborative model,” in *CVPR’12*. IEEE, 2012, pp. 1838–1845.

[12] B. Babenko, M.-H. Yang, and S. Belongie, “Visual tracking with online multiple instance learning,” in *CVPR’09*. IEEE, 2009, pp. 983–990.

[13] K. Zhang, L. Zhang, and M.-H. Yang, “Real-time compressive tracking,” in *ECCV’12*. Springer, 2012, pp. 864–877.

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “Exploiting the circulant structure of tracking-by-detection with kernels,” in *ECCV’12*. Springer, 2012, pp. 702–715.

[15] J. Kwon and K. M. Lee, “Tracking of abrupt motion using wang-landau monte carlo estimation,” in *ECCV’08*. Springer, 2008, pp. 387–400.

[16] X. Zhou, Y. Lu, J. Lu, and J. Zhou, “Abrupt motion tracking via intensively adaptive markov-chain monte carlo sampling,” *TIP*, vol. 21, no. 2, pp. 789–801, 2012.

[17] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, “Locality-constrained linear coding for image classification,” in *CVPR’10*, 2010, pp. 3360–3367.

[18] J. Ning, L. Zhang, D. Zhang, and C. Wu, “Scale and orientation adaptive mean shift tracking,” *IET Computer Vision*, vol. 6, no. 1, pp. 52–61, 2012.

[19] T. B. Dinh, N. Vo, and G. Medioni, “Context tracker: Exploring supporters and distracters in unconstrained environments,” in *CVPR’11*. IEEE, 2011, pp. 1177–1184.

[20] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-learning-detection,” *PAMI*, vol. 34, no. 7, pp. 1409–1422, 2012.

[21] H. T. Nguyen and A. W. Smeulders, “Tracking aspects of the foreground against the background,” in *ECCV’04*. Springer, 2004, pp. 446–456.

[22] S. Avidan, “Ensemble tracking,” *PAMI*, vol. 29, no. 2, pp. 261–271, 2007.

[23] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, “Visual tracking: An experimental survey,” *PAMI*, vol. 36, no. 7, pp. 1442–1468, 2014.

[24] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in *CVPR’13*. IEEE, 2013, pp. 2411–2418.

[25] ——, “Object tracking benchmark,” *PAMI*, 2015.

[26] S. Salti, A. Cavallaro, and L. Di Stefano, “Adaptive appearance modeling for video tracking: survey and evaluation,” *TIP*, 2012.

[27] S. Hare, A. Saffari, and P. H. Torr, “Struck: Structured output tracking with kernels,” in *ICCV’11*. IEEE, 2011, pp. 263–270.

[28] D. M. Chu and A. W. Smeulders, “Color invariant surf in discriminative object tracking,” in *ECCV’10w*. Springer, 2012, pp. 62–75.

[29] J. Winn and A. Blake, “Generative affine localisation and tracking,” in *NIPS’04*, 2004, pp. 1505–1512.

[30] Y. Huang and I. Essa, “Tracking multiple objects through occlusions,” in *CVPR’05*, vol. 2. IEEE, 2005, pp. 1051–1058.

[31] M. P. Kumar, P. Ton, and A. Zisserman, “Obj cut,” in *CVPR’05*, vol. 1. IEEE, 2005, pp. 18–25.

[32] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti, and R. Bolle, “Appearance models for occlusion handling,” *IMVC*, vol. 24, no. 11, pp. 1233–1243, 2006.

[33] X. Zhou, Y. Li, and B. He, “Game-theoretical occlusion handling for multi-target visual tracking,” *PR*, vol. 46, no. 10, pp. 2670–2684, 2013.

[34] R. Vezzani, C. Grana, and R. Cucchiara, “Probabilistic people tracking with appearance models and occlusion classification: The ad-hoc system,” *PRL*, vol. 32, no. 6, pp. 867–877, 2011.

[35] C. Wang, M. de La Gorce, and N. Paragios, “Segmentation, ordering and multi-object tracking using graphical models,” in *ICCV’12*. IEEE, 2009, pp. 747–754.

[36] K. Meshgi and S. Ishii, “The state-of-the-art in handling occlusions for visual object tracking,” *IEICE Transactions on Information and Systems*, vol. E98-D, no. 7, pp. 1260–1274, 2015.

[37] K. Meshgi, S.-i. Maeda, S. Oba, H. Skibbe, Y.-z. Li, and S. Ishii, “Occlusion aware particle filter tracker to handle complex and persistent occlusions,” *J. of Computer Vision and Image Understanding*, 2015.

[38] K. Meshgi and S. Ishii, “Expanding histogram of colors with gridding to improve tracking accuracy,” in *MVA’15*. IEEE, 2015, pp. 475–479.

[39] P. F. Gabriel, J. G. Verly, J. H. Piater, and A. Genon, “The state of the art in multiple object tracking under occlusion in video sequences,” in *Advanced Concepts for Intelligent Vision Systems*, 2003, pp. 166–173.

[40] B. Y. Lee, L. H. Liew, W. S. Cheah, and Y. C. Wang, “Occlusion handling in videos object tracking: A survey,” in *IOP Conference Series: Earth and Environmental Science*, vol. 18, no. 1. IOP Publishing, 2014, p. 012020.

[41] P. Guha, A. Mukerjee, and V. K. Subramanian, “Formulation, detection and application of occlusion states (OC-7) in the context of multiple object tracking,” in *AVSS’11*. IEEE, 2011, pp. 191–196.

[42] O. Javed and M. Shah, “Tracking and object classification for automated surveillance,” in *ECCV’02*. Springer, 2002, pp. 343–357.

[43] B. Bose, X. Wang, and E. Grimson, “Multi-class object tracking algorithm that handles fragmentation and grouping,” in *CVPR’07*. IEEE, 2007, pp. 1–8.

[44] K. Nummiaro, E. Koller-Meier, and L. Van Gool, “An adaptive color-based particle filter,” *IMVC*, vol. 21, no. 1, pp. 99–110, 2003.

[45] J. MacCormick and A. Blake, "A probabilistic exclusion principle for tracking multiple objects," *IJCV*, vol. 39, no. 1, pp. 57–71, 2000.

[46] F. Porikli, O. Tuzel, and P. Meer, "Covariance tracking using model update based on lie algebra," in *CVPR'06*, vol. 1. IEEE, 2006, pp. 728–735.

[47] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *CVPR'06*, vol. 1. IEEE, 2006, pp. 798–805.

[48] B. Han and L. S. Davis, "Probabilistic fusion-based parameter estimation for visual tracking," *CVIU*, vol. 113, no. 4, pp. 435–445, 2009.

[49] P. Chockalingam, N. Pradeep, and S. Birchfield, "Adaptive fragments-based tracking of non-rigid objects using level sets," in *ICCV'09*. IEEE, 2009, pp. 1530–1537.

[50] H. T. Nguyen and A. W. Smeulders, "Fast occluded object tracking by a robust appearance filter," *PAMI*, vol. 26, no. 8, pp. 1099–1104, 2004.

[51] M. Yang, J. Yuan, and Y. Wu, "Spatial selection for attentional visual tracking," in *CVPR'07*. IEEE, 2007, pp. 1–8.

[52] 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 *CVPR'09*. IEEE, 2009, pp. 1208–1215.

[53] J. Kwon, K. M. Lee, and F. C. Park, "Visual tracking via geometric particle filtering on the affine group with optimal importance functions," in *CVPR'09*. IEEE, 2009, pp. 991–998.

[54] D. Chen and J. Yang, "Robust object tracking via online dynamic spatial bias appearance models," *PAMI*, vol. 29, no. 12, pp. 2157–2169, 2007.

[55] F. Yang, H. Lu, and Y.-W. Chen, "Bag of features tracking," in *ICPR'10*. IEEE, 2010, pp. 153–156.

[56] S. Zhang, H. Yao, X. Sun, and X. Lu, "Sparse coding based visual tracking: Review and experimental comparison," *PR*, vol. 46, no. 7, pp. 1772–1788, 2013.

[57] F. Pernici, "Facehugger: The alien tracker applied to faces," in *ECCV'12w*. Springer, 2012, pp. 597–601.

[58] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust online appearance models for visual tracking," *PAMI*, vol. 25, no. 10, pp. 1296–1311, 2003.

[59] H. T. Nguyen and A. W. Smeulders, "Robust tracking using foreground-background texture discrimination," *IJCV*, vol. 69, no. 3, pp. 277–293, 2006.

[60] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *CVPR'10*. IEEE, 2010, pp. 1269–1276.

[61] S. Kwak, W. Nam, B. Han, and J. H. Han, "Learning occlusion with likelihoods for visual tracking," in *ICCV'011*, 2011.

[62] I. Matthews, T. Ishikawa, S. Baker *et al.*, "The template update problem," *PAMI*, vol. 26, no. 6, pp. 810–815, 2004.

[63] T. Zhang, B. Ghanem, C. Xu, and N. Ahuja, "Object tracking by occlusion detection via structured sparse learning," in *CVPR'13w*. IEEE, 2013, pp. 1033–1040.

[64] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *BMVC'06*, vol. 1, no. 5, 2006, p. 6.

[65] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *ECCV'08*, 2008.

[66] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," in *ICCV'11*. IEEE, 2011, pp. 1323–1330.

[67] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," in *CVPR'12*. IEEE, 2012, pp. 1940–1947.

[68] X. Mei, H. Ling, Y. Wu, E. P. Blasch, and L. Bai, "Efficient minimum error bounded particle resampling L1 tracker with occlusion detection," *TIP*, vol. 22, no. 7, pp. 2661–2675, 2013.

[69] L. Čehovin, M. Kristan, and A. Leonardis, "An adaptive coupled-layer visual model for robust visual tracking," in *ICCV'11*. IEEE, 2011, pp. 1363–1370.

[70] Z. Kalal, J. Matas, and K. Mikolajczyk, "Pn learning: Bootstrapping binary classifiers by structural constraints," in *CVPR'10*. IEEE, 2010, pp. 49–56.

[71] M. Godec, P. M. Roth, and H. Bischof, "Hough-based tracking of non-rigid objects," *CVIU*, vol. 117, no. 10, pp. 1245–1256, 2013.

[72] J. Fan, Y. Wu, and S. Dai, "Discriminative spatial attention for robust tracking," in *ECCV'10*. Springer, 2010, pp. 480–493.

[73] J. H. Yoon, D. Y. Kim, and K.-J. Yoon, "Visual tracking via adaptive tracker selection with multiple features," in *ECCV'12*. Springer, 2012, pp. 28–41.

[74] M. Spengler and B. Schiele, "Towards robust multi-cue integration for visual tracking," *MVA*, vol. 14, no. 1, pp. 50–58, 2003.

[75] W. Choi and S. Savarese, "A unified framework for multi-target tracking and collective activity recognition," in *ECCV'12*. Springer, 2012, pp. 215–230.

[76] J. Kwon and K. M. Lee, "Tracking by sampling trackers," in *ICCV'11*. IEEE, 2011, pp. 1195–1202.

[77] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *IJCV*, vol. 56, no. 3, pp. 221–255, 2004.

[78] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *PAMI*, vol. 25, no. 5, pp. 564–577, 2003.

[79] A. Milan, S. Roth, and K. Schindler, "Continuous energy minimization for multi-target tracking," *PAMI*, vol. 36, no. 1, 2014.

[80] H. Grabner and H. Bischof, "On-line boosting and vision," in *CVPR'06*, vol. 1. IEEE, 2006, pp. 260–267.

[81] T. Bando, T. Shibata, K. Doya, and S. Ishii, "Switching particle filters for efficient visual tracking," *RAS*, vol. 54, no. 10, pp. 873–884, 2006.

[82] M. Rodriguez, S. Ali, and T. Kanade, "Tracking in unstructured crowded scenes," in *ICCV'09*. IEEE, 2009, pp. 1389–1396.

[83] M. Yang, Y. Wu, and G. Hua, "Context-aware visual tracking," *PAMI*, vol. 31, no. 7, pp. 1195–1209, 2009.

[84] H. Grabner, J. Matas, L. Van Gool, and P. Cattin, "Tracking the invisible: Learning where the object might be," in *CVPR'10*. IEEE, 2010, pp. 1285–1292.

[85] N. Thome and S. Miguet, "A robust appearance model for tracking human motions," in *AVSS'05*. IEEE, 2005, pp. 528–533.

[86] A. Amer, "Voting-based simultaneous tracking of multiple video objects," *CSVT*, vol. 15, no. 11, pp. 1448–1462, 2005.

[87] T. Zhao and R. Nevatia, "Tracking multiple humans in complex situations," *PAMI*, vol. 26, no. 9, pp. 1208–1221, 2004.

[88] J. Sullivan and S. Carlsson, "Tracking and labelling of interacting multiple targets," in *ECCV'06*. Springer, 2006, pp. 619–632.

[89] Z. Khan, T. Balch, and F. Dellaert, "Multitarget tracking with split and merged measurements," in *CVPR'05*, vol. 1. IEEE, 2005, pp. 605–610.

[90] A. Genovesio and J.-C. Olivo-Marin, "Split and merge data association filter for dense multi-target tracking," in *ICPR'04*, vol. 4. IEEE, 2004, pp. 677–680.

[91] S.-W. Joo and R. Chellappa, "A multiple-hypothesis approach for multiobject visual tracking," *TIP*, vol. 16, no. 11, pp. 2849–2854, 2007.

[92] M. Bray, P. Kohli, and P. H. Torr, "Posecut: Simultaneous segmentation and 3d pose estimation of humans using dynamic graph-cuts," in *ECCV'06*. Springer, 2006, pp. 642–655.

[93] J. Malcolm, Y. Rathi, and A. Tannenbaum, “Multi-object tracking through clutter using graph cuts,” in *ICCV’07*. IEEE, 2007, pp. 1–5.

[94] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.

[95] L. Zhu, Y. Chen, A. Yuille, and W. Freeman, “Latent hierarchical structural learning for object detection,” in *CVPR’10*. IEEE, 2010, pp. 1062–1069.

[96] P. Perez, J. Vermaak, and A. Blake, “Data fusion for visual tracking with particles,” *Proc. of the IEEE*, vol. 92, no. 3, pp. 495–513, 2004.

[97] S. Walk, N. Majer, K. Schindler, and B. Schiele, “New features and insights for pedestrian detection,” in *CVPR’10*. IEEE, 2010, pp. 1030–1037.

[98] M. Enzweiler, A. Eigenstetter, B. Schiele, and D. M. Gavrila, “Multi-cue pedestrian classification with partial occlusion handling,” in *CVPR’10*. IEEE, 2010, pp. 990–997.

[99] P. Dollár, Z. Tu, P. Perona, and S. Belongie, “Integral channel features,” in *BMVC’09*, vol. 2, no. 3, 2009, p. 5.

[100] G. Duan, H. Ai, and S. Lao, “A structural filter approach to human detection,” in *ECCV’10*. Springer, 2010, pp. 238–251.

[101] W. Ouyang, X. Zeng, and X. Wang, “Modeling mutual visibility relationship in pedestrian detection,” in *CVPR’13*. IEEE, 2013, pp. 3222–3229.

[102] B. Wu and R. Nevatia, “Tracking of multiple, partially occluded humans based on static body part detection,” in *CVPR’06*, vol. 1. IEEE, 2006, pp. 951–958.

[103] D. Tang and Y.-J. Zhang, “Combining mean-shift and particle filter for object tracking,” in *ICIG’11*. IEEE, 2011, pp. 771–776.

[104] S. L. Dockstader and A. M. Tekalp, “Multiple camera tracking of interacting and occluded human motion,” *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1441–1455, 2001.

[105] A. Mittal and L. S. Davis, “M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene,” *IJCV*, vol. 51, no. 3, pp. 189–203, 2003.

[106] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, “Multicamera people tracking with a probabilistic occupancy map,” *PAMI*, vol. 30, no. 2, pp. 267–282, 2008.

[107] O. Javed, K. Shafique, Z. Rasheed, and M. Shah, “Modeling inter-camera space–time and appearance relationships for tracking across non-overlapping views,” *CVIU*, vol. 109, no. 2, pp. 146–162, 2008.

[108] S. Song and J. Xiao, “Tracking revisited using rgbd camera: Unified benchmark and baselines,” in *ICCV’13*, 2013.

[109] Z. Duan, Z. Cai, and J. Yu, “Occlusion detection and recovery in video object tracking based on adaptive particle filters,” in *Control and Decision Conference, 2009*. IEEE, 2009, pp. 466–469.

[110] Y. Wu, T. Yu, and G. Hua, “Tracking appearances with occlusions,” in *CVPR’03*, vol. 1. IEEE, 2003, pp. I–789.

[111] W. Hu, X. Zhou, M. Hu, and S. Maybank, “Occlusion reasoning for tracking multiple people,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 1, pp. 114–121, 2009.

[112] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “Cnn features off-the-shelf: an astounding baseline for recognition,” in *CVPR’14w*. IEEE, 2014, pp. 512–519.

[113] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, and et al., “Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol,” *PAMI*, vol. 31, no. 2, pp. 319–336, 2009.

[114] B. Y. Lee, L. H. Liew, W. S. Cheah, and Y. C. Wang, “Simulation videos for understanding occlusion effects on kernel based object tracking,” in *Computer Science and its Applications*, 2012.

[115] CAVIAR, “Context aware vision using image-based active recognition,” in *http://groups.inf.ed.ac.uk/vision/CAVIAR/*, 2004.

[116] R. Collins, X. Zhou, and S. K. Teh, “An open source tracking testbed and evaluation web site,” in *PETS’05*, 2005, pp. 17–24.

[117] A. T. Nghiem, F. Bremond, M. Thonnat, and V. Valentin, “Etiseo, performance evaluation for video surveillance systems,” in *AVSS’07*. IEEE, 2007, pp. 476–481.

[118] R. Vezzani and R. Cucchiara, “Visor: Video surveillance on-line repository for annotation retrieval,” in *ICME’08*. IEEE, 2008, pp. 1281–1284.

[119] A. Nilski, “An evaluation metric for multiple camera tracking systems: the i-lids 5th scenario,” in *SPIE Europe Security and Defence*. International Society for Optics and Photonics, 2008, pp. 711 907–711 907.

[120] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, “A mobile vision system for robust multi-person tracking,” in *CVPR’08*. IEEE, 2008, pp. 1–8.

[121] C. Wojek, S. Walk, and B. Schiele, “Multi-cue onboard pedestrian detection,” in *CVPR’09*. IEEE, 2009, pp. 794–801.

[122] P. Dollár, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: An evaluation of the state of the art,” *PAMI*, vol. 34, no. 4, pp. 743–761, 2012.

[123] D. M. Chu and A. W. Smeulders, “Thirteen hard cases in visual tracking,” in *AVSS’10*. IEEE, 2010, pp. 103–110.

[124] D. Baltieri, R. Vezzani, and R. Cucchiara, “3dpes: 3d people dataset for surveillance and forensics,” in *Proc. Int’l. ACM Workshop on Human Gesture and Behavior Understanding*. ACM, 2011, pp. 59–64.

[125] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, and T. Vojir, “The visual object tracking vot2014 challenge results,” in *ECCV’14w*, 2014.

[126] B. Liu, J. Huang, L. Yang, and C. Kulikowsk, “Robust tracking using local sparse appearance model and k-selection,” in *CVPR’11*. IEEE, 2011, pp. 1313–1320.

[127] X. Jia, H. Lu, and M.-H. Yang, “Visual tracking via adaptive structural local sparse appearance model,” in *CVPR’12*. IEEE, 2012, pp. 1822–1829.

[128] C. Bao, Y. Wu, H. Ling, and H. Ji, “Real time robust L1 tracker using accelerated proximal gradient approach,” in *CVPR’12*. IEEE, 2012, pp. 1830–1837.

[129] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai, “Minimum error bounded efficient L1 tracker with occlusion detection,” in *CVPR’11*, 2011.

[130] S. Stalder, H. Grabner, and L. Van Gool, “Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition,” in *ICCVw’09*. IEEE, 2009, pp. 1409–1416.

[131] P. Guha, A. Mukerjee, and K. Venkatesh, “Ocs-14: you can get occluded in fourteen ways,” in *IJCAI’11*, vol. 22, no. 1, 2011, p. 1665.

[132] W. Choi, C. Pantofaru, and S. Savarese, “A general framework for tracking multiple people from a moving camera,” *PAMI*, 2013.

[133] M. Isard and A. Blake, “Condensation—conditional density propagation for visual tracking,” *IJCV*, vol. 29, no. 1, pp. 5–28, 1998.

[134] X. Mei and H. Ling, “Robust visual tracking using L1 minimization,” in *ICCV’09*. IEEE, 2009, pp. 1436–1443.

[135] P. Brasnett, L. Mihaylova, D. Bull, and N. Canagarajah, “Sequential monte carlo tracking by fusing multiple cues in video sequences,” *IMVC*, vol. 25, no. 8, pp. 1217–1227, 2007.

[136] L. Mihaylova, P. Brasnett, N. Canagarajah, and D. Bull, “Object tracking by particle filtering techniques in video sequences,” *Advances and Challenges in Multisensor Data and Information*, pp. 260–268, 2007.

[137] J. Pan and B. Hu, “Robust object tracking against template drift,” in *ICIP’07*, vol. 3. IEEE, 2007, pp. III–353.

[138] M. K. Pitt and N. Shephard, "Filtering via simulation: auxiliary particle filters," *J. of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, 1999.

[139] K. Meshgi, S.-i. Maeda, S. Oba, and S. Ishii, "Fusion of multiple cues from color and depth domains using occlusion aware bayesian tracker," *Tech. Rep. of Neuro Computing*, vol. 113, no. 500, pp. 127–132, 2014.

[140] J. Shi and C. Tomasi, "Good features to track," in *CVPR'94*, 1994.

[141] B. Lo and S. Velastin, "Automatic congestion detection system for underground platforms," in *Intelligent Multimedia, Video and Speech Processing, 2001. Intl. Symp.* IEEE, 2001, pp. 158–161.

[142] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the hausdorff distance," *PAMI*, vol. 15, no. 9, pp. 850–863, 1993.

[143] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR'05*, vol. 1. IEEE, 2005, pp. 886–893.

[144] A. E. Johnson, "Spin-images: a representation for 3-d surface matching," Ph.D. dissertation, Citeseer, 1997.

[145] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *PAMI*, vol. 24, no. 7, pp. 971–987, 2002.

[146] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *IJCV*, vol. 88, no. 2, pp. 303–338, 2010.

[147] J.-f. Dou and J.-x. Li, "Robust visual tracking base on adaptively multi-feature fusion and particle filter," *Optik*, vol. 125, no. 5, pp. 1680–1686, 2014.

[148] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *ECCV'14w*, 2014.

[149] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *PAMI*, vol. 33, no. 3, pp. 500–513, 2011.

[150] M. Camplani, S. Hannuna, M. Mirmehdi, D. Damen, A. Paiement, L. Tao, T. Burghardt, and U. Bristol, "Real-time rgb-d tracking with depth scaling kernelised correlation filters and occlusion handling," 2015.

[151] R. Martín and J. M. Martínez, "Evaluation of bounding box level fusion of single target video object trackers," in *Hybrid Artificial Intelligence Systems*. Springer, 2014, pp. 200–210.

[152] D. P. Chau, M. Thonnat, F. Bremond, and E. Corvee, "Online parameter tuning for object tracking algorithms," *IMVC*, vol. 32, no. 4, pp. 287–302, 2014.

[153] P. Viola and M. J. Jones, "Robust real-time face detection," *IJCV*, vol. 57, no. 2, pp. 137–154, 2004.

[154] H.-T. Chen, T.-L. Liu, and C.-S. Fuh, "Probabilistic tracking with adaptive feature selection," in *ICPR'04*, vol. 2. IEEE, 2004, pp. 736–739.

[155] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[156] H.-P. P. Schwefel, *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

[157] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Aerospace Defense Sensing, Simulation, and Controls*. International Society for Optics and Photonics, 2001, pp. 95–102.

[158] D. P. Papadopoulos, A. D. Clarke, F. Keller, and V. Ferrari, "Training object class detectors from eye tracking data," in *ECCV'14*. Springer, 2014, pp. 361–376.

[159] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[160] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[161] I. Leichter, M. Lindenbaum, and E. Rivlin, "A general framework for combining visual trackers–the "black boxes" approach," *IJCV*, vol. 67, no. 3, pp. 343–363, 2006.

[162] B. Zhong, H. Yao, S. Chen, R. Ji, T.-J. Chin, and H. Wang, “Visual tracking via weakly supervised learning from multiple imperfect oracles,” *PR*, vol. 47, no. 3, pp. 1395–1410, 2014.

[163] B. McCane, B. Galvin, and K. Novins, “Algorithmic fusion for more robust feature tracking,” *IJCV*, vol. 49, no. 1, pp. 79–89, 2002.

[164] Y. Gao, R. Ji, L. Zhang, and A. Hauptmann, “Symbiotic tracker ensemble towards a unified tracking framework,” *CSVT*, 2014.

[165] M. Heber, M. Godec, M. Rüther, P. M. Roth, and H. Bischof, “Segmentation-based tracking by support fusion,” *CVIU*, vol. 117, no. 6, pp. 573–586, 2013.

[166] M. Yang and Y. Wu, “Tracking non-stationary appearances and dynamic feature selection,” in *CVPR’05*, vol. 2. IEEE, 2005, pp. 1059–1066.

[167] R. T. Collins, Y. Liu, and M. Leordeanu, “Online selection of discriminative tracking features,” *PAMI*, vol. 27, no. 10, pp. 1631–1643, 2005.

[168] J. Wang, X. Chen, and W. Gao, “Online selecting discriminative tracking features using particle filter,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 1037–1042.

[169] K. Zhang, L. Zhang, and M.-H. Yang, “Real-time object tracking via online discriminative feature selection,” *TIP*, vol. 22, no. 12, pp. 4664–4677, 2013.

[170] M. Johannes, N. Polson, and J. Stroud, “Exact particle filtering and parameter learning,” Working Paper, Tech. Rep., 2006.

[171] C. Nemeth, P. Fearnhead, and L. Mihaylova, “Sequential monte carlo methods for state and parameter estimation in abruptly changing environments,” *Signal Processing, IEEE Transactions on*, vol. 62, no. 5, pp. 1245–1255, 2014.

[172] S. Kwak, T. Lim, W. Nam, B. Han, and J. H. Han, “Generalized background subtraction based on hybrid inference by belief propagation and bayesian filtering,” in *ICCV’11*, 2011.

[173] M. Andriluka, S. Roth, and B. Schiele, “People-tracking-by-detection and people-detection-by-tracking,” in *CVPR’08*, 2008.

[174] Y. Wu, J. Cheng, J. Wang, H. Lu, J. Wang, H. Ling, E. Blasch, and L. Bai, “Real-time probabilistic covariance tracking with efficient model update,” *TIP*, 2012.

[175] Y. Rubner, J. Puzicha, C. Tomasi, and J. M. Buhmann, “Empirical evaluation of dissimilarity measures for color and texture,” *CVIU*, vol. 84, no. 1, pp. 25–43, 2001.

[176] D. Comaniciu and V. Ramesh, “Mean shift and optimal prediction for efficient object tracking,” in *ICIP’00*, vol. 3. IEEE, 2000, pp. 70–73.

[177] A. Zweng, T. Rittler, and M. Kampel, “Evaluation of histogram-based similarity functions for different color spaces,” in *CAIP’11*. Springer, 2011, pp. 455–462.

[178] O. Pele and M. Werman, “The quadratic-chi histogram distance family,” in *ECCV’10*. Springer, 2010, pp. 749–762.

[179] H. Ling and K. Okada, “Diffusion distance for histogram comparison,” in *CVPR’06*, vol. 1. IEEE, 2006, pp. 246–253.

[180] Y. Rubner, C. Tomasi, and L. J. Guibas, “The earth mover’s distance as a metric for image retrieval,” *IJCV*, vol. 40, no. 2, pp. 99–121, 2000.

[181] O. Pele and M. Werman, “Fast and robust earth mover’s distances,” in *ICCV’09*. IEEE, 2009, pp. 460–467.

[182] G. D. Hager, M. Dewan, and C. V. Stewart, “Multiple kernel tracking with ssd,” in *CVPR’04*, vol. 1. IEEE, 2004, pp. I–790.

[183] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, “Image indexing using color correlograms,” in *CVPR’09*. IEEE, 1997, pp. 762–768.

[184] S. T. Birchfield and S. Rangarajan, “Spatiograms versus histograms for region-based tracking,” in *CVPR’05*, vol. 2. IEEE, 2005, pp. 1158–1163.

[185] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *CVPR’06*, vol. 2. IEEE, 2006, pp. 2169–2178.

[186] K. Grauman and T. Darrell, “The pyramid match kernel: Discriminative classification with sets of image features,” in *ICCV’05*, vol. 2. IEEE, 2005, pp. 1458–1465.

[187] A. S. Jalal and V. Singh, “The state-of-the-art in visual object tracking,” *Informatica*, vol. 36, no. 3, 2012.

[188] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *PAMI*, vol. 27, no. 10, pp. 1615–1630, 2005.

[189] S. Belongie, J. Malik, and J. Puzicha, “Shape matching and object recognition using shape contexts,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 4, pp. 509–522, 2002.

[190] W. T. Freeman and E. H. Adelson, “The design and use of steerable filters,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 9, pp. 891–906, 1991.

[191] Y. Ke and R. Sukthankar, “Pca-sift: A more distinctive representation for local image descriptors,” in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, 2004, pp. II–506.

[192] J. J. Koenderink and A. J. van Doorn, “Representation of local geometry in the visual system,” *Biological cybernetics*, vol. 55, no. 6, pp. 367–375, 1987.

[193] S. Lazebnik, C. Schmid, and J. Ponce, “Sparse texture representations using affine-invariant neighborhoods,” in *In Proc. IEEE Conf. Comp. Vision Patt. Recog*. Citeseer, 2003.

[194] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

[195] F. Schaffalitzky and A. Zisserman, “Multi-view matching for unordered image sets, or “how do i organize my holiday snaps?”,” in *European Conference on Computer Vision*, vol. 1. Springer-Verlag, 2002, pp. 414–431.

[196] L. Van Gool, T. Moons, and D. Ungureanu, “Affine/photometric invariants for planar intensity patterns,” in *Computer Vision—ECCV’96*. Springer, 1996, pp. 642–651.

[197] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[198] R. Lan, Y. Zhou, Y. Y. Tang, and C. P. Chen, “Person reidentification using quaternionic local binary pattern,” in *Multimedia and Expo (ICME), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1–6.

[199] T. Kobayashi and N. Otsu, “Image feature extraction using gradient local auto-correlations,” in *Computer Vision–ECCV 2008*. Springer, 2008, pp. 346–358.

[200] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22. Prague, 2004, pp. 1–2.

[201] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos,” in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1470–1477.

[202] L. Ballan, M. Bertini, A. D. Bimbo, L. Seidenari, and G. Serra, “Effective codebooks for human action representation and classification in unconstrained videos,” *Multimedia, IEEE Transactions on*, vol. 14, no. 4, pp. 1234–1245, 2012.

[203] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders, “Kernel codebooks for scene categorization,” in *Computer Vision–ECCV 2008*. Springer, 2008, pp. 696–709.

[204] F. Perronnin, J. Sánchez, and T. Mensink, “Improving the fisher kernel for large-scale image classification,” in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 143–156.

[205] M. S. Sarfraz and O. Hellwich, “Learning probabilistic models for recognizing faces under pose variations,” in *VISAPP International Workshop on Image Mining*, 2008, pp. 122–132.

[206] T. Kobayashi and N. Otsu, “Motion recognition using local auto-correlation of space–time gradients,” *Pattern Recognition Letters*, vol. 33, no. 9, pp. 1188–1195, 2012.

[207] T. Kobayashi, “Bfo meets hog: feature extraction based on histograms of oriented pdf gradients for image classification,” in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 747–754.