Deep Convolutional Correlation Particle Filter for Visual Tracking

Reza Jalil Mozhdehi
Marquette University

DEEP CONVOLUTIONAL CORRELATION PARTICLE FILTER FOR VISUAL TRACKING

by

Reza Jalil Mozhdehi, B.S., M.S.

A Dissertation submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy

Milwaukee, Wisconsin

December 2021

ABSTRACT
DEEP CONVOLUTIONAL CORRELATION PARTICLE FILTER FOR VISUAL TRACKING

Reza Jalil Mozhdehi, B.S., M.S.

Marquette University, 2021

In this dissertation, we explore the advantages and limitations of the application of sequential Monte Carlo methods to visual tracking, which is a challenging computer vision problem. We propose six visual tracking models, each of which integrates a particle filter, a deep convolutional neural network, and a correlation filter. In our first model, we generate an image patch corresponding to each particle and use a convolutional neural network (CNN) to extract features from the corresponding image region. A correlation filter then computes the correlation response maps corresponding to these features, which are used to determine the particle weights and estimate the state of the target. We then introduce a particle filter that extends the target state by incorporating its size information. This model also utilizes a new adaptive correlation filtering approach that generates multiple target models to account for potential model update errors. We build upon that strategy to devise an adaptive particle filter that can decrease the number of particles in simple frames in which there is no challenging scenarios and the target model closely reflects the current appearance of the target. This strategy allows us to reduce the computational cost of the particle filter without negatively impacting its performance. This tracker also improves the likelihood model by generating multiple target models using varying model update rates based on the high-likelihood particles. We also propose a novel likelihood particle filter for CNN-correlation visual trackers. Our method uses correlation response maps to estimate likelihood distributions and employs these likelihoods as proposal densities to sample particles. Additionally, our particle filter searches for multiple modes in the likelihood distribution using a Gaussian mixture model. We further introduce an iterative particle filter that performs iterations to decrease the distance between particles and the peaks of their correlation maps which results in having a few more accurate particles in the end of iterations. Applying K-mean clustering method on the remaining particles determine the number of the clusters which is used in evaluation step and find the target state. Our approach ensures a consistent support for the posterior distribution. Thus, we do not need to perform resampling at every video frame, improving the utilization of prior distribution information. Finally, we introduce a novel framework which calculates the confidence score of the tracking algorithm at each video frame based on the correlation response maps of the particles. Our framework applies different model update rules according to the calculated confidence score, reducing tracking failures caused by model drift. The benefits of each of the proposed techniques are demonstrated through experiments using publicly available benchmark datasets.

ACKNOWLEDGEMENTS

First, I would like to sincerely thank my advisor Dr. Henry Medeiros for his invaluable supervision during my Ph.D. study at Marquette University. He is one of the most knowledgeable and respectful professors I know in academia. I cannot thank him enough for all his continuous support and devotion throughout my study.

My gratitude extends to Dr. Frederick Frigo, Dr. Edwin Yaz, Dr. Richard Povinelli and Dr. Cristinel Ababei for being part of my dissertation committee and providing beneficial feedback. Additionally, I would like to express gratitude to all my colleagues at the COVISS lab.

Finally, My appreciation beyond words goes out to my mother, father and sisters, who are the most important people in my world, for their encouragement and support through my PhD study. I sincerely dedicate my Ph.D. dissertation to them.

CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1
INTRODUCTION

One important field of artificial intelligence (AI) is computer vision, which enables machines to extract meaningful information from images and videos in a manner similar to the human visual system. Some applications of computer vision include activity recognition [5], object tracking [6], autonomous navigation [7], and security and surveillance [8]. Visual target tracking is a challenging computer vision problem, particularly in situations including target occlusions, deformations, and in-plane or out-of-plane rotations. In visual tracking, the size and location of a specific target are provided in the first video frame, and the target is then followed in subsequent frames by estimating its size and position. Fig. 1.1 illustrates an example of the visual tracking problem.

Every field of computer vision has seen considerable progress through the application of deep learning techniques, which currently represent some of the most effective machine learning methods [9, 10]. Machine learning refers to algorithms that are able to learn how to perform their tasks based on observed data, preferably with minimal human intervention [11, 12]. Deep learning [13, 14] entails "deeper" machine learning models, i.e., models that perform more sophisticated inferences by applying multiple levels of transformations to the input data. Deep belief networks [15], deep reinforcement learning [16],

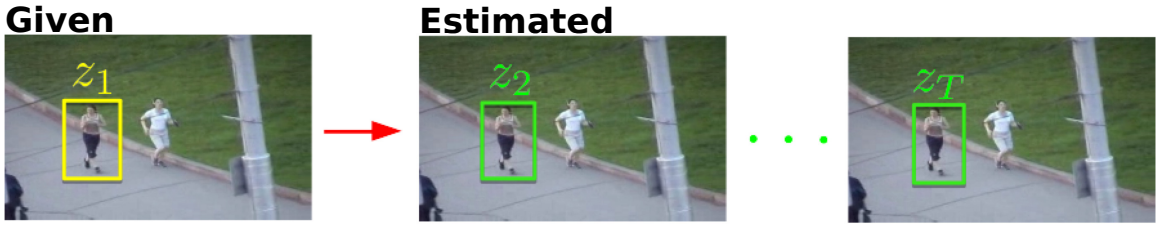


Figure 1.1: Illustration of the visual tracking problem. The yellow bounding box labeled z_1 is provided by the user at the first frame of the video sequence. The algorithm then automatically estimates the green bounding boxes $z_2, \ldots, z_T$ in the subsequent $T-1$ frames of the video.

deep recurrent neural networks [17], and deep convolutional neural networks [18] are some of many different structures used in deep learning models.

The successful application of deep convolutional neural networks (CNN) to object detection tasks [19, 20, 21, 22, 23, 24] has led to an increased interest in the utilization of such networks for visual tracking applications [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]. More specifically, we can consider visual tracking as a classification problem comprising two classes: the foreground, which corresponds to the target object, and background, which comprises the regions of the image that do not contain the target. This dissertation aims to contribute to the improvement of visual tracking techniques utilizing deep CNNs. More specifically, it intends to address the three problems described below.

1.1 Problem statement 1

Much of the performance of CNN-based visual trackers is due to the fact that they are trained utilizing the benchmark datasets on which they are evaluated. One of the most challenging aspects of proposing a visual tacker based on a CNN is to provide enough labeled samples for training the CNN. Neural network models need labeled image patches to learn how to distinguish different kinds of targets from the background. For this reason, they are trained using many labeled samples. If the network is evaluated on the same set of samples used in the training process, its performance is artificially inflated because the network already observed these samples. Thus, the evaluation is not accurate and the network does not show the same performance on unseen video sequences. As an example, state-of-the-art visual trackers such as MDNet [36] and SANet [37] owe much of their performance to the fact that they are trained utilizing the benchmark datasets on which they are evaluated.

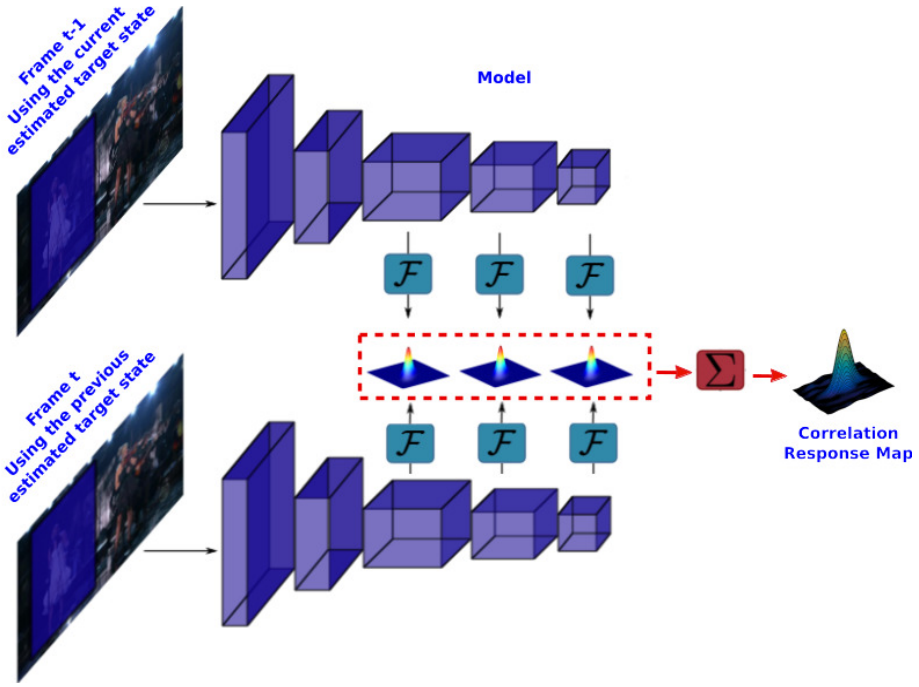


Figure 1.2: Schematic illustration of a typical CNN-correlation visual tracker. The previous frame and its estimated target state are given to a CNN to generate its convolutional features (purple boxes at the bottom). This process is repeated for the current frame using the previous target state (top purple boxes). The current convolutional features are compared with the model through a correlation filter in frequency domain (blue $\mathcal{F}$ boxes). For each layer of convolutional features, a correlation map is calculated (represented by the small normal distribution plots). The sum of all these maps generates the final correlation response map.

1.2 Problem statement 2

Correlation filters do not adapt well to changes in target appearances. One effective mechanism to determine the similarity between an image patch and the target model is to apply correlation filters in conjunction with CNNs [38, 39, 40, 41, 42, 43, 44, 45, 46, 47]. Fig. 1.2 shows how a CNN-correlation visual tracker works. Although applying correlation filters on visual features generated by a CNN is an efficient solution to improve the trackers' performance, especially on datasets not used in the training processes, changes in target appearance cause errors in model generation. These inaccuracies in model update lead to tracking failures that are generally unrecoverable.

1.3 Problem statement 3

Although particle filters have been widely used in visual tracking, correlation-particle filters are not sufficiently robust in sampling and calculating the posterior distributions. Combining particle filters with CNNs and correlation filters is a new approach for visual tracking. In this new combination, not violating Bayesian rules is challenging because correlation response maps create inter-dependencies among the target state and its appearance. Sampling particles and evaluating their relative likelihoods using a rigorous Bayesian update strategy are other challenging problems related to this new framework.

1.4 Objectives

We propose a novel visual tracking framework based on CNNs, correlation filters, and particle filters. Our framework has been one of the first to apply a particle filter to CNN-correlation visual trackers. Our proposed particle filter can efficiently estimate the target location and size accurately using correlation maps to weigh the particles. We propose an adaptive version of this particle filter which increases the number of particles in challenging frames, such as in the presence of target occlusion or motion blur. We also propose a likelihood version of this particle filter which samples particles more accurately after calculating a likelihood distribution for each frame. Using an iterative version of our particle filter, we considerably improve our sampling process and solve the problem of changing the support of the posterior observed in other state-of-the-art particle-correlation trackers. This iterative version enables us not to resample particles at every frame, which reduces the sample impoverishment problem. Additionally, we solve potential errors in model generation by applying novel strategies in the model update process. We apply different clustering methods on the correlation maps corresponding to each of the particles to find potential clusters which allows our model to accommodate multi-modal likelihoods.

Each of these contributions is discussed in further detail in the dissertation objectives described below.

This dissertation has objectives associated to each of the three problems stated above: 1) CNN-based trackers depend the training sets; 2) Correlation filters do not adapt well to changes in target appearances; 3) Correlation-particle filters are not sufficiently robust in sampling and calculating the posterior distributions. MATLAB has been used for the implementation of all proposed algorithms in this dissertation. The Parallel Computing toolbox of MATLAB has been also applied to take advantage of GPU hardware for computational efficiency.

1.4.1 Objective 1

Devise CNN-based trackers that operate successfully on datasets not used in the training process. An effective method to make CNNs robust against unseen samples is to combine CNNs with correlation and particle filters. Trackers based on correlation filters measure the correlation between a target model and an image patch in the frequency domain and are agnostic to the features used to represent the targets. Target models are constructed in the first frame based on a correlation filter designed to operate on the features generated by a CNN using ground truth information. The model is updated by adapting the correlation filter in subsequent frames based on the CNN features and the estimated target state. These target models are compared with the CNN features in the current frames through the correlation filter. The particle filter samples particles to estimate the target positions and sizes, thereby providing more accurate samples for the correlation filters to improve their performance against unseen datasets.

We propose a novel framework for visual tracking based on the integration of a deep convolutional neural network (CNN) and a particle filter. In the proposed framework, the position and the size of the target at each frame is predicted by a particle filter according to a motion model. Particles around the predicted position are then used as input to a CNN-

correlation tracker which adjusts their positions to the most likely target positions. The weights of the particles are then determined using the correlation map of the CNN tracker. Finally, the particles and their weights are used to calculate the position of the target in the current frame [45, 48].

1.4.2 Objective 2

Devise correlation filters that are robust against drifting errors in target model generation. One of the most challenging limitations of correlation filters is the fact that they update the target model based on the estimated target state. Thus, any error in calculating the final target state causes the target model to be incorrectly updated. Novel model update strategies are needed to address this issue.

We propose a new adaptive correlation filter to account for potential errors in model generation. Thus, instead of generating one model which is highly dependent on the estimated target position and size, we generate a variable number of target models based on high likelihood particles, which increases in challenging situations and decreases in less complex scenarios [48, 49]. We also apply different model update rates to each of the high-likelihood particles to create a variable number of models. Some models are useful in challenging frames because they are more influenced by previously generated models, while others are suitable for simple frames because they are less affected by previous models [50].

1.4.3 Objective 3

Devise a recursive Bayesian estimation framework for CNN-correlation visual trackers. There are many challenges in applying particle filters in CNN-correlation trackers. These challenges involve designing principled transition and likelihood models as well as proposal distributions that enable CNN-correlation frameworks to use particle filters for

improving accuracy in the estimation of the target location and the size of its bounding box while respecting Bayesian update rules in the calculation of the posterior distributions.

More technically, we present an adaptive particle filter to decrease the number of particles in simple frames in which there is no challenging scenario and the target model closely reflects the current appearance of the target. In simple frames, target estimation is easier, therefore many particles may converge to the same point. Consequently, the number of particles should be allowed to decrease in these frames. This strategy allows us to resort to resampling only when the number of particles or their corresponding weights are too low in comparison to those computed in the first frame using the ground truth model [50].

Additionally, we propose a novel likelihood particle filter for CNN-correlation visual trackers. Our method uses correlation response maps to estimate likelihood distributions and employs these likelihoods as proposal densities to sample particles. Likelihood distributions are more reliable than proposal densities based on target transition distributions because correlation response maps provide additional information regarding the target's location. Additionally, our particle filter searches for multiple modes in the likelihood distribution, which improves performance in target occlusion scenarios while decreasing computational costs by more efficiently sampling particles. In challenging scenarios, such as those involving motion blur, where only one mode is present but a larger search area may be necessary, our particle filter allows for the variance of the likelihood distribution to increase [51].

Finally, we propose a novel framework for visual tracking based on the integration of an iterative particle filter, a deep convolutional neural network and a correlation filter. The iterative particle filter enables the particles to correct themselves and converge to the correct target position. We employ a novel strategy to assess the likelihood of the particles after by clustering them into multiple modes using the K-means algorithm [52]. Our iterative particle filter ensures a consistent support for the distribution of the posterior. Thus,

we do not need to perform resampling at every video frame and discard prior information [49].

1.5 Dissertation organization

This dissertation consists of nine chapters. Chapter 1 provides an introduction regarding visual tracking and the three main problems addressed by methods proposed in this dissertation. Chapter 2 contains the basic information, main datasets, and existing methods in the literature for visual object tracking. Chapter 3 explains our proposed visual tracking framework based on a deep convolutional neural network, a correlation filter, and a particle filter to find target positions. Chapter 4 proposes a framework to find target sizes as well as to generate multiple target models to improve the performance of correlation filters. Chapter 5 discusses our adaptive particle filter. In Chapter 6, we describe our likelihood particle filter as well as our model to clusters the correlation response maps into a multi-modal likelihood model. Chapter 7 explains our iterative particle filter as well as an alternative method to cluster the particles. In Chapter 8, we explain our framework that uses a short-term memory mechanism and a finite state machine to improve robustness to model drift. Finally, Chapter 9 concludes this dissertation and provides possible directions for future work.

CHAPTER 2
BACKGROUND

In this section, we provide a brief overview of the background and previous well-known algorithms related to the three problems stated above. We also discuss relevant benchmarks and evaluation metrics for visual tracking methods.

2.1 Basic information on visual object tracking

In this section, we provide relevant background information regarding deep neural networks, correlation and particles filter, and how to apply them to visual object tracking.

2.1.1 Deep neural networks

Neural networks having multiple hidden layers are called *deep neural networks*. Deep neural networks have a huge limitation which is their need for a large amount of annotated data for training. Because providing such a large collection of data in computer vision problems is not simple, deep neural networks had not been widely used in computer vision for a long time. However, the introduction of the large publicly available datasets ImageNet [53], which is a dataset of over 14 million images belonging to $21k$ categories, enabled the application of deep neural networks to different fields of computer vision.

Training deep neural networks with huge datasets was another significant, long-standing problem, which was solved by using graphics processing units (GPUs) [54, 55]. Deep belief networks, deep reinforcement learning, deep recurrent neural network,s and deep convolutional neural network are different structures used in deep learning. Among all of them, convolutional neural networks (CNNs) are widely used in computer vision applications. CNNs employ *convolutional filters* to create sparse connections among layers [56]. To provide more details about CNNs, we review the VGG architecture [10], which is a well-known deep neural network used in visual tracking applications.

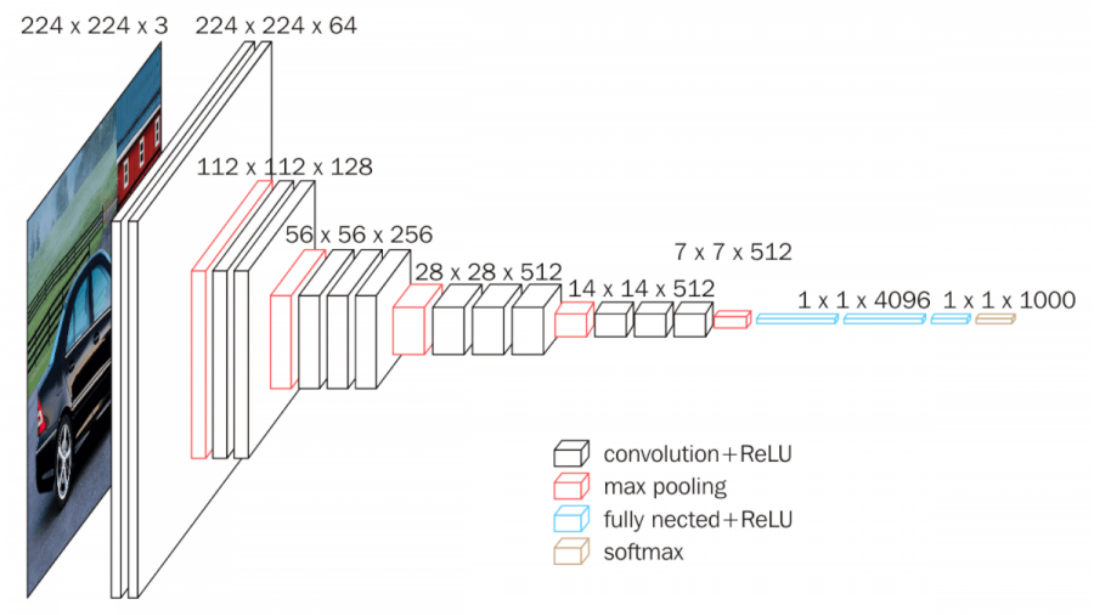


Figure 2.1: Diagram illustrating the architecture of VGG16 introduced in [1].

2.1.1.1 VGG network

VGG is a convolutional neural network model proposed by K. Simonyan and A. Zisserman. The model's accuracy is 92.7% on ImageNet. The reason for its improvement over AlexNet, which is another well-known network in computer vision [9], is that it applies multiple 3×3 kernel filters one after another instead of the large kernel-sized filters used in AlexNet (11×11 and 5×5 in the first and second convolutional layers, respectively) [10]. Fig. 2.1 illustrates the structure of VGG. There are multiple versions of VGG. The first version "VGG16" has 13 convolutional layers while "VGG19" consists of 16 convolutional layers. The size of the input is $224 \times 224 \times 3$. The convolution stride is 1 pixel. Spatial padding is used to preserve the spatial resolution of the image.

In VGG16 for example, the first two convolutional layers have 64 filters. So, their output is a $224 \times 224 \times 64$ volume. After these two convolutional layers, max-pooling is implemented over a 2×2 pixel window with stride 2. Max pooling finds the maximum value among multiple inputs and replaces all of those inputs with that maximum value.

Thus, the output's size reduces to $112 \times 112 \times 64$. The two next convolutional layers having 128 filters result in an output with the size of $112 \times 112 \times 128$, which is decreased to $56 \times 56 \times 128$ by another Max-pooling layer. Next, three convolution layers including 256 filters each followed by Max-pooling result in an output with the size of $28 \times 28 \times 256$. The last step is repeated two more times to generates an output with the size of $7 \times 7 \times 512$ because the convolutional layers include 512 filters. VGG19 has 4 convolutional layers in the last three steps instead of the 3 convolutional layers found in VGG16.

Both versions include three Fully-Connected (FC) layers as well. The first two FC layers have 4096 channels, while the third one has 1000 channels to predict 1000 classes. The network has a soft-max output layer. All hidden layers use the rectified linear units (ReLU) as an activation function. ReLU is effective for hidden layers because its output is 1 for values larger than 0, while it is 0 for negative values. Fig. 2.2 makes a comparison among the two versions of VGG and AlexNet. All the algorithms described in this dissertation use VGG19 to generate convolutional features.

2.1.2 Correlation filters

An effective mechanism to assess the similarity between an image patch and a target model in visual tracking is the correlation filter. Correlation filters produce sharp peaks in their output to localize targets [57]. More specifically, a typical correlation tracker such as [58] learns a discriminative classifier and estimates the translation of target objects by searching for the maximum value of the corresponding correlation response map [44].

Ma et al. in [44] used a correlation filter in conjunction with a deep neural network for the first time in the visual tracking literature. They selected the correlation filter proposed in [32] and the VGG16 network. In their model, let f be layer l of the convolutional feature vector generated by VGG16. f has D channels with size of $M \times N$. The authors train their correlation filter with all the circular shifts of f along its first two dimensions. Consider $f_{m,n}$, where $(m, n) \in \{0, 1, ..., M-1\} \times \{0, 1, ..., N-1\}$ as a shifted sample.

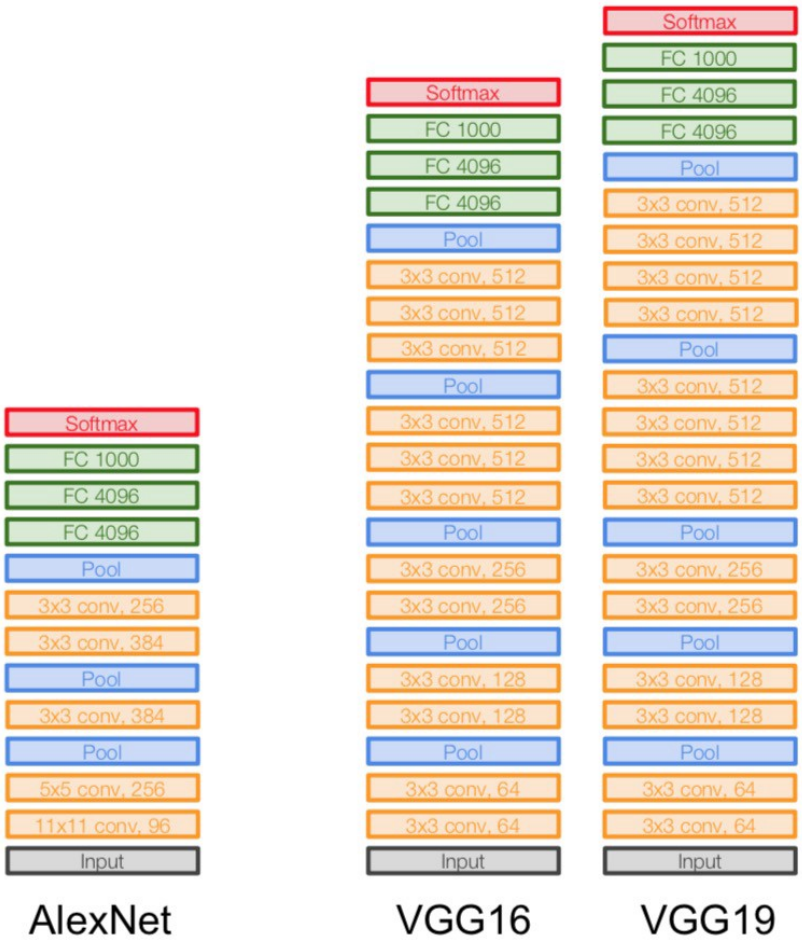


Figure 2.2: A comparison among the VGG16, VGG19, and AlexNet network structures [2].

The authors define a Gaussian function label $g(m,n) = e^{-\frac{(m-M/2)^2(n-N/2)^2}{2\sigma^2}}$ for each shifted sample, where σ is the kernel width. A correlation filter c is then learned by solving the following minimization problem [44]:

$$c^* = \arg\min_c \sum_{m,n} ||c.f_{m,n} - g(m,n)||^2 + \lambda||c||_2^2, \tag{2.1}$$

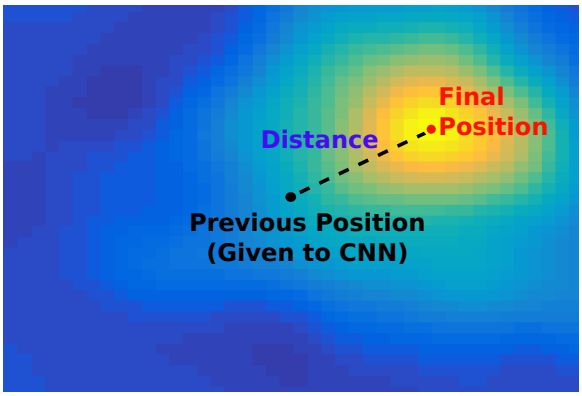


Figure 2.3: Example of a correlation map of a given frame with respect to the previous target position.

where λ is a regularization parameter and $c.f_{m,n} = \sum_{d=1}^{D} c_{m,n,d}^{T} f_{m,n,d}$ [44]. Transferring Eq. 2.1 to the frequency domain results in solving the minimization problem:

$$C^d = \frac{G \odot \bar{F}^d}{\sum_{i=1}^{D} F^i \odot \bar{F}^i + \lambda}, \tag{2.2}$$

where the capital letters represent the corresponding Fourier transformed signals [44]. In Eq. 2.2, $d \in \{1, ..., D\}$ is the channel, the bar means complex conjugation, and $\odot$ is the Hadamard (element-wise) product. Let y be l-th channel of an image patch with size of $M \times N \times D$, layer l of the correlation response map is given by

$$R_l = \mathcal{F}^{-1} \left(\sum_{d=1}^{D} C^d \odot \bar{Y}^d \right), \tag{2.3}$$

where $\mathcal{F}^{-1}$ represents the inverse Fourier transform and the size of R_l is $M \times N$. The position of maximum value of R_l represents the target location in layer l.

As Fig. 2.3 illustrates, correlation filter-based trackers attempt to determine the new position of the target by analyzing the displacement between the center of the correlation map, which corresponds to the previous target position, and the new peak in the map. More specifically, the correlation between the target model generated at previous frames and image features extracted from the current frame are used to determine the new target position.

2.1.3 Particle filters

The sequential importance sampling (SIS) algorithm is the basis of the particle filtering framework [59]. This technique implements a recursive Bayesian filter by sampling a set of N random particles $x_t^{(i)}$, $i = 1, \ldots, N$, calculating their weights, and estimating the posterior distribution based on these particles and weights. The higher the number of particles, the more accurate the model. In a particle filter, the particle weights are calculated by [59]

$$\omega_{x_t}^{(i)} \propto \omega_{x_{t-1}}^{(i)} \frac{p(y_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1})}{q(x_t^{(i)}|x_{t-1}, y_t)}, \tag{2.4}$$

where $p(x_t^{(i)}|x_{t-1})$ and $p(y_t|x_t^{(i)})$ are the transition and likelihood distributions, $q(x_t^{(i)}|x_{t-1}, y_t)$ is the proposal distribution used to sample the particles, and $\omega_{x_{t-1}}^{(i)}$ represents the previous weights. The posterior distribution is then approximated by

$$\hat{Pr}(x_t|y_t) \approx \sum_{i=1}^{N} \varpi_{x_t}^{(i)} \delta(x_t - x_t^{(i)}), \tag{2.5}$$

where $\varpi_t^{(i)}$ are the normalized particle weights. However, particle filters used in correlation trackers generally sample particles from the transition distribution, i.e., $q(x_t^{(i)}|x_{t-1}, y_t) = p(x_t^{(i)}|x_{t-1})$. These methods also re-sample particles at every frame, which removes the term corresponding to previous weights $\omega_{x_{t-1}}^{(i)}$ from Eq. 2.4. Finally, the weight of each particle in these trackers is given by [60]

$$\omega_{x_t}^{(i)} \propto p(y_t|x_t^{(i)}). \tag{2.6}$$

2.1.3.1 Sample degeneracy and impoverishment

The degeneracy phenomenon is a significant challenge in particle filtering. In this scenario, all but a few particles have negligible weight after a few iterations. Degeneracy in the particle filter model means spending substantial computational resources on updating particles that have negligible influence on the posterior distribution [59]. Resampling is the solution for degeneracy, and it consists of generating new particles with equal weights.

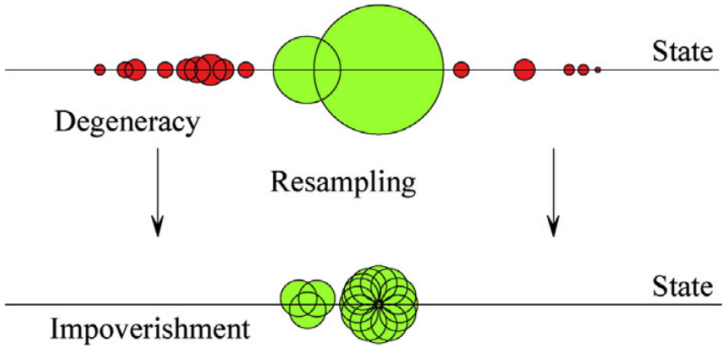


Figure 2.4: Degeneracy and impoverishment in particle filters [3].

However, a frequent resampling causes a decrease in particle diversity. This problem is referred to as sample impoverishment [3]. Fig. 2.4 illustrates the degeneracy and impoverishment problems in particle filters. Thus, resampling should be carried out only if it is strictly needed.

2.2 CNN visual trackers

The successful application of deep convolutional neural networks to object detection tasks [19, 20, 21, 22, 23, 24] has led to an increased interest in the utilization of such networks for visual tracking applications. Most CNN-based tracking algorithms use the CNN to examine image patches and determine the likelihood that a particular patch corresponds to the target. Li et al. [61] presented a tracker which samples image patches from the region surrounding the previous target position and uses multiple image cues such as hue, intensity, and gradient as inputs to a simple CNN. The network weights are updated at every frame by employing a structural loss cost function that decreases the importance of new image patches as their distance to the estimated target position increases. The authors later employed Bagging [62] to improve the robustness of their online network weight update process [63]. Vital, proposed by Song et al. [64], solves the problems of using deep classification networks in the tracking-by-detection framework, which consists

of two stages: 1) sampling around the target, 2) classifying each sample as the target object or as background. For the problem of overlapping positive samples, they integrate adversarial learning into their framework. Adversarial learning predicts discriminative features from different samples. The classifier is then able to focus on temporally robust features. Another successful strategy is to employ a CNN that generates a prediction map to evaluate the likelihood that the object is present in a larger search region [65]. This method considerably decreases the number of candidate patches to be compared with the target. The multi-domain network (MDNet) tracker [36] samples image patches at multiple positions and scales to account for target size variations. MDNet uses three convolutional layers to extract common target features and several domain- (or target-) specific fully connected layers to differentiate between a certain target category and the background. SANet [37] extends the MDNet architecture by employing a recurrent neural network (RNN) to predict the target position. However, much of the performance of MDNet and SANet is due to the fact that they are trained utilizing the benchmark datasets on which they are evaluated.

2.3 CNN-correlation visual trackers

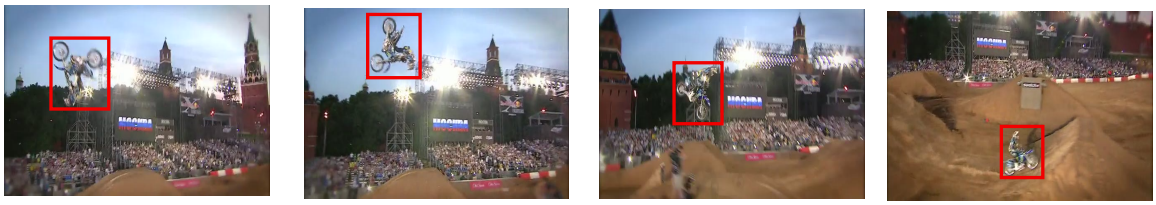
One effective mechanism to improve the performance of CNN trackers on unseen datasets is to apply correlation filters. This framework determines the similarity between an image patch and a target model using frequency domain, feature agnostic filters [38, 39, 40]. As a consequence, most state-of-the art CNN-based trackers integrate convolutional features and correlation filters [66]. By employing correlation filters on the hierarchical convolutional features generated by multiple layers of a deep CNN, HCFT [44] shows substantial performance improvement in comparison with other visual trackers. Later, Qi et al. [43] introduced HDT, a new CNN tracker based on the HCFT structure that employs a hedging algorithm to assign weights to the outputs of the convolutional layers. This is because different convolutional layers encode different levels of semantic and spatial information and their combination may improve tracking results [67]. Instead

of considering convolutional layers independently, Danelljan et al. proposed C-COT [68], which employs a continuous fusion method among multiple convolutional layers and uses a joint learning framework to leverage different spatial resolutions. The authors later addressed C-COT's problems of computational complexity and model over-fitting in the ECO algorithm [69]. Their factorized convolution operator and their novel model update method decrease the number of parameters in the model and improve tracking speed and robustness.

Recently, several methods have been proposed to improve the performance of correlation based trackers. One strategy entails combining different types of features and constructing multiple correlation-based experts [70]. Spatial-temporal information can also be used to address unwanted boundary effects in correlation trackers by spatially penalizing the filter coefficients [40]. To address the additional computational costs associated with such strategies, in contrast to methods that train the model using samples from the current and previous frames, the authors of [71] update the correlation model using samples from the current frame and the previously learned correlation filter. In [72], Zu et al. further extend such strategies through a spatial-temporal attention mechanism that uses optical flow information in consecutive frames. Finally, Sun et al. [73] use an approach based on reliability information [74], which performs real-time tracking by estimating the importance of sub-regions within the correlation filter. The reliability information highlights more reliable regions.

2.4 Particle filters in CNN-correlation visual Trackers

Particle filters provide an effective and general framework for improving the performance of CNN-correlation trackers [45, 75, 48, 76, 77]. All of the particle filters applied in CNN-correlation trackers use the transition distribution as the sampling (or proposal) distribution and calculate the weight of each particle based on the correlation response map obtained from the correlation filter and the CNN features. However, using particle filters



MotorRolling: "motorcycle rolling in a race"

Jogging-1: "Woman jogging and passing a traffic light"

Figure 2.5: Sample frames of the video sequences comprising the OTB50 and OTB100 benchmarks. The first row is *MotorRolling* data sequence included in OTB50 and OTB100. The second row is *Jogging-1* included only in OTB100.

in conjunction with correlation filters also introduces additional challenges. As shown in [45, 76], particle-correlation trackers use the sum of the elements of the correlation maps as the weights of the particles. However, in challenging situations, such as in the presence of occlusions or target deformations, the correlation maps are not reliable and generate weights that do not reflect the similarity between the target model and the image patch under consideration. Additionally, particle-correlation trackers generally estimate the target state based on the particle with the maximum weight [45, 76, 75], which is not always an accurate method because many particles may have similar weights. Furthermore, the aforementioned correlation-particle trackers perform resampling at every frame and consequently lose previous particle information. As shown in [78], iterative particle filters can improve sampling and lead to more distinctive particle likelihood models. However, such methods have been not used in conjunction with CNN-correlation trackers so far.

Bear-12: "white bear walking on grass around the river bank"

Bus-19: "red bus running on the highway"

Horse-1: "brown horse running on the ground"

Person-14: "boy in black suit dancing in front of people"

Figure 2.6: Sample frames of the video sequences comprising the LaSOT benchmark [4]

2.5 Benchmarks

In most computer vision problems, we need benchmarks to evaluate the performance of different methods and compare it with other works. There are three important benchmarks for visual tracking, which are described in the following subsections.

2.5.1 OTB50 and OTB 100

The OTB50 [79] and OTB100 [6] benchmarks contain 50 and 100 fully annotated video sequences, respectively. For each video sequence, the annotations correspond to the bounding boxes of one target of interest over all the video frames. Each of these data sequences is associated with one or more attributes. The dataset considers 11 different attributes covering the most common challenging scenarios observed in visual tracking.

These attributes include: illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, out-plane and in-plane rotations, out of view, background clutter and low resolution. What makes visual tracking particularly challenging is the fact that the target motion and appearance may change significantly under these conditions. Fig 2.5 shows examples of the video sequences contained in these benchmarks.

2.5.2 LaSOT

LaSOT is currently the largest publicly available benchmark for object tracking [4]. It provides high-quality dense manual annotations with 14 attributes representing challenging aspects of tracking. In comparison with OTB50 and OTB100, the additional attributes considered in this benchmark are partial occlusion, full occlusion, and viewpoint change. The benchmark consists of $1,400$ videos with an average of $2,512$ frames per sequence. Fig. 2.6 illustrates examples of this benchmark.

2.5.3 Evaluation metrics

Two of the most commonly used metrics to evaluate the performance of visual tracking algorithms are the precision plot and the success plot, which are illustrated in Fig. 2.7. Briefly, *Precision plots* correspond to the percentage of video frames for which the average Euclidean distance between the tracked locations and the ground truth is below a certain threshold while for the *Success plots*, the threshold is based on the area of overlap between the predicted bounding box and the respective ground truth [79].

Precision plot. This metric indicates how precise a tracker is. It represents the Euclidean distance between the target locations calculated by the tracker and centers of the manually labeled ground truth bounding boxes [79]. The precision plot is shown in Fig. 2.7. The location error threshold on the horizontal axis shows the magnitude of the threshold distances in comparison with the ground truth. The calculated precisions on the vertical axis

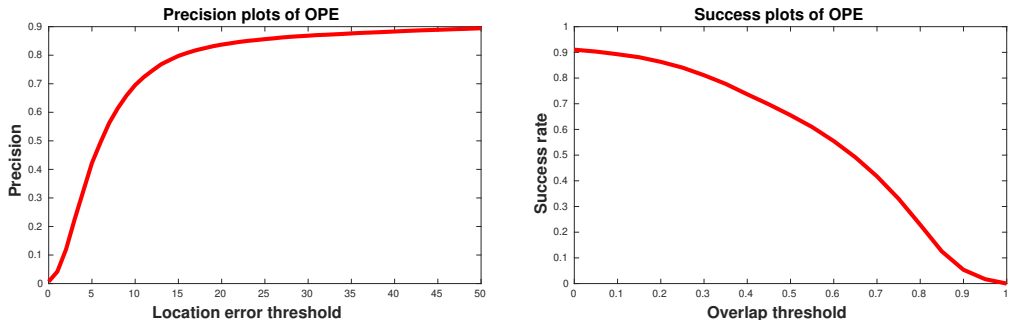


Figure 2.7: Left: the precision plot represents the precision in the calculation of target positions with respect to the location error threshold. Right: The success plot represents the precision in the calculation of target bounding box with respect to the overlap threshold.

represent the percentage of frames in which the estimated target locations are within these thresholds.

Success plot. This metric represents the overlap between the bounding boxes estimated by a tracker and the corresponding ground truth bounding boxes [79]. The success plot is illustrated in Fig. 2.7. The overlap threshold on the horizontal axis shows the ratio of overlap between the estimated target bounding box and the corresponding ground truth. The calculated precisions on the vertical axis represent the percentage of frames in which the target bounding box overlaps with the ground truth within these ratios. This overlap is computed using the Jaccard index (or intersection-over-union), which is given by

$$Success = \frac{BB_{Est} \cap BB_{GT}}{BB_{Est} \cup BB_{GT}}, \tag{2.7}$$

where BB_{Est} and BB_{GT} are the estimated bounding box and the ground truth bounding box, respectively, while $\cap$ and $\cup$ are the set intersection and union operators, respectively.

2.5.3.1 Evaluation strategies

The precision and success metrics are generally computed using the so called One-Pass Evaluation (OPE) strategy. In OPE, the ground truth for the first video frame is used to initialize the target position, which is then tracked throughout the video sequence without

intermediate re-initialization steps. In the alternative Spatial Robustness Evaluation (SRE), the initialization is subject to some disturbance. Temporal Robustness Evaluation (TRE) focuses on short-term tracking [79].

CHAPTER 3
DEEP CONVOLUTIONAL PARTICLE FILTER

This chapter proposes a novel framework for visual tracking based on the integration of a deep convolutional neural network (CNN), a correlation filter and a particle filter [45]. In the proposed framework, the position of the target at each frame is predicted by a particle filter according to a motion model. The motion model in conjunction with the particle filter's ability to sample several image patches allow it to overcome temporary target losses caused by dramatic temporary appearance changes or occlusions. Particles around the predicted position are then used as input to the HCFT CNN-based tracker [44] which adjusts their positions to the most likely target positions. Our framework utilizes the output of the convolutional neural network and correlation filter, which we henceforth call a feature map, to determine the weights of the particles. Finally, the particles and their weights are used to calculate the target's position in the current frame. We evaluated the performance of the proposed framework using the OTB50 benchmark dataset [79]. Our results show that this method improves the performance of HCFT, especially in challenging scenarios, such as those involving target deformation, illumination changes, out-of-plane and in-plane rotations.

3.1 Structure of HCFT

The CNN used in HCFT was originally proposed by Simonyan et al. in [10] for object detection. It includes five convolutional layers and five pooling layers. The outputs of the different layers of HCFT for a specific frame of the *motor-rolling* data sequence are illustrated in Fig. 3.1. The deconvolutional neural network proposed by Zeiler et al. in [80] clarified that layers 1 and 2 tend to respond to image edges, layer 3 recognizes similar textures, layer 4 illustrates significant variation, and layer 5 shows entire objects with significant pose variation. Thus, semantic information of the target is extracted from

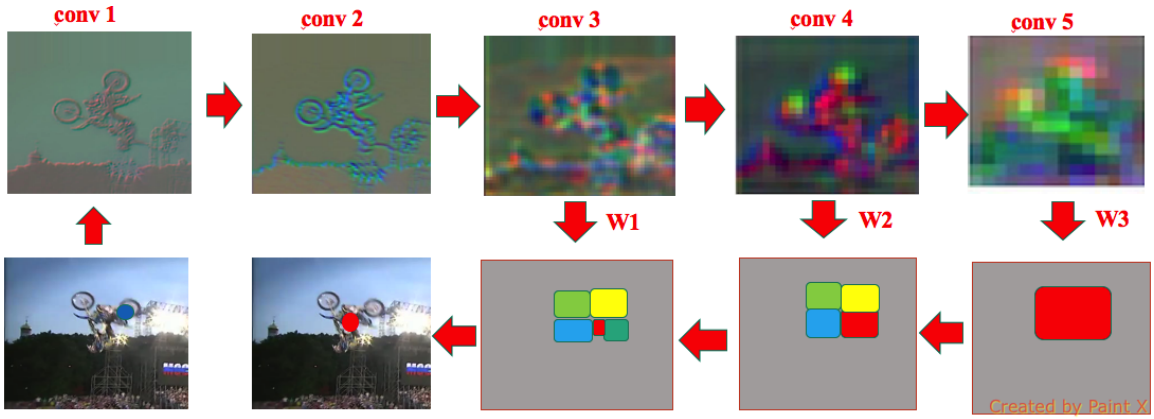


Figure 3.1: The outputs of the different layers of the CNN and the determination of the exact position of the target by applying a coarse-to-fine method. $W1$, $W2$ and $W3$ refer to the correlation filters. The red boxes are the estimated location of the target at each layer. The yellow, green and blue boxes show the areas from the search at each layer. Blue and red circles show the previous and current positions, respectively.

the later layers of the CNN and spatial details are obtained from the early layers. HCFT applies a correlation filter to the output of each convolutional layer. Then, by applying a coarse-to-fine localization method, it moves back from the fifth layer to the third layer to determine the exact position of the target. Fig. 3.1 also illustrates the coarse-to-fine method proposed in [44].

Algorithm 3.1 Proposed Visual Tracking Algorithm

Input: Current frame, previous target state z_{t-1}
Output: Current position and velocity of the target z_t

1: **repeat**
2: Predict $\hat{x}_t$ based on Eqs. 3.3 and 3.4
3: Generate initial particles $x_t^{(i)}$ around $\hat{x}_t$ based on Eq. 3.5
4: Give the patch corresponding to $x_t^{(i)}$ to the CNN and compute its correlation map
5: Extract the weight of each particle $\omega^{(i)}$ from its correlation map
6: Compute the normalized weights $\varpi_t^{(i)}$
7: Estimate x_t based on 3.7
8: **until** end of the video sequence

3.2 Particle Filter Design

Let the target state be defined as

$$z_t = [x_t, \dot{x}_t]^T, \tag{3.1}$$

where x_t represents the location of the target in the frame on the horizontal and vertical image axes

$$x_t = \left[u_t, v_t, \right]^T \tag{3.2}$$

and $\dot{x}_t$ is the velocity of x_t. We apply a first-order motion model to z_{t-1} according to

$$\hat{z}_t = A z_{t-1}, \tag{3.3}$$

where $\hat{z}_t$ represents the predicted target state for frame t and A is the process matrix defined by

$$A = \left[\begin{array}{c|c} I_2 & I_2 \\ \hline 0_{(2,2)} & I_2 \end{array}\right]. \tag{3.4}$$

The predicted target position is then disturbed by adding samples $\eta^{(i)} \in \mathbb{R}^8$ drawn from a zero-mean normal distribution to generate an initial set of particles $x_t^{(i)}$ according to

$$x_t^{(i)} = \hat{x}_t + \eta^{(i)}, \tag{3.5}$$

These particles are then used as inputs to the VGG network [10] to generate their convolutional features. These features are then fed to the correlation filter proposed in [44] to calculate the response maps corresponding to the particles. As shown in Eq. 2.6, the weight, $\omega^{(i)}$, of each particle is equal to the particle's likelihood which is calculated by

$$\omega^{(i)} = \sum_{m=1}^{M} \sum_{q=1}^{Q} R_{(m,q)}^{(i)}, \tag{3.6}$$

where $R_{(m,q)}^{(i)}$ is computed using Eq. 2.3 for each particle i. The intuition behind this choice is that feature maps that correspond to the target tend to show substantially higher

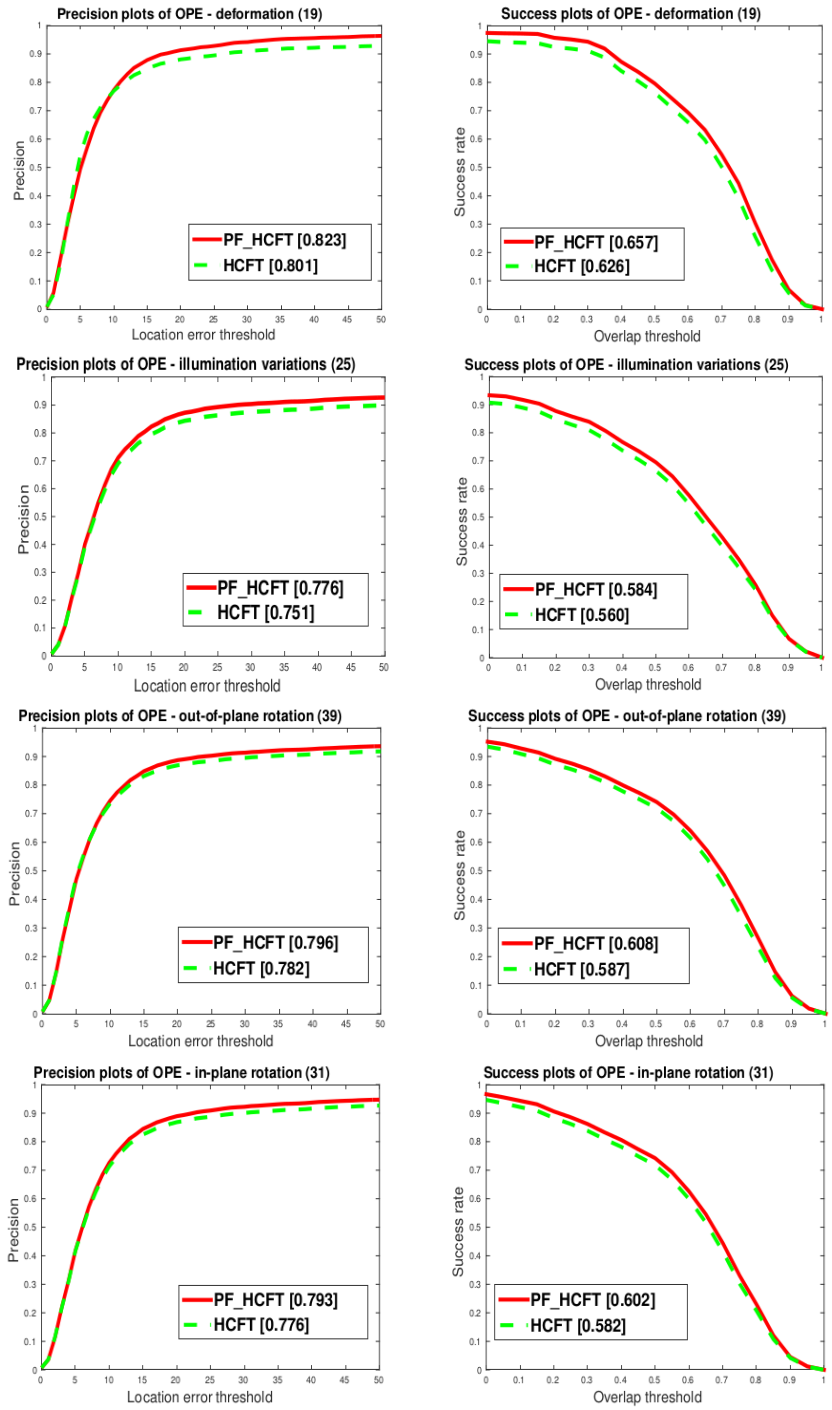


Figure 3.2: Performance comparison of our tracker versus HCFT on OPE. The red plots correspond to our tracker and the green plots to the baseline (HCFT).

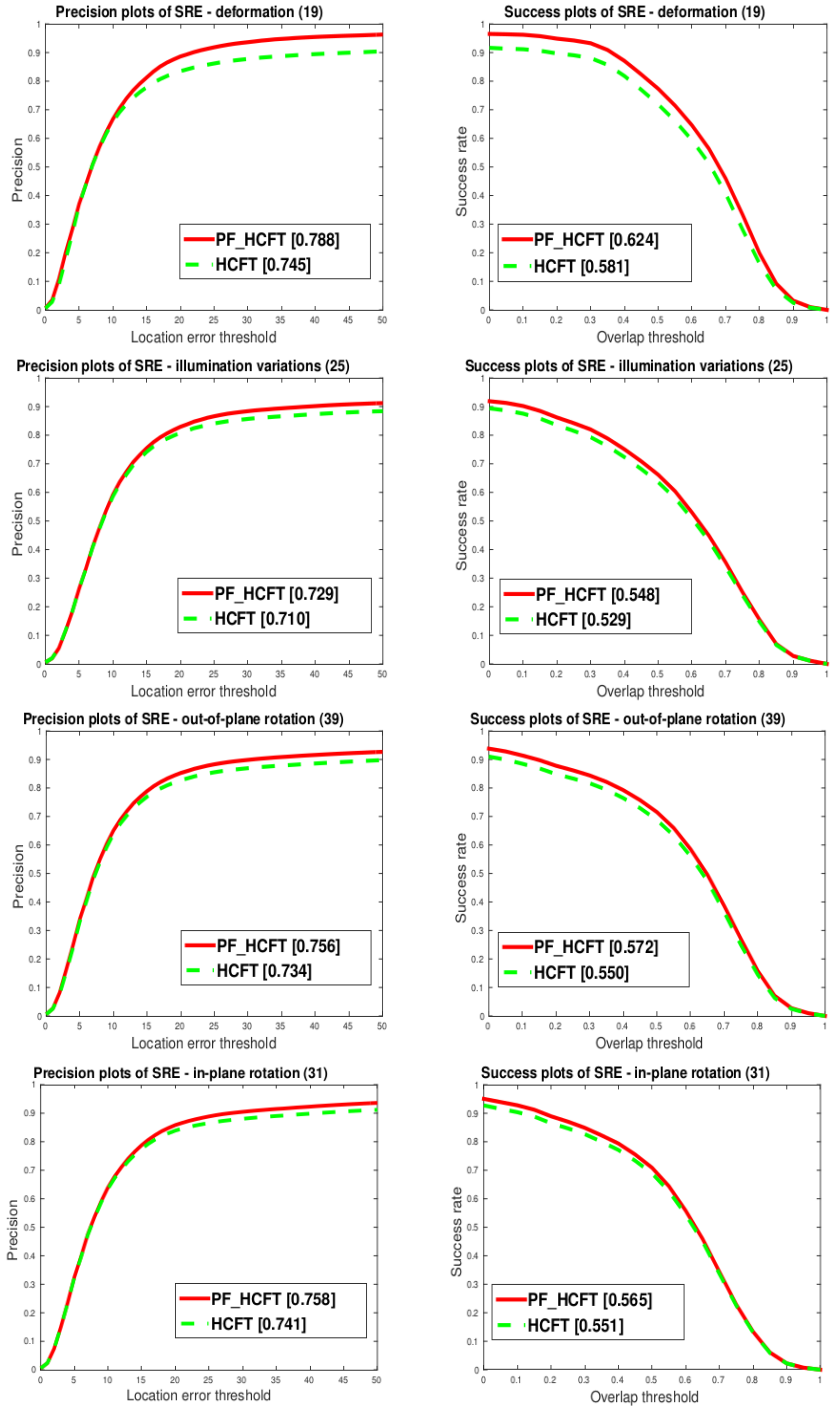


Figure 3.3: Performance comparison of our tracker versus HCFT on SRE metrics. The red plots correspond to our tracker and the green plots to the baseline (HCFT).

Figure 3.4: Comparison between our tracker and HCFT on six different data sequences. The performance of HCFT is shown in green and ours in red. The three test sequences on the left show OPE results and the three sequences on the right show SRE results.

correlation values than background patches. In the next step, each particle is shifted to the peak of its correlation response map as shown in Fig. 2.3. Finally, the position of the target in the current frame is estimated according to

$$x_t \approx \sum_{i=1}^{N} \varpi_t^{(i)} \tilde{x}_t^{(i)}, \tag{3.7}$$

where $\tilde{x}_t^{(i)}$ represents the shifted location and $\varpi_t^{(i)}$ is the normalized weight. Algorithm 3.1 summarizes the proposed visual tracker.

3.3 Results and Discussion

We evaluate our algorithm using the well known visual tracking benchmark OTB50 [79]. Fig.S 3.2 and 3.2 shows a quantitative evaluation of the performance of the proposed approach on OPE and SRE in comparison with HCFT for the attributes in which our approach shows the most significant improvement. As the figures shows, the proposed framework improves the performance of HCFT on several attributes. In attributes such as temporary deformation, illumination, in-plane and out-of-plane rotations in which the correlation filter loses track of the target, the motion model allows the successful prediction of

the position of the target and the particles are then able to recover using the weights generated by the CNN. Under challenging conditions that include deformation, illumination variation, out-of-plane and in-plane rotations, our method shows improvements of approximately 7.5%, 4.5%, 4% and 3.5%, respectively. The overall OPE success rate improvement is approximately 3.5%.

Fig. 3.4 shows a qualitative illustration of the performance of our tracker in comparison with HCFT on some sequences in which HCFT fails. The three sequences on the left show OPE results and the three sequences on the right illustrate SRE results. As the sequences indicate, the baseline tracker gets easily confused in situations such as deformation, occlusion, motion blur, and out-of-plane rotations. The particle filter is able to sample several image patches and it is hence capable of overcoming these difficulties.

CHAPTER 4
TARGET SIZE ESTIMATION AND ADAPTIVE CORRELATION MAPS

In this chapter, we propose a novel framework named DCPF2 for visual tracking [48]. The most important aspect of visual target tracking is to accurately determine the target position as well as its size. Despite the substantial performance gains obtained in recent years by the aforementioned CNN-correlation methods, their main disadvantage is their inability to vary the size of the target bounding box [44, 43]. We extend the particle filter described in the previous chapter (DCPF) to estimate the target size as well as the target position. Briefly, DCPF uses multiple particles as inputs to the VGG deep convolutional neural network [10]. For each particle, it then applies the correlation filter used in HCFT [44] on the extracted hierarchical convolutional features to construct the correlation map. The target position at the current frame is calculated based on the response of the correlation maps. However, similar to HCFT, DCPF tracks a bounding box of fixed size.

Another limitation of trackers based on conventional correlation filters is the fact that they generate only one target model. Thus, errors in calculating the final target state cause the target model to be incorrectly updated. In this chapter, we employ a new adaptive correlation filter to account for potential errors in the model generation. Thus, instead of generating one model which is highly dependent on the estimated target position and size, we generate a variable number of target models based on high likelihood particles. In frames where the target can be easily tracked, this number is low because the best particle has a high likelihood and fewer particles have similar likelihoods. Conversely, in challenging situations, the target is less similar to the model and hence the likelihood of most particles decreases and the particle weight distribution becomes less centralized. Experimental results on OTB50 [79] demonstrate that our proposed framework significantly outperforms state-of-the-art methods.

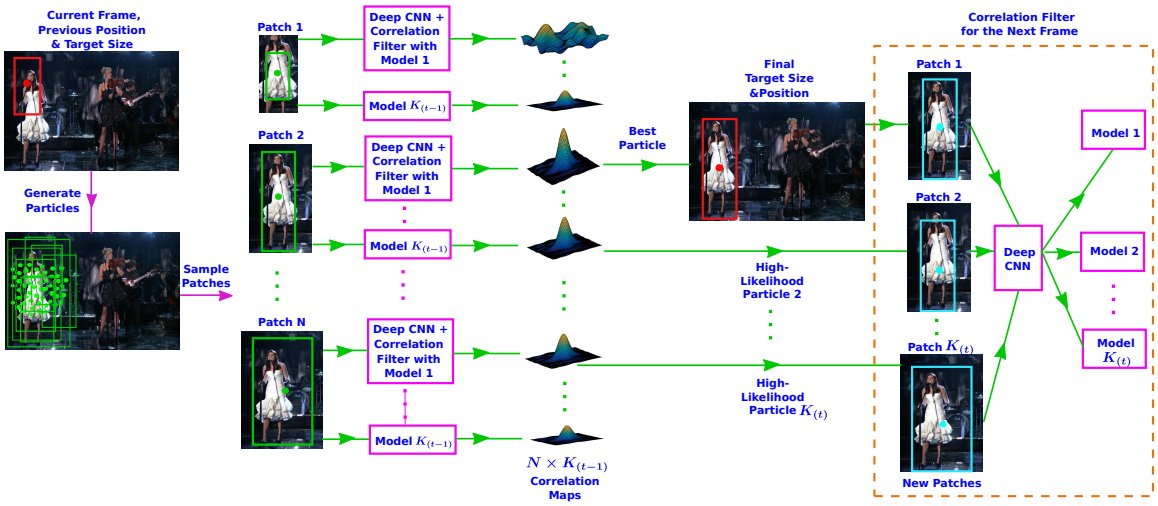


Figure 4.1: Overview of our proposed tracker which estimates target size and employs an adaptive correlation filter.

4.1 Proposed Algorithm

In this section, we first explain how our particle filter estimates the target size and its position. Our adaptive correlation filter is then discussed. Fig. 4.1 illustrates the proposed approach.

4.1.1 Particle Filter to Estimate the Target Bounding Box

Let the target state be

$$z_t = \left[x_t, \dot{x}_t\right]^T, \tag{4.1}$$

where

$$x_t = \left[u_t, v_t, h_t, w_t\right]^T \tag{4.2}$$

and $\dot{x}_t$ is the velocity of x_t. u_t and v_t are the locations of the target on the horizontal and vertical image axes at frame t, and h_t and w_t are its width and height. The tracker employs a linear motion model to predict the current state of the target $\hat{z}_t$ based on the previous

target state z_{t-1}. The predicted target state is given by

$$\hat{z}_t = Az_{t-1}, \tag{4.3}$$

where A is a standard constant velocity process matrix defined by

$$A = \left[\begin{array}{c|c} I_4 & I_4 \\ \hline 0_{(4,4)} & I_4 \end{array}\right]. \tag{4.4}$$

Particles $x^{(i)}$ are then generated by Eq. 3.5. In order to limit the number of particles needed, rather than drawing $\eta^{(i)}$ directly from an eight-dimensional distribution, we draw its samples individually, and change their height and width simultaneously using the same sample (i.e., we change the target scale but not its aspect ratio).

In the next step, $z^{(i)}$ are used to sample different patches from the video frame at time t. Each patch is then fed into VGG to calculate its convolutional feature map. Let $f_{l,d}^{(i)} \in \mathbb{R}^{M\times Q}$ be the convolutional feature map, where M, Q are the width and height of the map, l is the convolutional layer and d is the number of the channels for that layer $d = 1, ..., D$. Then, its correlation response map $R_l^{(j)(i)} \in \mathbb{R}^{M\times Q}$ is given by [44]

$$R_l^{(j)(i)} = \mathfrak{F}^{-1}\left(\sum_{d=1}^{D} C_{l,d}^{(j)} \odot \bar{F}_{l,d}^{(i)}\right), \tag{4.5}$$

where $\bar{F}_{l,d}^{(i)}$ is the complex conjugate Fourier transform of $f_{l,d}^{(i)}$, $\mathfrak{F}^{-1}$ represents the inverse Fourier transform, $j = 1, ..., K_{t-1}$ illustrates the number of models generated in the previous frame $t-1$, $C_{l,d}^{(j)}$ represents channel d of layer l of the correlation filter of the model j, the bar represents complex conjugation and $\odot$ is the Hadamard product. The final correlation response map $R^{(j)(i)}$ for particle i and model j is calculated based on a weighted sum of the maps for all the CNN layers [44]

$$R^{(j)(i)} = \sum_{l=1}^{L} \Upsilon_l(R_l^{(j)(i)}), \tag{4.6}$$

where Υ_l is a regularization term for each layer term [44]. The likelihood or weight of each correlation response map is calculated by

$$\omega^{(j)(i)} = \sum_{m=1}^{M} \sum_{q=1}^{Q} R_{(m,q)}^{(j)(i)}, \tag{4.7}$$

where $R_{(m,q)}^{(j)(i)}$ refers to the element of the final correlation response map on row m and column q. In total, we have $N \times K_{t-1}$ weights. Unlike the method discussed in Chapter 3, here we use the highest likelihood particle to estimate the target state. Thus, we find the maximum weight ω_{max} over all the particles and models

$$\omega^* = \max_{j,i} \omega^{(i,j)}. \tag{4.8}$$

Let the indexes corresponding to ω^* be $i = i^*$ (the best particle) and $j = j^*$ (the best model). Then, the final target size is given by $h^{(i^*)}$ and $w^{(i^*)}$. That is, the i^*-th patch with dimensions $h^{(i^*)}$ and $w^{(i^*)}$ is the most similar to the best model $C^{(j^*)}$. Additionally, let $R^{*(j^*)(i^*)}$ be the correlation response map associated with ω^*, its peak is located at

$$[\delta_u, \delta_v] = \arg\max_{m,q} R_{(m,q)}^{*(j^*)(i^*)}, \tag{4.9}$$

where $m = 1, ..., M$ and $q = 1, ..., Q$. The final target position is then calculated by shifting the best particle towards the peak of its correlation map

$$[\tilde{u}, \tilde{v}] = [u^{(i^*)} + \delta_u, v^{(i^*)} + \delta_v], \tag{4.10}$$

where $u^{(i^*)}$ and $v^{(i^*)}$ correspond to the position of the best particle. Thus, the target state at the frame t is

$$z_t = \left[x^*, \dot{x}^{(i^*)} \right]^T, \tag{4.11}$$

where $\dot{x}^{(i^*)}$ is the velocity of the best particle and

$$x^* = \left[\tilde{u}, \tilde{v}, h^{(i^*)}, w^{(i^*)} \right]^T. \tag{4.12}$$

Algorithm 4.1 summarizes our method to estimate the target state.

Algorithm 4.1 Calculate the current target state

Input: Current frame, previous target state z_{t-1}, correlation filters $C^{(j)}$, $j = 1, \ldots, K_{t-1}$ generated in the previous frame

Output: Current target state z_t, maximum weight ω^*, N particles $x^{(i)}$, $N \times K_{t-1}$ correlation response maps $R^{(j)(i)}$ and their weights $\omega^{(j)(i)}$

```
Generate N particles around the predicted target state x̂_t according to Eqs. 4.3 to 4.4
for Each particle x^(i) do
    for Each of the K_{t-1} correlation filters C^(j) do
        Generate the K_{t-1} correlation response maps R^(j)(i) according to Eq. 4.5
        Compute the weight ω^(j)(i) based on Eq. 4.7
    end for
end for
Determine the best particle using Eq. 4.8
Update the target state z_t according to Eqs. 4.9 to 4.12
```

4.1.2 Adaptive Correlation Filter

After finding ω_{max}, we examine the following relationship for all $N \times K_{t-1}$ weights

$$\omega^{(j)(i)} > \alpha\omega^*, \tag{4.13}$$

where α is a constant. If Eq. 4.13 is true, the corresponding particle is considered a high likelihood particle. Let i' and j' be the indices of the selected particle weights. Then, $h^{(i')}$ and $w^{(i')}$ calculated by Eq. 3.5 are considered the target size. Additionally, the correlation response map $R^{*(j')(i')}$ is used to calculate the estimated target position $\tilde{u}^{i'}$ and $\tilde{v}^{i'}$ similar to Eq. 4.9 and Eq. 4.10. Thus the corresponding high likelihood particle $z_{high}^{(s)}$ is given by

$$z_{high}^{(s)} = \left[\tilde{u}^{i'}, \tilde{v}^{i'}, h^{(i')}, w^{(i')}\right]^T, \tag{4.14}$$

where $s = 1, ..., K_t$ and K_t is the number of the high-likelihood particles. We then generate a patch from frame t for each of the K_t high-likelihood particles. In the next step, these patches are fed into the CNN to extract K_t convolutional feature maps, and a new correlation filter $C_{l,d}^{(s)}$ is then generated for each of the K_t high likelihood particles. The generated models are used in frame $t + 1$ to be compared with the convolutional features generated by each particle.

Algorithm 4.2 Adaptive Correlation Filter

Input: Current frame, maximum weight ω^*, N particles $x^{(i)}$, their $N \times K_{t-1}$ correlation response maps $R^{(j)(i)}$ and weights $\omega^{(j)(i)}$

Output: Correlation filters $C^{(j)}$, $j = 1, \ldots, K_t$ to be used in the next frame

1: **for** Each weight ω_i^j **do**
2: **if** Eq. 4.13 is true **then**
3: Generate a high-likelihood particle $z_{high}^{(s)}$ according to Eq. 4.14
4: **end if**
5: **end for**
6: **for** Each particle $z_{high}^{(s)}$ **do**
7: Calculate its correlation filter $C_{l,d}^{(s)}$
8: **end for**

Algorithm 4.2 summarizes our adaptive correlation filter procedure. The comparison between the best model with the most accurate target size and position generates more accurate correlation maps. As previously mentioned, by varying the number of models K_t with the number of high likelihood particles, we are able to maintain a larger number of tentative models in challenging scenarios such as in the presence of illumination variation, motion blur, or partial occlusion due to the wider distribution of the particle weights under these conditions.

4.2 Results and Discussion

We evaluate our tracker on the OTB50 dataset. We heuristically set $\alpha = 0.8$ and $N = 300$. Fig. 4.2 provides a quantitative evaluation of our proposed approach in comparison with 11 state-of-the-art trackers [44, 43, 81, 82, 83, 84, 83, 85, 86, 87, 88]. In attributes such as scale variation, illumination and out-of-plane rotations where the common correlation filter loses track of the target, our adaptive correlation filter in conjunction with the particle filter are then able to recover using the weights generated by the CNN. For challenging scenarios of scale variation, illumination variation and out-of-plane rotation as

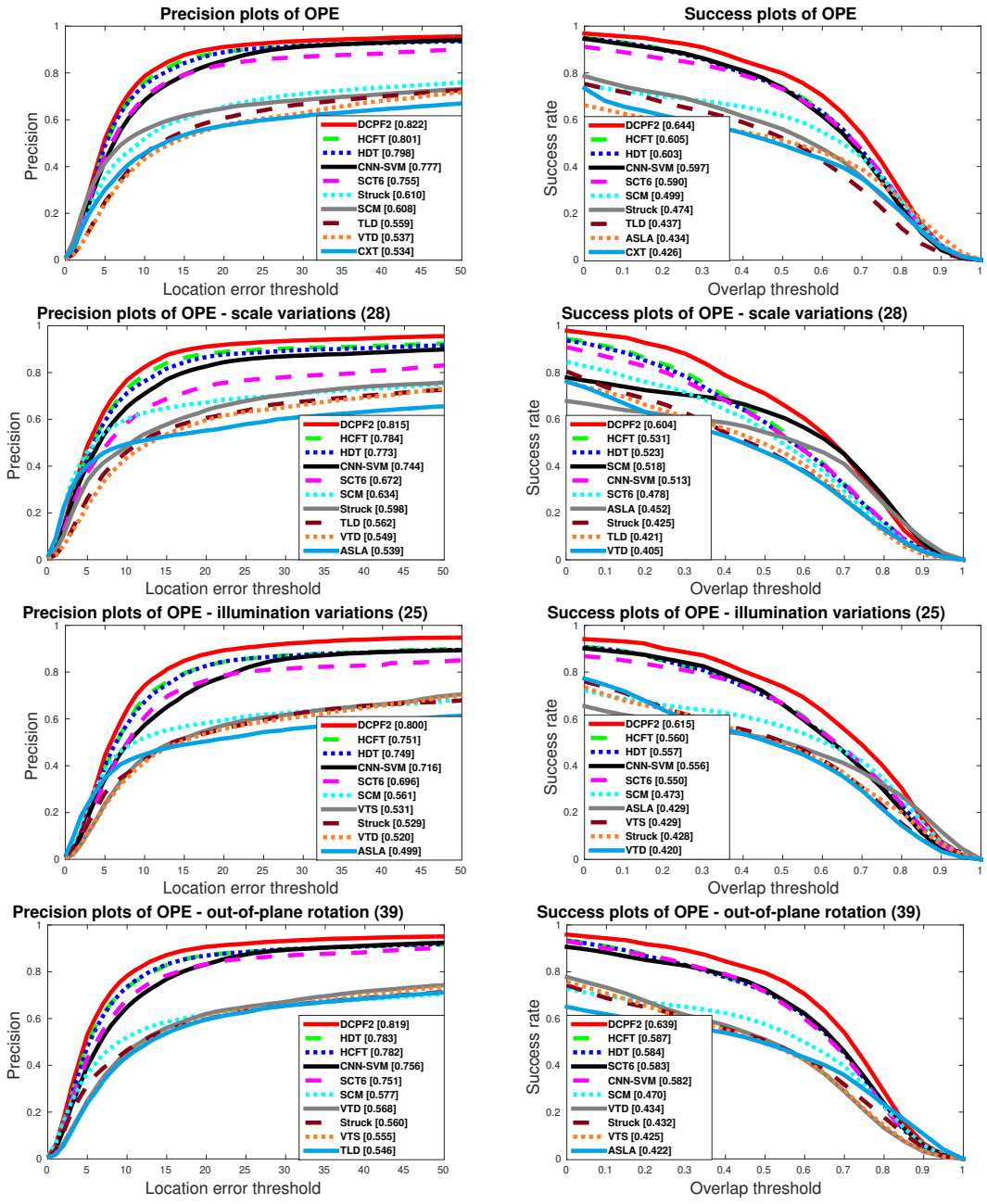


Figure 4.2: Quantitative evaluation of our tracker and fourteen state-of-the-art trackers on OPE.

well in terms of overall performance, our tracker shows improvements of approximately 14%, 10%, 9% and 7%, respectively, in comparison with the second best tracker HCFT.

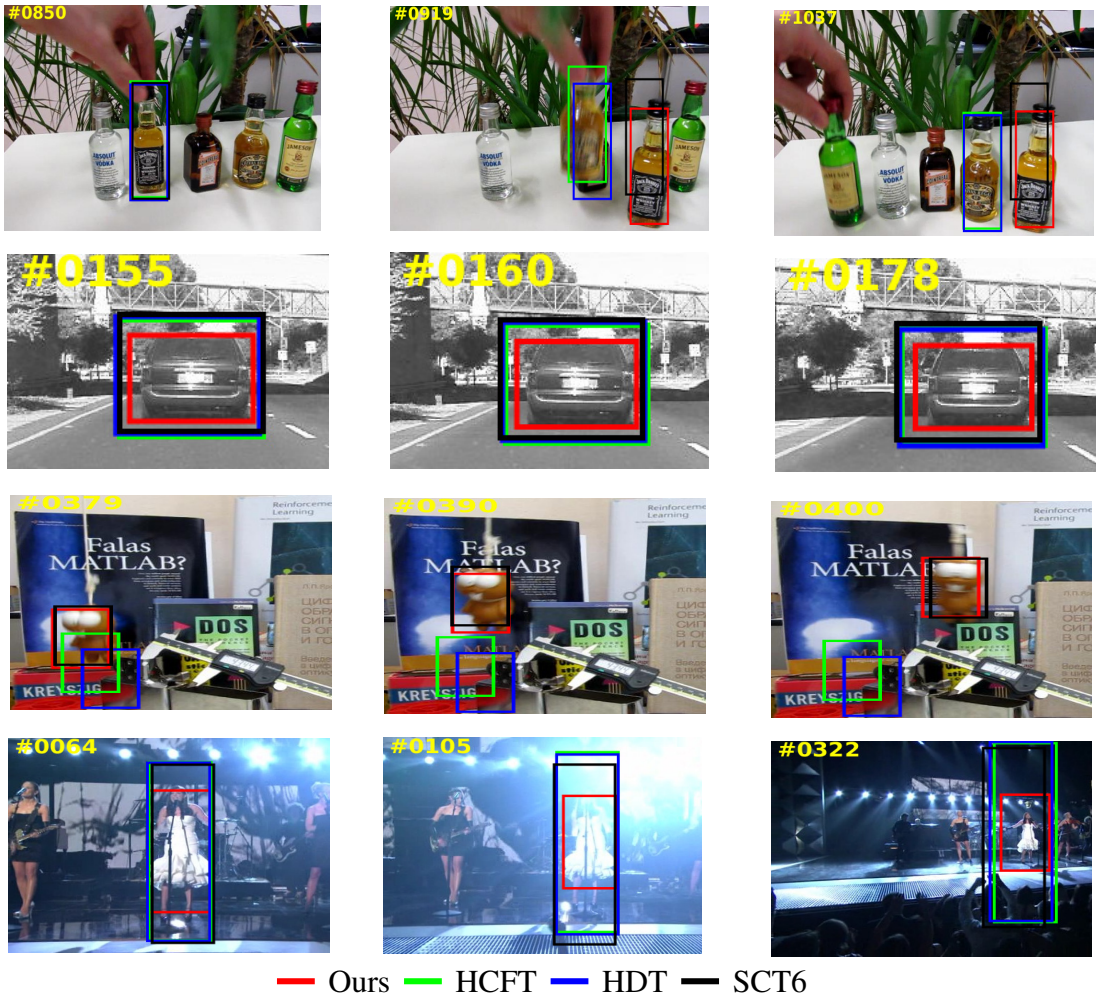


Figure 4.3: Qualitative evaluation of our tracker, *HCFT*, *HDT* and *SCT6* on six challenging sequences (from left to right and top to bottom are *Liquor*, *Car4*, *Lemming* and *Singer1*, respectively).

Fig. 4.3 qualitatively illustrates the performance of our tracker in comparison with three trackers: the CNN-based trackers HCFT and HDT as well as the correlation filter tracker SCT6 [82]. As the results in Fig. 4.3 indicate, the baseline trackers get easily confused in situations such as scale variation, illumination variation, or in-plane and out-of-plane rotations. The proposed particle-correlation filter is able to sample several image patches and it is hence capable of overcoming these difficulties.

CHAPTER 5
ADAPTIVE PARTICLES FILTER FOR VISUAL TRACKING

In this chapter, we describe a visual tracker named Deep Convolutional Adaptive Particle Filter with Multiple Correlation Models (CAP-mc) [50], in which we replace the particle filter proposed in DCPF2 with an adaptive particle filter. Although particle filters improve the performance of CNN-correlation trackers, especially in challenging scenarios such as those involving target occlusion and deformation, they considerably increase the computational cost. Adaptive particle filters can improve the results of object tracking [89] while decreasing the computational costs. However, they have not been used in conjunction with CNNs and correlation filters yet. Our adaptive particle filter decreases the number of particles and the computation cost in simple frames in which there is no challenging scenario and the target model closely reflects the current appearance of the target. In such scenarios, many particles may converge to the same location and the number of particles is allowed to decrease in these frames. Additionally, our adaptive particle filter can refine the particles' locations to be used in the next frame. This method is more reliable than sampling new particles in every frame which was employed in DCPF2. We use the weight calculated in the first frame as one of the resampling criteria because it is calculated by comparing with the ground truth target model. Another threshold for resampling is the number of particles. We perform resampling only when the number of particles or the weight of the selected particle is too small.

Additionally, we realized that the model update rate is a critical parameter in correlation-base trackers. The adaptive correlation filter proposed in DCPF2 generates several target models based on all the high-likelihood particles to cover probable errors instead of generating one model based on the selected particle. Our new tracker applies different model update rates to generate several target models for each high-likelihood particle. Thus, we create multiple models; some are less affected by the previous model and are useful in

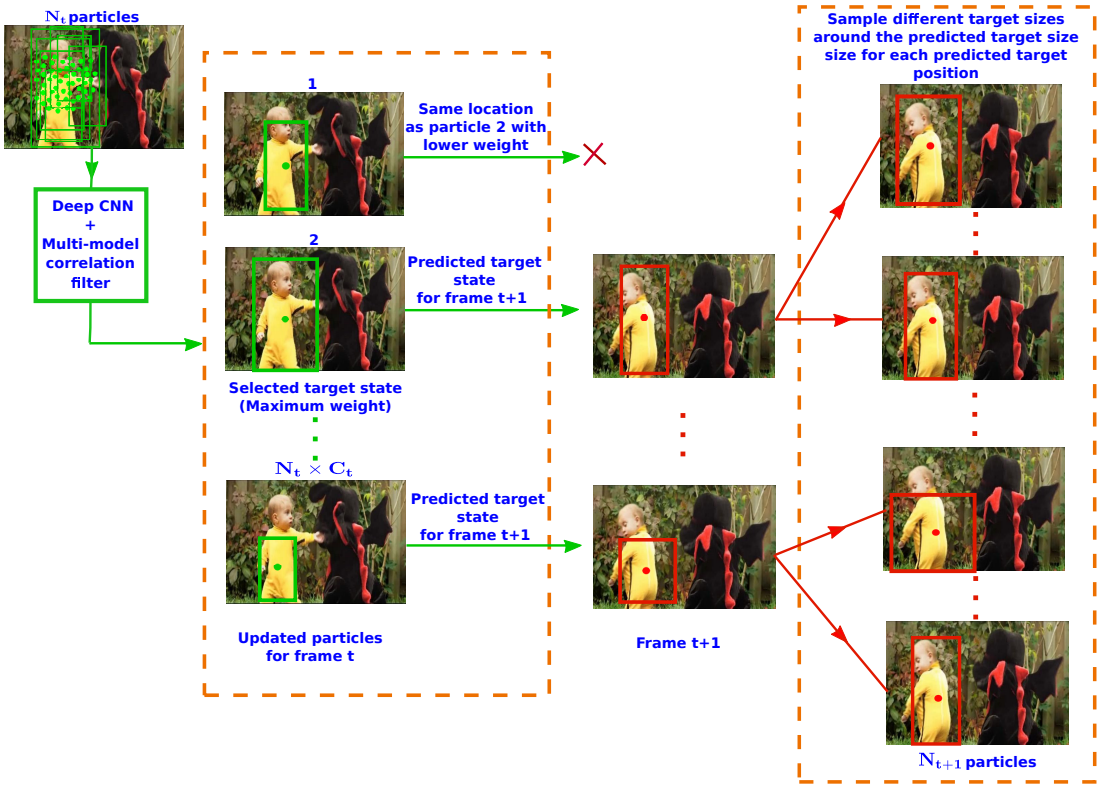


Figure 5.1: Our proposed adaptive particle filter (when resampling is not needed).

simple frames, while other models are affected by the previous model, and are hence suitable for challenging frames. We tested our tracker on OTB100 [6], and the results show substantial performance improvements over state-of-the-art methods.

5.1 Proposed Algorithm

In this section, we present our adaptive particle filter illustrated in Fig. 5.1. We then discuss our new adaptive correlation filter based on variable model update rates.

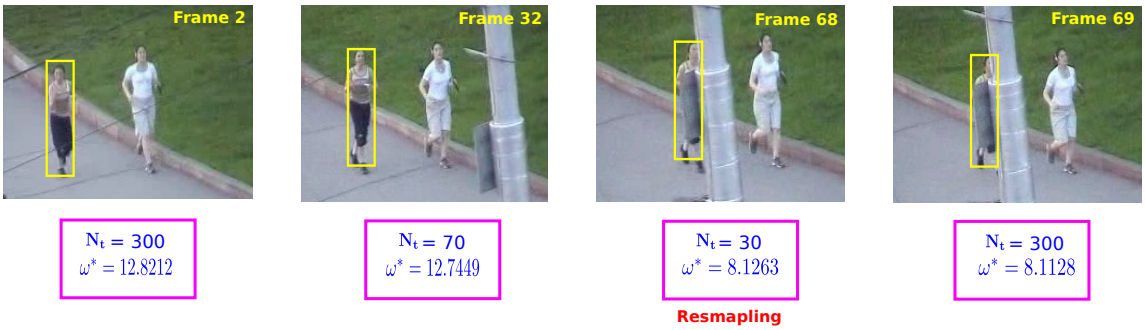


Figure 5.2: Decreasing the number of particles in simple frames and implementing resampling in difficult frames.

5.1.1 Adaptive Particle Filter

Algorithm 5.1 summarizes our adaptive particle filter. Let z_1 be the ground truth target position and size in the first frame

$$x_1 = \left[u_1, v_1, h_1, w_1\right]^T, \tag{5.1}$$

where u_1 and v_1 are the ground truth locations of the target and h_1 and w_1 are its ground truth width and height. The initial target state is defined by

$$z_1 = \left[x_1, \dot{x}_1\right]^T, \tag{5.2}$$

where $\dot{z}_1$ is the velocity of z_1, which is assumed to be zero in the first frame. After extracting a patch from the first frame based on z_1, we provide this patch to a CNN [10] to calculate its convolutional features. The ground truth target model is then generated by computing the Fourier transform of the convolutional features as explained in Section 2.1.2. The ground truth target model is used to calculate a threshold for the resampling process as explained later in this chapter. Additionally, this model is updated during the next frames as discussed in the next section.

In the subsequent frames (i.e., $t > 1$), we generate and evaluate the initial particles as explained in Algorithm 5.2. Considering z_{t-1} as the previous target state, the predicted

Algorithm 5.1 Adaptive particle filter

Input: Current frame, previous target state z_{t-1} and C_{t-1} target models $\mho_{t-1}^{(j)}$
Output: Current target state z_t, particles $x_{t+1}^{(i)}$ for the next frame

```
Generate initial particles to find the final target state z_t according to Algorithm 5.2
if t = 1 then
    Calculate T_w
end if
Update particles and remove redundant ones using Algorithm 5.3
Examine the resampling conditions according to Eq. 5.10 and Eq. 5.11
if resampling is needed then
    Generate particles x_{t+1}^{(i)} for the next frame based on Algorithm 5.2
else
    Calculate the predicted states x̂_{(t+1)}^{(p)} for frame t + 1 using Eq. 5.12
    for each x̂_{(t+1)}^{(p)} do
        Generate β samples of the target size
        Calculate the particles for the next frame t + 1 according to Eq. 5.13
    end for
end if
```

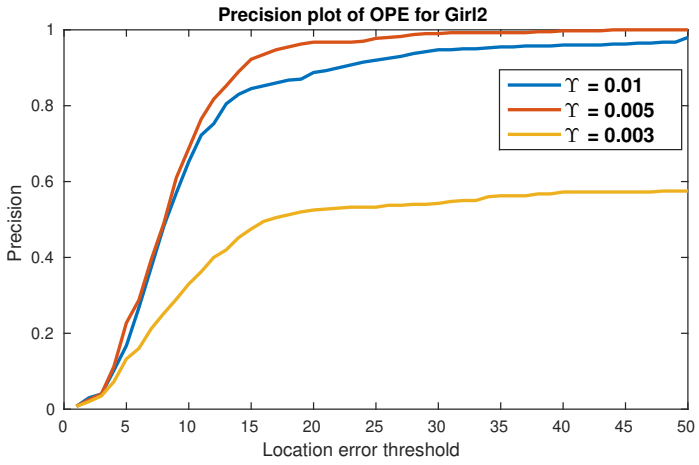


Figure 5.3: How to influence different adjusting rates on DCPF2's performance and giving the idea to generate multiple target models based on applying different adjusting rates in CAP-mc.

target state $\hat{x}_t$ is calculated by Eqs. 4.3 and 4.4. It is clear that $\hat{x}_2 = x_1$ because $\dot{z}_1 = 0$. Let

Algorithm 5.2 Generate and evaluate initial particles

Input: Current frame, previous target state z_{t-1} and C_{t-1} target models $\mho_{t-1}^{(j)}$

Output: Current target state z_t, N_t particles $x_t^{(i)}$, their correlation response maps $R^{(i)(j)}$, their weights $\omega^{(i)(j)}$, maximum weight ω^* and the best target model $\mho_{t-1}^*$

```
Calculate the predicted target state ẑ_t according to Eq. 4.3 and Eq. 4.4
Generate N_t particles x_t^(i) around the predicted target state according to Eq. 3.5
for each particle x_t^(i) do
    for each of the C_{t-1} target models Ʊ_{t-1}^(j) do
        Generate the correlation response map R^(i)(j)
        Calculate its weight ω^(i)(j) according to Eq. 4.7
    end for
end for
Find the maximum weight ω* based on Eq. 5.3
Consider the particle corresponding to ω* as the final target state z_t
Consider the target model corresponding to ω* as the best model Ʊ*_{t-1}
```

$x_t^{(i)}$ represent the particles sampled by Eq. 3.5, where $i = 1, ..., N_t$ and N_t is the number of the particles.

In the next step, different patches from frame t are generated based on $z_t^{(i)}$. For each patch, a convolutional feature map is calculated using the CNN. Let $R^{(i)(j)} \in \mathbb{R}^{M \times Q}$ be the final correlation response map for particle i and target model $j, j = 1, ..., C_{t-1}$ (the generation of different target models in the previous frame is explained in the next section). M and Q are the length and width of the final correlation response map. These correlation response maps are computed by comparing the target models and the convolutional feature maps [44]. For each correlation response map, the weight is calculated by Eq. 4.7.

In the second frame, after comparing the convolutional features with the ground truth model , we save the weight calculated from the correlation response map as a threshold T_w which is a reliable representative of the target because it is calculated based on the ground truth model. The location of the particle with the maximum weight [48]

$$[i^*, j^*] = \arg\max_{i,j} \omega^{(i)(j)}. \tag{5.3}$$

Algorithm 5.3 Update particles and remove redundant ones

Input: N_t particles $x_t^{(i)}$, their correlation response maps $R^{(i)(j)}$, their weights $\omega^{(i)(j)}$
Output: Remaining updated particle $\bar{x}_t^{(i)(j)}$

```
for each R^(i)(j) do
    Calculate its peak according to Eq. 5.4
    Update its x_t^(i) to find x̄_t^(i)(j) using Eq. 5.5 to Eq. 5.7
end for
for every two particles do
    if Eq. 5.8 is correct that means two particles converge the same target
          position then
        Remove the particle with the smaller weight using Eq. 5.9
    end if
end for
```

We define ω^* as the weight corresponding to $[i^*, j^*]$. The target size corresponding to the maximum weight is then selected as the size of the bounding box for the current frame [48]. For the target position, the peak of the correlation response map of the maximum weight is added to the location corresponding to the maximum weight as discussed in [48]. Thus, the final target state x_t is calculated by comparing the convolutional features of particle i^*th (the best particle) and the j^*th target model (the best target model). We define the best model as $\mho_{t-1}^*$ which is one of the C_{t-1} target models generated in frame $t-1$.

We then use the positions estimated in frame t as the locations of the new particles for frame $t+1$ as explained in Algorithm 5.3. The peak of the correlation response map of particle i compared with target model j is given by [48]

$$[\delta_u^{(i)(j)}, \delta_v^{(i)(j)}] = \arg\max_{m,q} R_{(m,q)}^{(i)(j)}. \tag{5.4}$$

The target position estimated by that particle and target model is then given by [48]

$$[\tilde{u}_t^{(i)(j)}, \tilde{v}_t^{(i)(j)}] = [u_t^{(i)} + \delta_u^{(i)(j)}, v_t^{(i)} + \delta_v^{(i)(j)}], \tag{5.5}$$

where $[u_t^{(i)}, v_t^{(i)}]$ corresponds to the location of particle i. For the location of each particle $x_t^{(i)}$, we estimate C_{t-1} target positions. The updated state of particle i compared with target

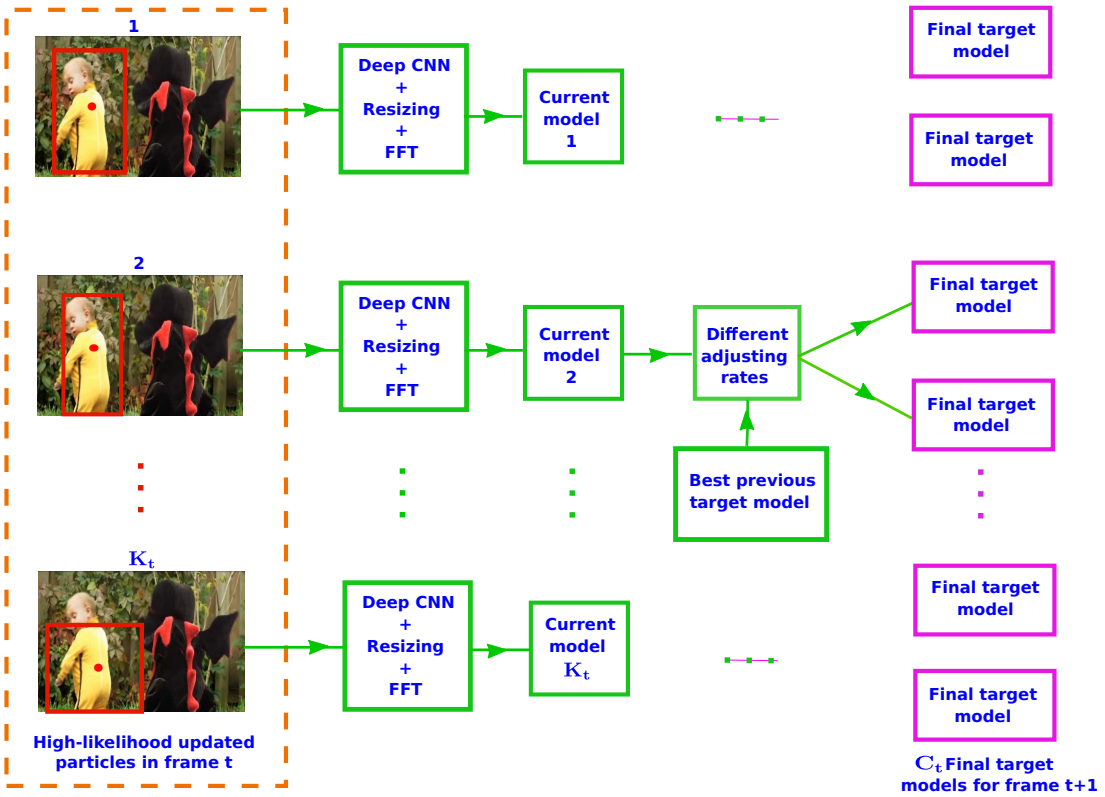


Figure 5.4: Our proposed multiple models with different adjusting rates.

Algorithm 5.4 Generate multiple target models

Input: Current frame, maximum weight ω^*, updated particles $\bar{x}_t^{(i)(j)}$, their weights $\omega^{(i)(j)}$ and the best target model $\mho_{t-1}^*$

Output: C_t target models $\mho_t^{(j)}$ for the next frame $t+1$

Examine Eq. 4.13 to determine the high-likelihood states
Generate K_t current target models $\breve{\mho}_t^{(j)}$ based on the high-likelihood states
for Each $\breve{\mho}_t^{(j)}$ **do**
    if Eq. 5.10 is correct **then**
        Select the set with bigger adjusting rates S_1
    else
        Select the set with smaller adjusting rates S_2
    end if
    Generate Γ final target models $\mho_t^{(j)}$ for the next frame based on Eq. 5.15
end for

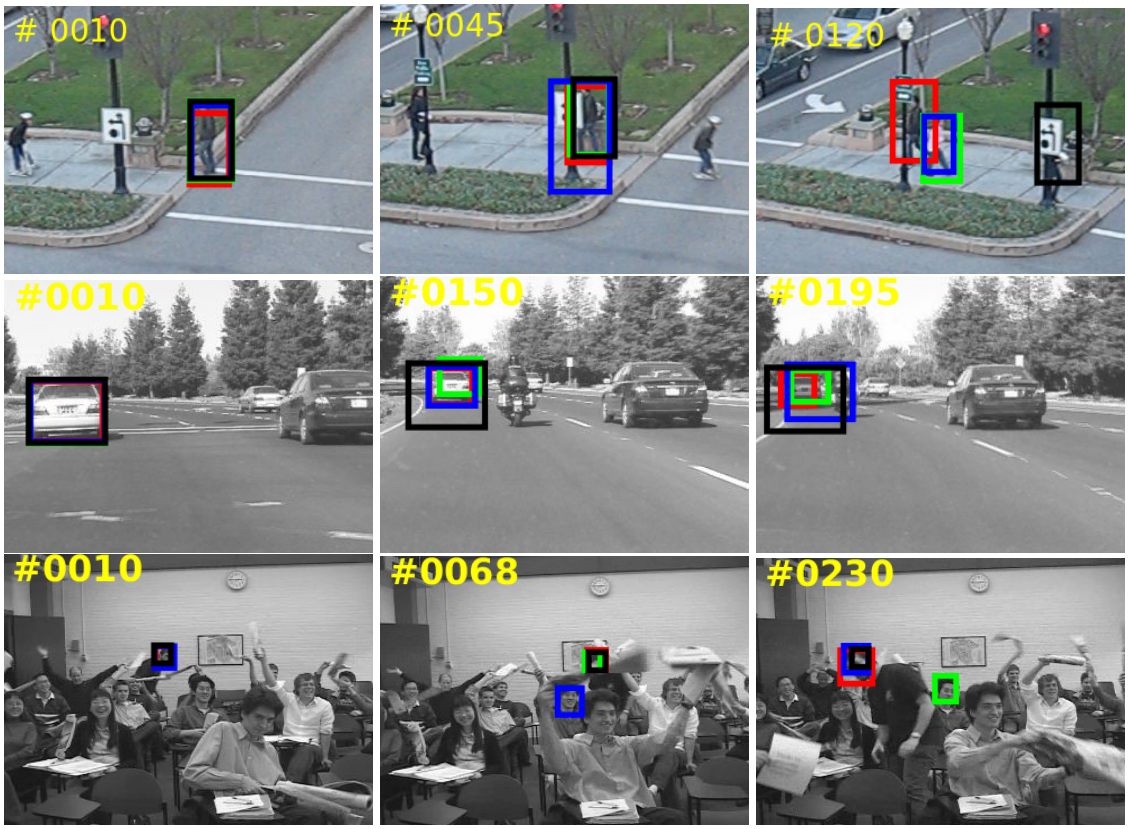


Figure 5.5: Qualitative evaluation of our tracker, *DCPF2*, *SINT* and *HCFT* on three challenging sequences (from up to down are *Human3*, *Car1* and *Freeman4*, respectively).

model j in frame t is

$$\bar{z}_t^{(i)(j)} = \left[\bar{x}_t^{(i)(j)}, \dot{\bar{x}}_t^{(i)(j)}\right]^T, \tag{5.6}$$

where $\dot{\bar{z}}_t^{(i)(j)}$ is the updated version of $\dot{x}_t^{(i)}$ based on $[\tilde{u}_t^{(i)(j)}, \tilde{v}_t^{(i)(j)}]$ and

$$\bar{x}_t^{(i)(j)} = \left[\tilde{u}_t^{(i)(j)}, \tilde{v}_t^{(i)(j)}, h_t^{(i)}, w_t^{(i)}\right]^T. \tag{5.7}$$

$\bar{x}_t^{(i)(j)}$ is rounded because target positions and sizes are discrete quantities measured in pixels. After rounding, several $\bar{x}_t^{(i)(j)}$ may have the same location. Since the initial particles can refine their locations for subsequent frames, these refined particles perform better than newly sampled particles in every frame. Their locations can also merge especially in simple frames to decrease the number of particles. Thus, the number of particles in simple frames

is lower. For the target size, our tracker samples around each remaining particle based on its velocity. Consider two particles $\bar{x}_t^{(i')(j')}$ and $\bar{x}_t^{(i'')(j'')}$, if

$$[\tilde{u}_t^{(i')(j')}, \tilde{v}_t^{(i')(j')}] = [\tilde{u}_t^{(i'')(j'')}, \tilde{v}_t^{(i'')(j'')}], \tag{5.8}$$

and

$$\omega^{(i')(j')} > \omega^{(i'')(j'')}, \tag{5.9}$$

$\bar{x}_t^{(i')(j')}$ is selected and $\bar{x}_t^{(i'')(j'')}$ is removed, that is two particles converge to the same target position, the one with the highest weight is selected and the other is removed. In the next step, the C_t target models are generated as discussed in the following section. The resampling conditions are then examined for frame $t+1$. The first condition is

$$\omega^{(i^*)(j^*)} > \varphi \times T_w. \tag{5.10}$$

As explained earlier, T_w is the maximum particle weight in the second frame. This maximum weight is calculated based on comparing the model generated by the ground truth in the first frame. Assuming there is no challenging scenario in the second frame, T_w is a good representative of the target. When the maximum weight in frame t is less than $\varphi \times T_w$, where φ is a threshold, it means our tracker could not produce a reliable correlation response map because of challenging scenarios such as occlusion. Therefore, the particles can not properly refine their locations based on these weak correlation response maps. In these scenarios, we resample new particles. The second resampling condition is

$$N_t \cdot C_{t-1} - Z > T_t, \tag{5.11}$$

where T_t is the minimum number of particles to transfer to the next frame. Let Z be the number of $\bar{x}_t$ which are removed. When too many particles converge to the same location and Z is too high, we increase the number of the particles by resampling. We then predict the new particles for frame $t+1$ based on the remaining particles in frame t according to

$$\hat{x}_{t+1}^{(p)} = A\bar{x}_t^{(i)(j)}, \tag{5.12}$$

where $p = 1, ..., N_t \cdot C_{t-1} - Z$ is the index of the particles which are transferred to the next frame $t + 1$. We generate β samples to be added to the target size of each $\hat{x}_{(t+1)}^{(p)}$ according to

$$x_{t+1}^{(f)} = \hat{x}_{t+1}^{(p)} + [0_{(1,4)}, \zeta_{t+1}], \tag{5.13}$$

where $\zeta_{t+1} \in \mathbb{R}^4$ is drawn from a zero-mean normal distribution and $f = 1, ..., \beta$. Thus, the number of particles for frame $t + 1$ is

$$N_{t+1} = \beta \cdot (N_t \cdot C_{t-1} - Z). \tag{5.14}$$

If the resampling conditions are met, the selected target state x_t for frame t is applied in Eq. 4.3 , Eq. 4.4 and 3.5 to generate the new particles. Fig. 5.2 illustrates how the number of particles decreases in simple frames. Additionally, the figure shows when the maximum weight significantly decreases comparing with T_w, resampling is implemented.

5.1.2 Multiple Correlation Models

In this section, we generate C_t target models for frame t to be employed in frame $t+1$. A target model is generated by giving a patch to the CNN and applying Fourier transform to the calculated convolutional features. This model distinguishes the background from the foreground. Fig. 5.4 illustrates our method for generating several target models. As discussed in [48], a comparison between the best model and the most accurate target size and position results in higher weight. Similar to [48] and explained in the previous chapter, we select all high-likelihood target states by examining the following equation over all $N_t \times C_{t-1}$ weights according to 4.13.

where we set α to 0.8. the target states corresponding with the selected weights are considered as high-likelihood candidates [48]. A target model is generated based on each K_t selected high-likelihood particles in frame t, as discussed in [48]. Let $\breve{\mathfrak{O}}_t^{(j)}$ represent one of K_t target models generated from the current frame, where $j = 1, ..., K_t$. The final

target models to be used in frame $t+1$ are a combination of the current target models and the previously selected target model according to [44]

$$\mho_t^{(j)} = (1-\Upsilon)\mho_{t-1}^{*} + \Upsilon\breve{\mho}_t^{(j)}, \tag{5.15}$$

where Υ is named adjusting rate. As seen in Fig. 5.3, the adjusting rate has a significant influence on the correlation tracker's performance. In our tracker, we consider different adjusting rates and then apply them to Eq. 5.15. Thus, instead of having K_t target models, we have C_t ones

$$C_t = \Gamma K_t, \tag{5.16}$$

where Γ is the number of adjusting rates. We define two sets of adjusting rates S_1 and S_2. Based on Eq. 5.10, we use one of those sets. It means that when the tracker can not pass Eq. 5.10, the correlation response maps are not reliable because of challenging scenarios and we should use smaller adjusting rates to increase the effect of the previous target model and decrease the current one. When Eq. 5.10 is passed, we can use bigger adjusting rates. Algorithm 5.4 explains the method of generating multiple target models.

5.2 Results and Discussion

We used OTB100 [6] to test our tracker performance. We select $N_t = 300$, $\varphi = 0.7$, $Tr2 = 4$, $\beta = 5$, $\Gamma = 3$, $S_1 = [0.075, 0.01, 0.015]$ and $S_2 = [0.075, 0.005, 0.001]$. The number of adjusting rates in each set is limited to three because of the computation costs. However, these sets are defined based on experiments.

We compared our tracker with eight state of the art trackers including: CFNet-conv3 [90], SiameseFC [91], SINT, LCT [31] and CNN-SVM [81], HDT, HCFT and DCPF2. As illustrated in Fig. 5.6, our overall performance of precision and success are improved by 3.5% and 5.5% compared to DCPF2, which are the second and fourth-best trackers in the precision and success plots, respectively. Our tracker outperforms SINT, the second-best

tracker in the success plot, by around 4.5%. On deformation and occlusion, our performance is better because we employ different adjusting rates and decrease the updating rate of the model in challenging frames. As seen in Fig. 5.6, for deformation and occlusion, our performance is improved around 4.5% on the precision plot and 6% on the success plot in comparison with the second-best tracker. For other challenging scenarios such as motion blur, background clutter and out of plane rotation, our tracker shows improvements of approximately 5%, 3.5% and 3%, respectively, compared to the second-best tracker.

Fig. 5.5 illustrates the qualitative evaluation of our tracker comparing to DCPF2, SINT [92] and HCFT. In the first data sequence *Human3*, the lower adjusting rate helps our tracker to use the previous target model more in the correlation filter. In the second data sequence *Car1* and third one *Freeman4*, the influence of using a more reliable sampling method for generating particles and employing different adjusting rates is shown.

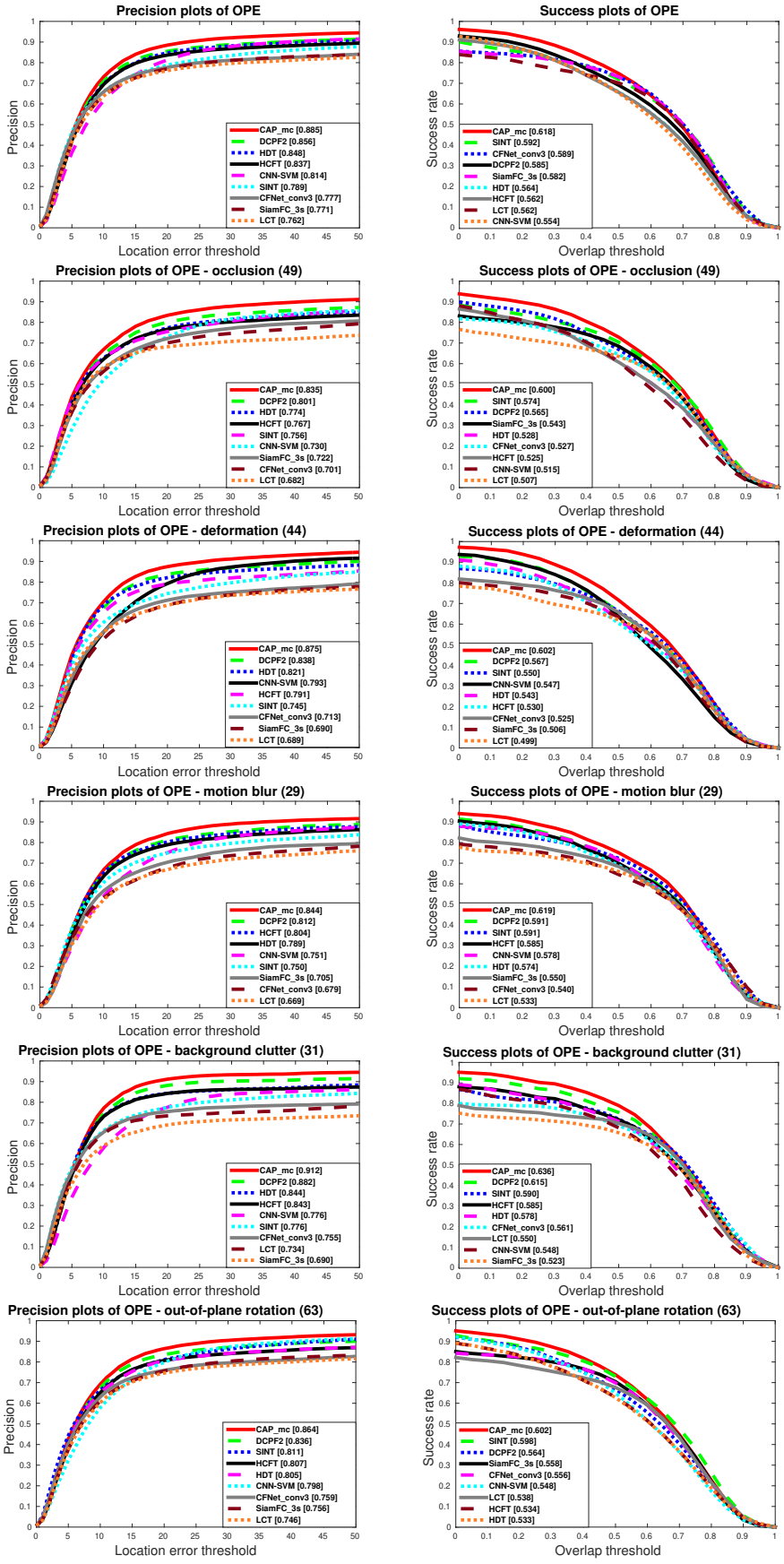


Figure 5.6: Quantitative evaluation of our tracker in comparison with state-of-the-art trackers on OPE.

CHAPTER 6
LIKELIHOOD PARTICLE FILTER

In this chapter, we propose a novel particle filter for CNN-correlation visual trackers. Our method uses correlation response maps to estimate the likelihood distributions and employs these likelihoods as proposal densities to sample particles. Likelihood distributions are more reliable than proposal densities based on target transition distributions such as [60, 77, 45, 48] because correlation response maps provide additional information regarding the target's location. Additionally, these trackers calculate the posterior distribution based on the peaks of the correlation maps without considering them in the computation of the particle weights. Our method also solves this problem by calculating a likelihood distribution and using it as the proposal density.

Furthermore, our particle filter searches for multiple modes in the likelihood distribution, which improves performance in target occlusion scenarios while decreasing computational costs by more efficiently sampling particles. In other challenging scenarios such as those involving motion blur, where only one mode is present but a larger search area may be necessary, our particle filter allows for the variance of the likelihood distribution to increase. We tested our algorithm on OTB100 [6] and our experimental results demonstrate that our framework outperforms state-of-the-art methods.

6.1 The change of support problem in convolution-correlation particle filters

In particle filters used in correlation trackers such as [45, 75, 48, 50, 51, 76, 77], the particles are shifted to the peak of the correlation maps and the posterior distribution in Eq. 2.5 is then changed to

$$p(x_t|y_t) \approx \sum_{i=1}^{N} \varpi_t^{(i)} \delta(x_t - \tilde{x}_t^{(i)}), \tag{6.1}$$

where $\tilde{x}_t^{(i)}$ is the peak of the correlation response map corresponding to the i-th particle and $\varpi_t^{(i)}$ is the normalized weight of the i-th particle.

However, the posterior distribution must take into consideration the weights corresponding to the shifted locations, not the original particles. As seen in the top and middle rows of Fig. 6.1, it is possible that multiple particles with different correlation maps and weights converge to the same location, which means the posterior distribution would then include multiple particles at a common location but with different weights, which invalidates the assumption that the likelihood depends solely on $x_t^{(i)}$. In other words, the patch centered in the shifted location generates different features and consequently a different weight from the corresponding patch centered in the particle as shown in the middle and bottom rows of Fig. 6.1. Thus, the posterior distributions estimated by these trackers are not accurate.

Because the posterior distributions generated by these trackers are not reliable, they resort to resampling in every frame. While resampling is a suitable solution to avoid sample degeneracy that should be performed when necessary, resampling at every frame causes loss of information. It also causes sample impoverishment (i.e., loss of diversity among particles) and may cause all the particles to collapse to a single point within a few frames. To solve the mentioned problems, we sample particles from the likelihood distribution instead. Particle filters that sample from likelihood distributions generate more accurate particles, but sampling from the likelihood distribution is not always possible. Fortunately, CNN-correlation trackers generate correlation maps that can be used in the construction of likelihood distributions.

6.2 Proposed Algorithm

Our algorithm generates an initial correlation response map for the current frame based on the previously estimated target state to calculate an initial likelihood distribution. That is, we generate a patch from the current frame based on the previous target

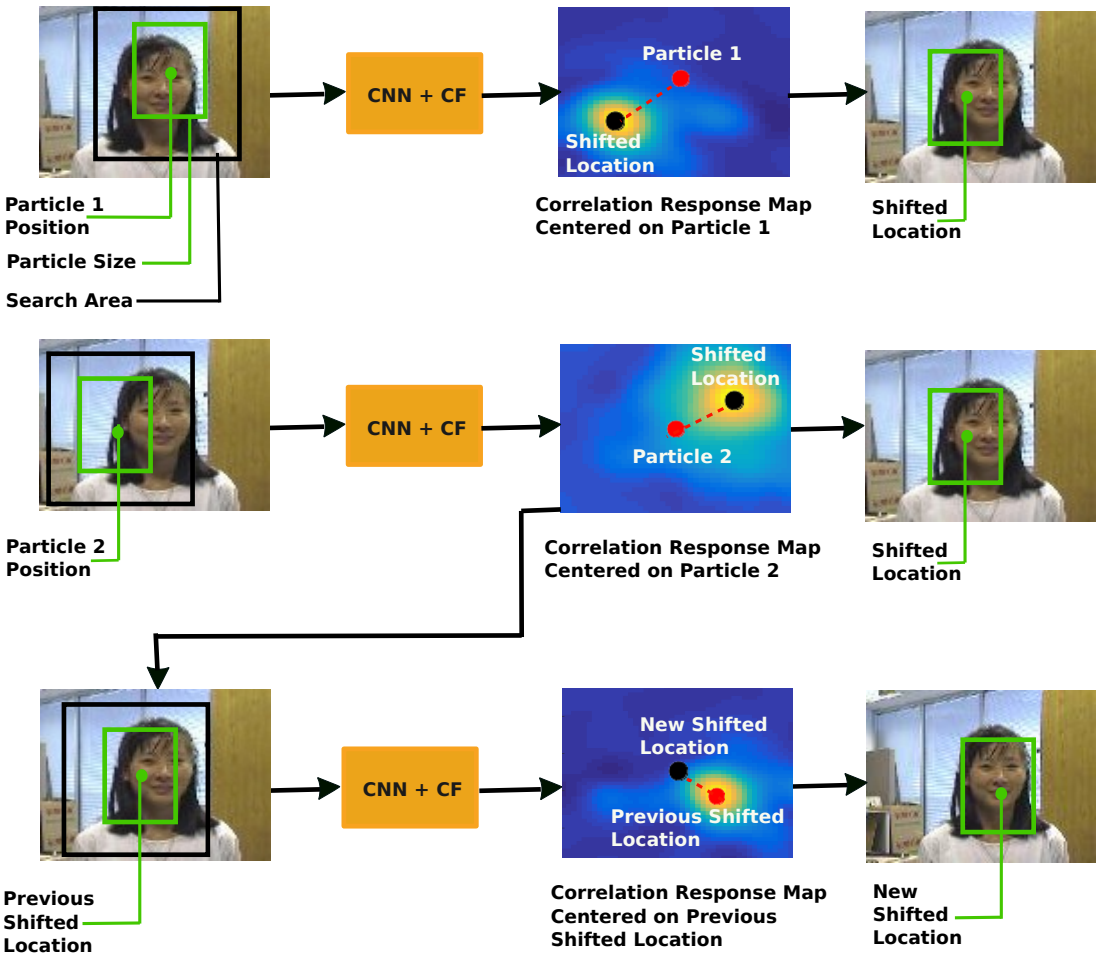


Figure 6.1: The top and middle rows illustrate how two distinct particles with different weights converge to the same shifted location. Two different patches centered in particles 1 and 2 are given to the CNN and correlation filter to generate their correlation response maps. Each of these particles is then shifted to the peak of its corresponding correlation response map. As shown, their shifted locations are identical. Thus, this location is associated with two different weights in the posterior distribution because the correlation maps corresponding to the two particles are different. The middle and bottom rows show how the particle and its shifted location may generate different correlation maps. In the bottom row, a patch centered in the shifted location of particle 2 is generated. This patch results in a different correlation map at a different shifted location in comparison with the patch corresponding to particle 2. Thus, if the shifted location of particle 2 is used in the computation of the posterior distribution, its weight should be calculated based on the correlation map on the bottom row instead of the one on the middle row.

state and use a CNN [10] to extract the convolutional features from this patch. We then compare these features with the target model to calculate the final correlation response

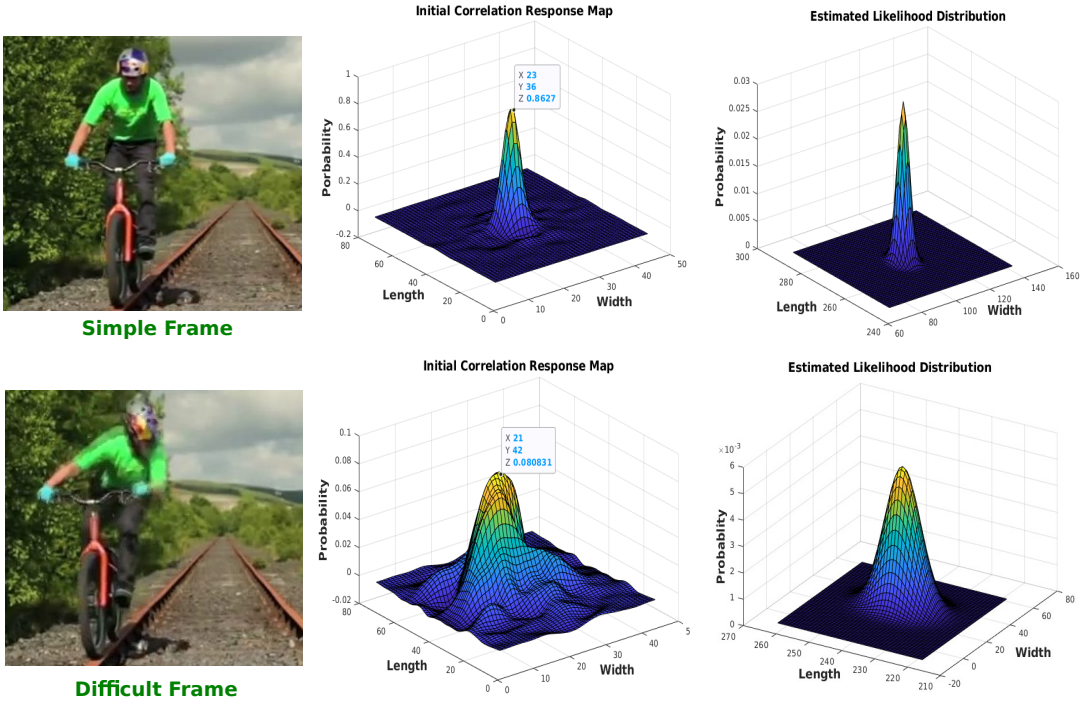


Figure 6.2: Estimated likelihood distributions for common scenarios (simple frame) and a challenging scenario involving fast motion (difficult frame).

map [44]. As seen in Fig. 6.2, in most scenarios (which we call "simple frames"), the correlation response map corresponds to a sharp Gaussian distribution with a prominent peak. In challenging scenarios ("difficult frames"), correlation maps are wider with less pronounced peaks. We need to estimate likelihood distributions consistently in both scenarios. To address this issue, we fit a Gaussian distribution to the correlation response maps while disregarding elements with probability lower than a threshold τ. By disregarding low probability elements, we mitigate the impact of the background on the computation of the model. We compute the mean of the correlation response map using

$$\mu \approx \frac{\sum_{i=1}^{u} q_i s_i}{\sum_{i=1}^{u} q_i}, \tag{6.2}$$

where s_i and q_i represent the elements of the correlation response map and their respective probabilities, and u is the number of elements with probability higher than τ. The variance

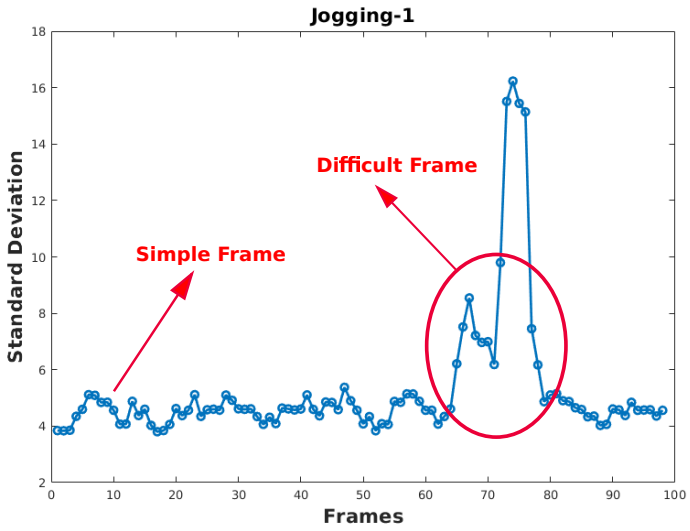


Figure 6.3: Standard deviations of the estimated likelihood distributions in data sequence *Jogging-1* of the OTB-100 dataset.

of the response map is then given by

$$\sigma^2 \approx \frac{\sum_{i=1}^{u} q_i (s_i - \mu)^2}{\sum_{i=1}^{u} qi}. \tag{6.3}$$

Thus, our model assigns low probabilities to pixels that are likely to belong to the background while assigning relatively high probabilities to all the regions that might correspond to the target. As a result, our samples concentrate in regions where the target is more likely to be present.

Fig. 6.2 shows our estimated likelihood distributions for two different frames of the *Biker* data sequence of the OTB100 benchmark. In the difficult frame, the target undergoes motion blur, which causes the correlation response map to be wider with a lower peak. Our estimated variance is then correspondingly higher, which helps our tracker to sample particles over a wider area to compensate for tracking uncertainties in difficult scenarios. The example in Fig. 6.3 shows how the variance increases as the target approaches difficult frames.

Although allowing for higher variances in challenging scenarios such as those involving fast motion helps our tracker address such issues, this strategy alone cannot handle multi-modal correlation response maps. To resolve this issue, we propose to determine the peaks of the distribution using the approach described below.

6.2.1 Multi-modal likelihood estimation

The existence of multiple peaks in a correlation response map usually indicates the presence of confusing elements in the background of the frame, as the example in Fig. 6.4 illustrates. In the frame shown in the figure, there are two peaks in the correlation response map when partial target occlusion occurs. The peaks correspond to the woman on the left side of the image (the target) and the pole partially occluding her. By applying a threshold to remove low probability elements from the correlation response map, two clusters become apparent.

To identify the peaks of the correlation map while disregarding additional background clutter, we remove from the map points with probability lower than a threshold τ. We then fit a Gaussian mixture model to the remaining feature map points [93]. Our model tests different numbers of clusters up to 3 groups and finds the optimal number k using the silhouette score method [94]. Fig. 6.5 shows two instances of correlation response maps in which we identify $k = 2$ and $k = 3$ clusters. The likelihood corresponding to each peak is then given by a normal distribution with mean and variance given by Eqs. 6.2 and 6.3. Algorithm 6.1 summarizes our proposed approach to estimate the likelihood distribution for each cluster.

6.2.2 Particle sampling

We sample particles from the Gaussian likelihood distributions obtained from the correlation response maps in the current frame. The probability that a particle is sampled

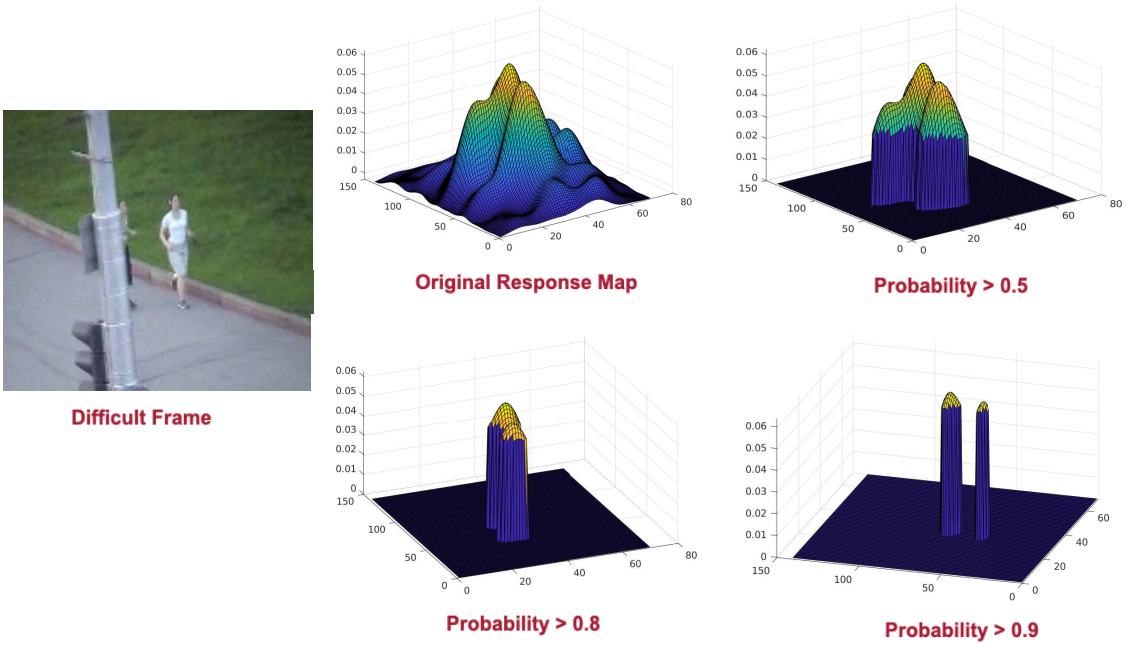


Figure 6.4: A difficult frame including target occlusion. Its correlation response map has two peaks. By increasing the threshold to remove low probability elements, two clusters corresponding to the target and the pole are seen.

Algorithm 6.1 Multi-modal likelihood estimation

Input: Current frame y_t and previous target state x_{t-1}
Output: One likelihood distribution for each correlation map cluster

1: Extract a patch from the current frame based on the previous target state
2: Extract the CNN features of the patch and calculate its correlation response map
3: Remove points with probability lower than τ
4: Fit a Gaussian mixture model to the map and find the clusters
5: Estimate the likelihood distribution of each cluster based on the mean and variance of its elements in the map according to Eqs. 6.2 and 6.3

from the likelihood distribution is given by

$$p(x_t^{(i)}|y_t) \propto \sum_{j=1}^{k} \mathcal{N}\left(x_t^{(i)}; \mu_j, \sigma_j\right), \tag{6.4}$$

where μ_j and σ_j are the mean and variance of the j-th mode of the likelihood. We generate a patch for each particle and extract its features using a CNN. After calculating the

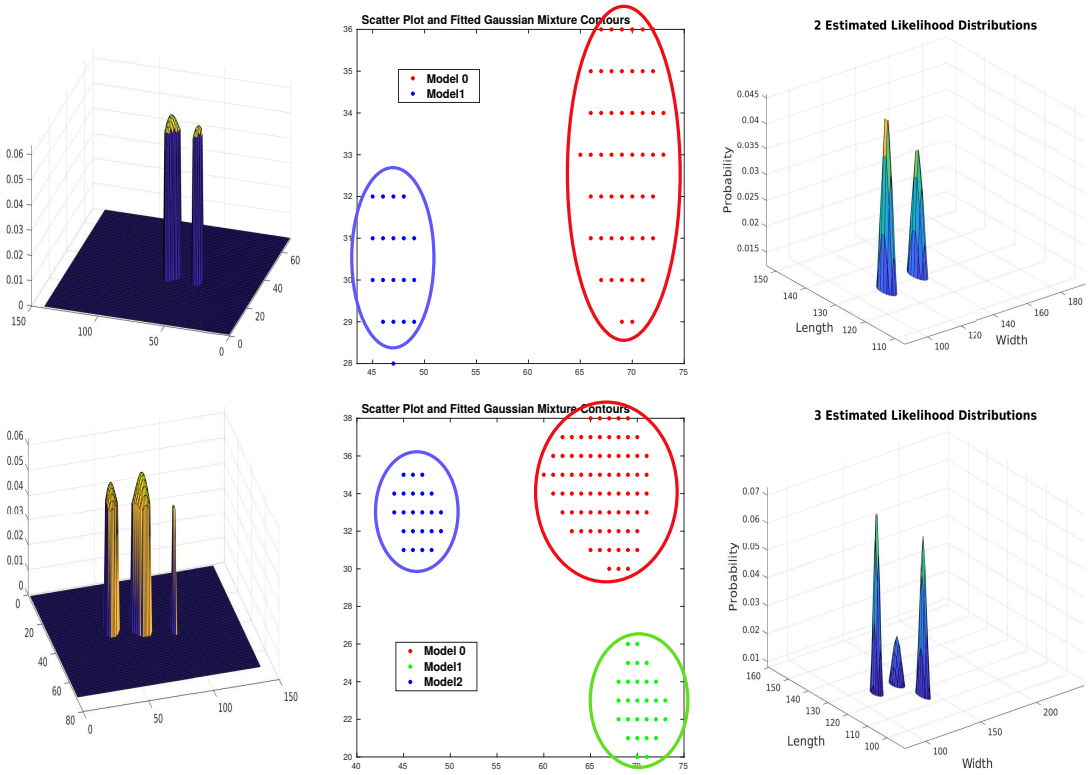


Figure 6.5: Finding clusters; left: correlation response maps with two and three clusters, middle: clusters of the correlation response maps obtained by fitting a Gaussian mixture model, right: estimated likelihood distributions for each cluster.

correlation response map for each particle, we shift the particles to the peaks of their respective correlation response maps. The peak of each correlation response map is the estimated target position based on the patch centered at the corresponding particle. Because each particle is shifted to the peak of the correlation response map, we consider $p(\tilde{x}_t^{(i)}|x_t^{(i)}) = 1$, where $\tilde{x}_t^{(i)}$ is the peak of the corresponding correlation response map. As a result, $p(x_t^{(i)}|y_t) = p(\tilde{x}_t^{(i)}|y_t)$.

6.2.3 Calculating the weights and posterior distribution

By computing the weight corresponding to each shifted particle $\tilde{x}_t^{(i)}$, we can accurately estimate the posterior based on the shifted particles and their correct weights, which

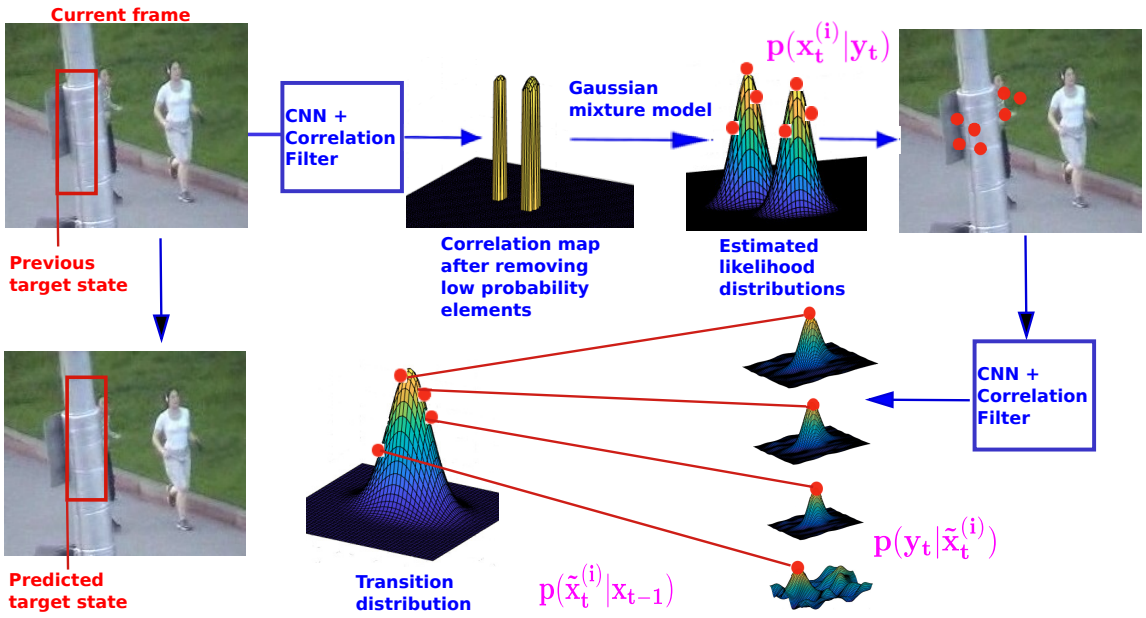


Figure 6.6: Overview of the steps comprising the proposed DCPF-Likelihood visual tracker.

addresses the problem of incorrect support points observed in previous works. The correct weight corresponding to each shifted particle is then given by

$$\omega_{\tilde{x}_t}^{(i)} \propto \omega_{x_{t-1}}^{(i)} \frac{p(y_t|\tilde{x}_t^{(i)})p(\tilde{x}_t^{(i)}|x_{t-1})}{q(\tilde{x}_t^{(i)}|x_{t-1}, y_t)}, \tag{6.5}$$

where the term corresponding to the previous weight is removed because we perform re-sampling at every frame. Additionally, the proposal density is given by

$$q(\tilde{x}_t^{(i)}|x_{t-1}, y_t) = p(\tilde{x}_t^{(i)}|y_t). \tag{6.6}$$

Thus, the weight of each shifted particle is

$$\omega_{\tilde{x}_t}^{(i)} \propto \frac{p(y_t|\tilde{x}_t^{(i)})p(\tilde{x}_t^{(i)}|x_{t-1})}{p(\tilde{x}_t^{(i)}|y_t)}. \tag{6.7}$$

Let the target state be defined based on Eq. 4.1 and Eq. 4.2. We apply a first-order motion model to the previous target state z_{t-1} according to Eq. 4.3 and Eq. 4.4 to find the predicted target state $\hat{z}_{t-1}$. We use a Gaussian distribution $\mathcal{N}(\bar{x}_{t-1}, \sigma^2)$ to find the probability of each estimated particle in the current frame $p(\tilde{x}_t^{(i)}|x_{t-1})$.

Additionally, $p(y_t|\tilde{x}_t^{(i)})$ is the likelihood of each shifted particle. The correlation response map $R^{(i)}$ is then calculated based on Eq. 4.5 and Eq. 4.6. The peak, $\tilde{x}_t^{(i)}$, of the correlation response map is then calculated based on Eq. 4.9 and Eq. 4.12. The likelihood of $\tilde{x}_t^{(i)}$ is calculated by [48]

$$p(y_t|\tilde{x}_t^{(i)}) = \frac{1}{M \times Q} \sum_{m,q} R^{(i)}(m, q). \tag{6.8}$$

The posterior distribution based on the shifted particles and their respective weights is then

$$\hat{Pr}(x_t|y_t) \approx \sum_{i=1}^{N} \varpi_{\tilde{x}_t}^{(i)} \delta(x_t - \tilde{x}_t^{(i)}), \tag{6.9}$$

where $\varpi_{\tilde{x}_t}^{(i)}$ is the normalized version of $\omega_{\tilde{x}_t}^{(i)}$. Fig. 6.6 summarizes the steps of our method, and Algorithm 6.2 describes the details of our approach.

Algorithm 6.2 DCPF-Likelihood visual tracker.

Input: Current frame y_t and previous target state x_{t-1}
Output: Current target state x_t

1: Estimate a likelihood distribution for each cluster using Algorithm 6.1
2: Sample particles from the likelihood distributions $p(x_t^{(i)}|y_t)$
3: Extract the CNN features of the patches corresponding to each particle and calculate its correlation response map
4: Shift the particles to the peaks of their correlation response maps
5: Calculate the likelihood $p(y_t|\tilde{x}_t^{(i)})$ based on Eq. 6.8
6: Calculate the transition probability $p(\tilde{x}_t^{(i)}|x_{t-1})$
7: Compute the weight of each shifted particle $\omega_{\tilde{x}_t}^{(i)}$ according to Eqs. 6.5 to 6.7
8: Calculate the posterior distribution according to Eq. 6.9

6.3 Experimental results

We assess the performance of our tracker on the OTB100 dataset. Fig. 6.7 shows the OPE results for our tracker in comparison with DCPF, HCFT, and CNN-SVM. Our overall performance improvements over DCPF, the second best tracker, in terms of precision and

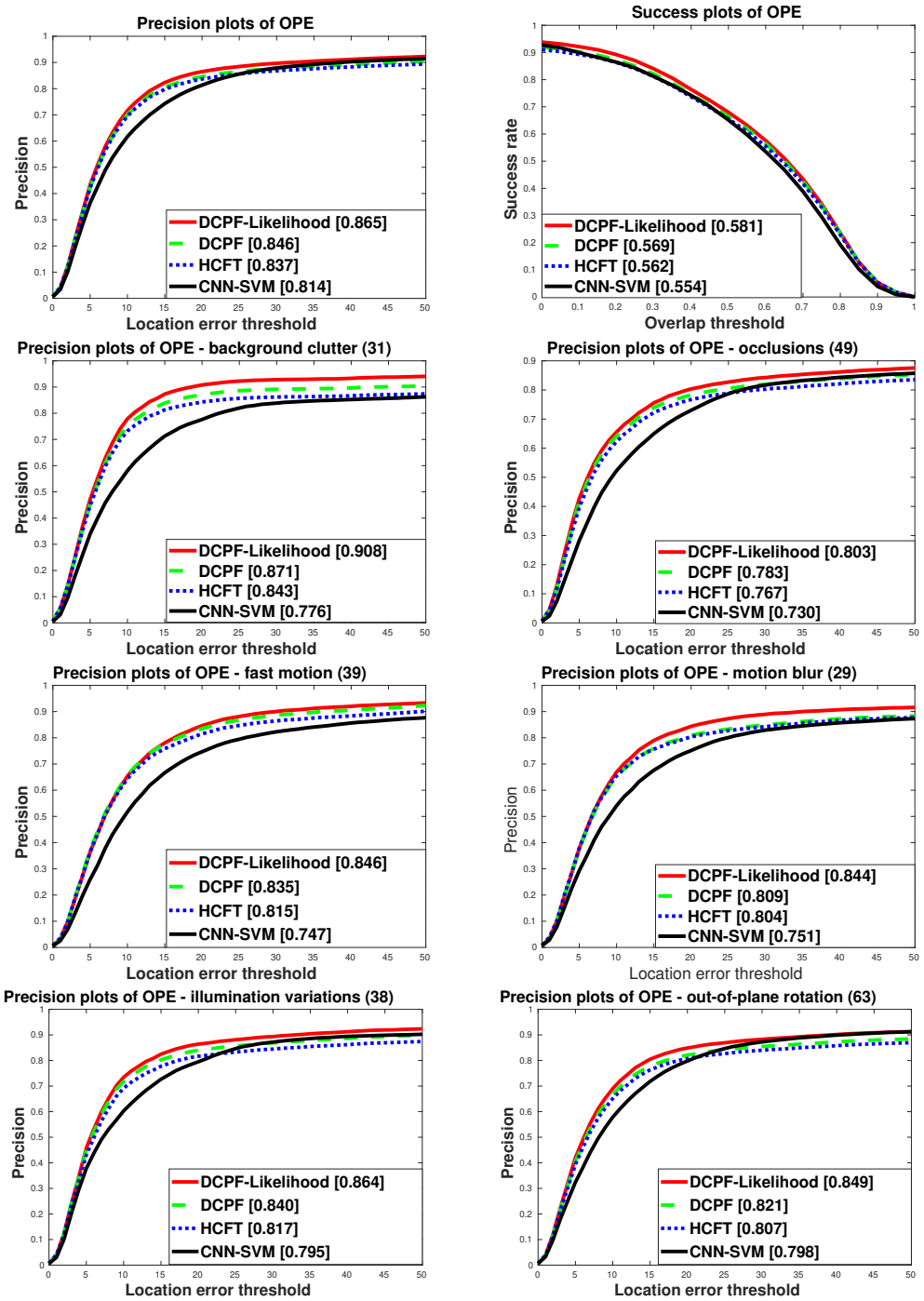


Figure 6.7: One pass evaluation of our tracker in comparison with three state-of-the-art approaches.

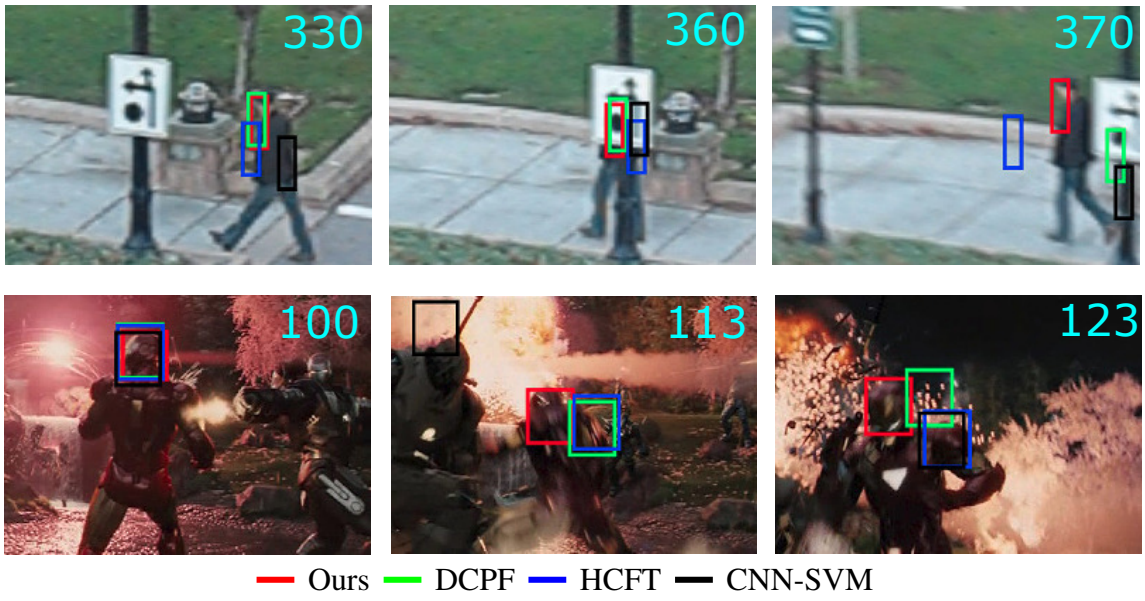


Figure 6.8: Qualitative evaluation of our tracker against *DCPF*, *HCFT*, and *CNN-SVM* on two challenging sequences: *Human6* (top) and *Ironman* (bottom).

success rates are 2.5% and 2%, respectively. Our method outperforms DCPF particularly in scenarios involving occlusions (+3%) and background clutter (+4.5%). DCPF uses the transition distribution as the proposal density, a common approach in particle-correlation trackers. Our results show that the likelihood is a more effective proposal distribution. In scenarios involving motion blur and fast motion, our performance improvements over DCPF are around 4.5% and 2%, respectively, because our tracker increases the variance of the likelihood distribution to spread out particles across a wider area. Our method also outperforms DCPF in scenarios involving illumination variation (+3%), out-of-plane rotation (+3.5%), and deformation (+3%). Our method also decreases the computational cost of the algorithm. Our tracker uses 100 particles, which is significantly less than the 300 particles used in DCPF.

Fig. 6.8 shows qualitative results comparing our tracker with DCPF [45], HCFT [44], and CNN-SVM [81]. In both data sequences shown in the figure, our method suc-

cessfully handles occlusion scenarios. These results highlight the impact of using more reliable sampling distributions.

CHAPTER 7
ITERATIVE PARTICLE FILTER

In this chapter, we propose a deep convolutional correlation iterative particle filter (D2CIP) tracker. Our proposed tracker uses multiple particles as the inputs to a CNN [10] and then applies the correlation filter used in the ECO tracker [69] to generate the correlation map of each particle. Large displacements between the previous target position and the peak of the correlation map, shown in Fig. 2.3, lead to a degradation in the quality of the correlation map, since the corresponding convolutional features are less similar to the target model. Our proposed iterative particle filter decreases this distance for each particle through an iterative procedure. At each iteration, the particles approach the target location and an improved correlation map is computed. To our knowledge, iterative particle filters have not been used in conjunction with CNNs and correlation filters before.

The second major contribution of this chapter is a novel target state estimation strategy. In state-of-the-art particle-correlation trackers such as [45, 75, 48, 50, 51, 76, 77], assessing the likelihood of the particles is challenging because many particles may be in close proximity to one another. In our framework, the particles converge to a few locations after a series of iterations. Thus, we propose a novel method based on particle convergence consistency and K-means clustering to evaluate the final particle locations. This novel method enables our proposed tracker to overcome challenges associated with multi-modal likelihood distributions.

Additionally, state-of-the-art particle-correlation trackers must perform resampling at every frame because shifting the particles to the peak of the correlation maps changes the support of the posterior distribution as we discussed in Section 6.1. We addressed the aforementioned problems in the previous chapter by proposing a likelihood particle filter. In that method, although the peaks of the correlation maps are used as the proposal and transition distributions, the weights of the peaks are still calculated based on the likelihood

of the particles instead of the likelihood of the corresponding peaks. Our proposed iterative particle filter is a novel solution for correlation-convolutional trackers to calculate an accurate posterior without performing resampling at every frame. As a third contribution, our iterative particle filter overcomes this issue and hence does not disregard information from prior samples. We tested our tracker on the OTB100 and the LaSOT datasets, and the results show that our tracker outperforms several state-of-the-art methods.

7.1 Proposed Algorithm

This section discusses our proposed strategy to generate particles that better reflect the actual position of the target while avoiding the change of support problem discussed in Section 6.1. Our approach is based on an iterative particle filter that gradually shifts the particle positions to locations that are closer to the peak of the correlation response map while also updating the response maps themselves so that they become less sensitive to background clutter and better aligned with the target position. As the particles are updated, their corresponding weights are also recomputed based on the new correlation response maps.

7.1.1 Iterative Particle Filter

As Fig. 7.1 illustrates, correlation filter-based trackers attempt to determine the position of the target by analyzing the displacement between the center of the correlation map, which corresponds to each particle in our tracker, and the peak of the map. A particle filter allows us to generate multiple samples around the predicted target state and hence increase our chances of finding maps with low displacement [45]. If this displacement is sufficiently low, then the features extracted from the CNN are similar to the model and the correlation response map is reliable. Our iterative particle filter considerably decreases this displacement and generates more reliable correlation maps for all the particles. As shown in Fig. 7.1, after generating the correlation response map for one particle, the distance be-

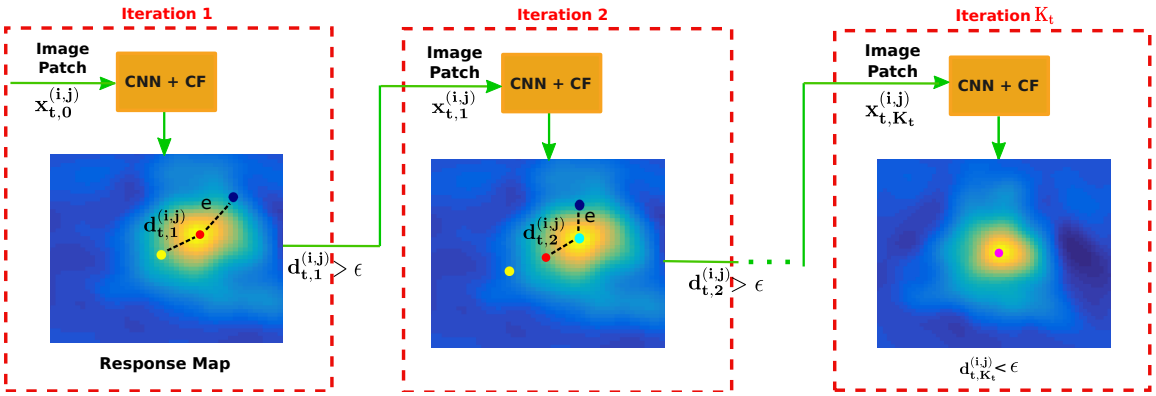


Figure 7.1: Illustration of the proposed iterative particle position refinement. In the first iteration, particle $x_{t,0}^{(i,j)}$ shown by the yellow point is given to the CNN and the correlation filter to calculate its response map. Because the displacement $d_{t,1}^{(i,j)}$ between the estimated position and the particle is higher than ϵ, the particle needs to be refined. In this scenario, the estimated position is not accurate and there is an error e between the peak of the correlation map and the ground truth position (black point). The red point is then considered the new particle $x_{t,1}^{(i,j)}$ for the second iteration. The cyan point shows the estimated position in the second iteration, which needs further refinement despite the reduction in the error e. The purple point represents the position at the K_t-th iteration, which does not need to be refined because $d_{t,K_t}^{(i,j)} < \epsilon$.

tween the particle position (yellow point) and the peak of the map (red point) is calculated. If the distance is larger than a small threshold ϵ, the correlation response map is not reliable enough to estimate the target position because it was generated based on an image patch centered at a position (yellow point) far from the ground truth location (black point). In such scenarios, the corresponding particle needs to be refined. To that end, the peak of the map is considered the new particle position and its corresponding correlation response map is calculated in a subsequent iteration. Although the peak of the new map (cyan point) is closer to the ground truth, the corresponding particle needs further refinement because the distance between the new particle position (which is now the red point) and the peak of the new map is larger than ϵ. Finally, in the K_t-th iteration, the calculated distance is smaller than ϵ and the iterative refinement procedure terminates. The peak of the final map (purple point) is considered the estimated target position for this particle. Since no shifting

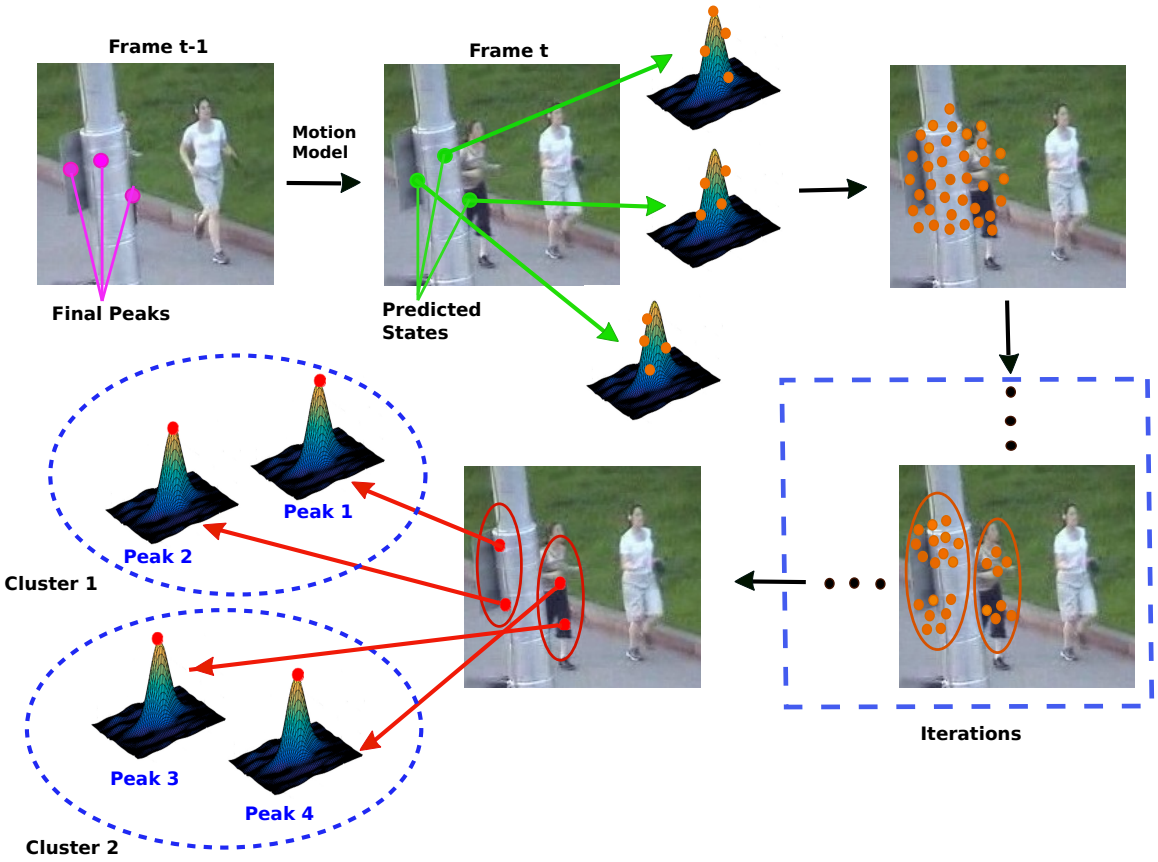


Figure 7.2: Illustration of the particle selection process for $J_{t-1} = 3$. The particles are sampled from three distributions whose means are given by the previous correlation map peaks. At time t, the particles converge to four final peaks at the end of the iterations. Two clusters are found by applying K-means to the final particle locations. After selecting the best cluster, the peak of the correlation response map corresponding to the cluster with the highest number of particles is selected as the target state.

is performed in the last iteration, the particle support problem discussed above is avoided. Our iterative particle filter is explained in greater detail in the following subsections.

7.1.1.1 Particle prediction model

As shown in the top row of Fig. 7.2, the posterior distribution of the target at time $t-1$ is modeled by a mixture of J_{t-1} normal distributions $\mathcal{N}(z_{t-1}^{(j)}, \sigma^2)$ where $j = 1, \ldots, J_{t-1}$ and $z_{t-1}^{(j)}$ is given by Eq. 4.1 and Eq. 4.2. For simplicity in calculating distances,

we consider $p_{t-1}^{(j)} = [u_t, v_t,]^T$ and $s_{t-1}^{(j)} = [h_t, w_t,]^T$. The predicted state is then calculated based on Eqs. 4.3 and 4.4. Our transition distribution is then given by the mixture

$$p(x_{t,0}|x_{t-1}) = \frac{1}{J_{t-1}} \sum_{j=1}^{J_{t-1}} \mathcal{N}(\hat{z}_t^{(j)}, \sigma^2). \tag{7.1}$$

We then sample $N_t \times J_{t-1}$ new particles $x_{t,0}^{(i,j)} \sim p(x_{t,0}|x_{t-1})$, where $i = 1, \ldots, N_t$. To increase the efficiency of our strategy, instead of sampling all the particles directly from the mixture distribution, we employ a stratified strategy and sample N_t particles from each of the J_{t-1} predicted normal distributions. Fig. 7.2 also illustrates the processes of refining and clustering the particles, which are discussed in more detail in the following sections. Algorithm 7.1 summarizes our iterative particle filter algorithm. Lines 1-4 correspond to the sampling method described above. The remaining steps of the algorithm are also discussed in the following sections.

Algorithm 7.1 Deep Convolutional Correlation Iterative Particle (D2CIP).

Input: Current frame at time t, target models, previous final peaks $z_{t-1}^{(j)}$ and their normalized weights $\varpi_{t-1}^{(j)}$

Output: Estimated target state x_t^*, updated target models, current final peaks $z_{t,K_t}^{(i,j)}$ and their normalized weights $\varpi_{t,K_t}^{(i,j)}$

1: Find the predicted distributions $\mathcal{N}(\hat{z}_t^{(j)}, \sigma^2)$
2: **for** each predicted state $\hat{z}_t^{(j)}$ **do**
3: Sample N_t initial particles $x_{t,0}^{(i,j)} \sim \mathcal{N}(\hat{z}_t^{(j)}, \sigma^2)$
4: **end for**
5: **for** each initial particle $x_{t,0}^{(i,j)}$ **do**
6: Find its corresponding final peak using Alg. 7.2
7: **end for**
8: **for** each final peak $x_{t,K_t}^{(i,j)}$ **do**
9: Calculate the peak weight $\omega_{t,K_t}^{(i,j)}$
10: **end for**
11: Estimate the target state x_t^* based on the final peaks $x_{t,K_t}^{(i,j)}$ using Alg. 7.3
12: Find the updated target models based on the final peaks $x_{t,K_t}^{(i,j)}$
13: Resample if the effective sample size is lower than γ

7.1.1.2 Iterative particle refinement

Particle $x_{t,0}^{(i,j)}$ is used to sample a patch from the current frame at time t and to generate the corresponding convolutional features. These features are compared with the target models to calculate the correlation response map $R_{t,0}^{(i,j)}$ using the correlation filter proposed in [69]. We maintain one target model for each of the predicted distributions, but to simplify the notation in this section, we refrain from explicitly differentiating the models. Let $p(y_t|x_{t,0}^{(i,j)})$ be the likelihood of $x_{t,0}^{(i,j)}$, which is given by the sum of the elements of $R_{t,0}^{(i,j)}$. We discard the samples for which $p(y_t|x_{t,0}^{(i,j)}) < L_{min}$, where L_{min} is the threshold to consider a correlation response map acceptable. As illustrated in Fig. 7.1, at each iteration $k = 1, \ldots, K_t$, the remaining samples are shifted to $x_{t,k}^{(i,j)}$, which is defined as

$$x_{t,k}^{(i,j)} = [p_{t,k}^{(i,j)}, s_{t,0}^{(i,j)}], \tag{7.2}$$

where $p_{t,k}^{(i,j)} = \arg\max(R_{t,k-1}^{(i,j)})$, i.e., the peak of the associated correlation response map at step $k-1$ of the iterative refinement process. We then have $d_{t,k}^{(i,j)} = ||p_{t,k}^{(i,j)} - p_{t,k-1}^{(i,j)}||$ as the Euclidean distance between $p_{t,k}^{(i,j)}$ and $p_{t,k-1}^{(i,j)}$. For each particle, the refinement procedure continues until $d_{t,k}^{(i,j)} < \epsilon$. Since particles in close proximity tend to generate correlation response maps whose peaks share a common location, all the particles converge to a small number high-likelihood points in the image. These peaks determine the means of the normal distributions used to generate the prediction model described in Section 7.1.1.1.

As seen in Fig. 7.2, let $\mathcal{N}(x_{t,k}^{(i,j)}, \sigma^2)$ be the normal distributions after the convergence of the k-th iteration. We select the mean of these normal distributions as the particles for the next iteration if $d_{t,k}^{(i,j)} \geq \epsilon$. After the iterations, the particles reach the final peaks $x_{t,K_t}^{(i,j)} = [p_{t,K_t}^{(i,j)}, s_{t,0}^{(i,j)}]$, which do not need further refinement because $d_{t,K_t}^{(i,j)} < \epsilon$. Thus, we have

$$p_{t,K_t}^{(i,j)} = p_{t,0}^{(i,j)} + \sum_{k=1}^{K_t} d_{t,k}^{(i,j)}. \tag{7.3}$$

Algorithm 7.2 explains how to reach the final peaks in our iterative particle filter. Additionally, all the normal distributions $\mathcal{N}(x_{t,K_t}^{(i,j)}, \sigma^2)$ are used in the process of updating

the target models as well. ECO examines only the estimated target state to update the target models, while our iterative particle filter provides all $\mathcal{N}(x_{t,K_t}^{(i,j)}, \sigma^2)$ for ECO to examine in the target model update process.

Algorithm 7.2 Iterative Particle Refinement.

Input: Current frame t, initial particles $x_{t,0}^{(i,j)}$, target models
Output: Final current peaks $x_{t,K_t}^{(i,j)}$

for each particle $x_{t,0}^{(i,j)}$ **do**
  $d_{t,k}^{(i,j)} = \infty$
  while $d_{t,k}^{(i,j)} > \epsilon$ **do**
    Generate the correlation response map $R_{t,k-1}^{(i,j)}$
    Calculate the likelihood $p(y_t|x_{t,k-1}^{(i,j)})$ based on $R_{t,k-1}^{(i,j)}$
    if $p(y_t|x_{t,k}^{(i,j)}) > L_{min}$ **then**
      $p_{t,k}^{(i,j)} = \arg\max(R_{t,k-1}^{(i,j)})$
      $d_{t,k}^{(i,j)} = ||p_{t,k}^{(i,j)} - p_{t,k-1}^{(i,j)}||$
    else
      Discard particle $x_{t,k-1}^{(i,j)}$
    end if
  end while
  Find $J_{t,K}$ for each final peak
end for

7.1.1.3 Weight update model

The posterior distribution for the particle filter is approximated by

$$p(x_t|y_t) \approx \sum \varpi_{t,K_t}^{(i,j)} \delta(x_t - x_{t,K_t}^{(i,j)}), \tag{7.4}$$

where $\varpi_{t,K_t}^{(i,j)}$ represents the normalized weights of the final correlation map peaks. As discussed in Section 6, all of the correlation-based particle filters proposed in the literature so far compute the likelihood of the particles based on the correlation response maps of their initial positions without considering the change of support caused by shifting the particles.

Since our approach keeps track of the particles that converge to a common location, it allows us to update the particle posterior distribution based on the likelihood of their final locations and their corresponding prior weights, i.e.,

$$\omega_{t,K_t}^{(i,j)} \propto p(y_t|x_{t,K_t}^{(i,j)}) \max_{\varpi_{t-1}^{(j)} \in \mathcal{X}_{t-1}^{j}} \varpi_{t-1}^{(j)}, \tag{7.5}$$

where $p(y_t|x_{t,K_t}^{(i,j)})$ is the likelihood of the final peak based on its correlation response map and $\mathcal{X}_{t-1}^{j}$ is the set of weights of $\mathcal{N}(z_{t-1}^{(j)}, \sigma^2)$ that converge to $x_{t,K_t}^{(i,j)}$. This approach allows us to refrain from unnecessarily resampling the particles at every frame. Instead, we perform resampling only when the effective sample size $\hat{N}_s$ is lower than a threshold N_{thr} according to [11]

$$\hat{N}_s = \frac{1}{\sum_{i=1}^{N} \omega_t^{(i)^2}}, \tag{7.6}$$

where if $\hat{N}_s <= N_{thr}$, resampling is then performed.

7.1.2 Target state estimation

Using Eq. 7.5 in simple frames that do not involve any challenging scenario results in particle weights very similar to one another. Again, this is because the particles converge to a few nearby peaks after the iterations. Hence, the correlation maps related to these final peaks are similar as well. Thus, evaluation of the particles based on likelihoods calculated from the correlation maps is not sufficiently accurate in such simple frames. However, after the iterations, it is possible to determine the location to where most particles converge. As Fig. 7.3 illustrates, our initial particles gradually converge to a few peaks at the end of the iterative refinement procedure. The plots in the middle column of the figure show how the iterations decrease the area covered by the particles. As the plots indicate, after the iterations, the particles reach a sharp posterior distribution from a wide initial distribution. The plots in the left column of the figure show that the weights based on the correlation maps are similar to one another after the iterations. As seen in the bottom right plot, which shows the weights based on Eq. 7.5, the weight of the peak located exactly on the ground

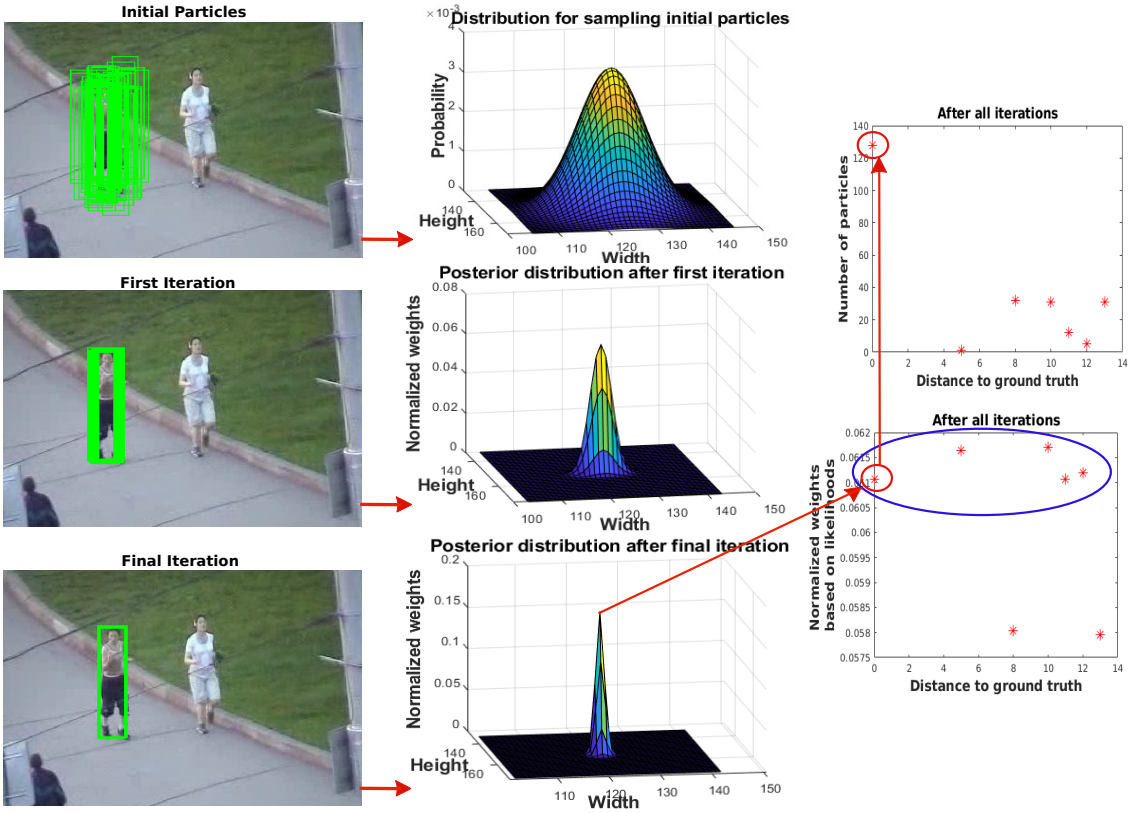


Figure 7.3: The images in the left column show how our initial particles converge to the target after the iterations. The plots in the middle column illustrate how the particles reach a sharp final posterior distribution from the wide initial sampling distribution after the iterations. In the right column, the plots show the normalized weights and the number of particles converging to the peaks after the iterations.

truth is slightly lower than the weights of farther peaks (shown within the blue ellipse). However, most particles converge to the peak closest to the ground truth location as shown in the plot at the top of the right column. Thus, the final state x_t^* is calculated by

$$x_t^* = \arg\max_{x_{t,K_t}^{(i,j)}} J_{t,K_t}, \tag{7.7}$$

where J_{t,K_t} is the number of particles $x_{t,0}^{(i,j)}$ that converge to the common final peak $x_{t,K_t}^{(i,j)}$.

However, when the tracker faces a challenging scenario, the area covered by the particles does not necessarily decrease after the iterations. This is because the particles

may converge to different image regions. As seen in Fig. 7.4, after the iterations, the particles converge to the pole and the jogger, which correspond to two clusters of particles in this challenging scenario. In such scenarios, we first determine the number of clusters and select the one that best represents the posterior. Since the particles converge to distinct image regions, our proposed method can partition them using K-means clustering [52]. We determine the number of modes in the distribution using simplified silhouette analysis based on the Euclidean distances among particles [95]. The plots in the middle column of Fig. 7.4 illustrate how the particles form a posterior distribution with two sharp modes from the wide initial sampling distribution. This posterior distribution is calculated based on the particle weights according to Eq. 7.5. As seen in the top plot of the right column of Fig. 7.4, the number of particles converging to each cluster is not sufficiently accurate to find the image region corresponding to the target. As the plot indicates, only a few particles converge to the region surrounding the jogger. The bottom plot of the right column of the figure shows that the weights calculated by Eq. 7.5, on the other hand, provide an accurate method to distinguish the clusters. Because of the considerable distance between the clusters, the correlation response maps within different clusters are not similar to each other. Thus, particle evaluation based on the likelihoods according to Eq. 7.5 is reliable because correlation maps closer to the target generate higher likelihoods. Thus, we first find the clusters using K-means after performing the iterations, and the mode of each cluster is then selected based on the number of particles reaching the final peaks. The best cluster is then selected based on the correlation response maps corresponding to each mode according to Eq. 7.5. Algorithm 7.3 summarizes our mode clustering and target state estimation method.

7.2 Results and Discussion

We evaluate our algorithm on two publicly available visual tracking benchmarks: the large-scale single object tracking benchmark (LaSOT) [4] and the visual tracker bench-

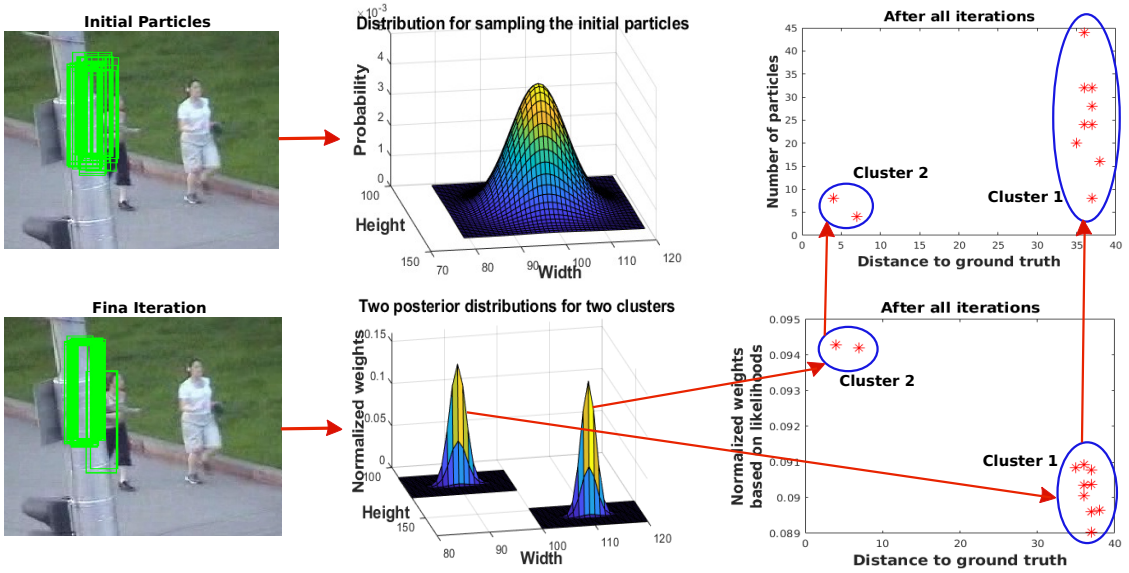


Figure 7.4: The images in the left column show that the initial particles reach two clusters after the iterations in a challenging scenario. The plots in the middle column illustrate that the two clusters correspond to two sharp posterior distribution modes from the wide initial sampling distribution after applying K-means to the normalized particle weights. The right column shows that the weights are more reliable for distinguishing the clusters than the number of particles converging to their modes.

mark v1.1 beta (OTB100) [6]. All the results shown in this section correspond to a particle filter with 200 particles.

Algorithm 7.3 Target State Estimation.

Input: Final peaks $x_{t,K_t}^{(i,j)}$ at time t, their weights $\omega_{t,K_t}^{(i,j)}$ and J_{t,K_t}
Output: Current target state x_t^*

1: Apply K-means to all final current peaks $x_{t,K_t}^{(i,j)}$ to find the clusters
2: Find the mode of each cluster based on J_{t,K_t}
3: Compare the weight of the calculated modes based on $\omega_{t,K_t}^{(i,j)}$ to select the best mode
4: Consider the best mode as the current target state x_t^*

7.2.1 LaSOT evaluation

Fig.s 7.5 and 7.6 present precision and success plots of a quantitative assessment of our proposed approach using a one-pass evaluation (OPE) on LaSOT in comparison with 10 state-of-the-art trackers including ECO, ASRCF [26], DSiam [27], CFNet [28], HCFT [44], BACF [29], CSRDCF [30], SRDCF [40], LCT [31] and KCF [32]. In the one-pass evaluation, the tracker is initialized with the ground truth location of the target at the first frame of the image sequence and is allowed to keep track of the target over the remaining frames without reinitialization. As seen in Fig.s 7.5 and 7.6, our tracker outperforms all the other trackers in terms of overall precision and success. In particular, it outperforms ASRCF by 1.2% and 2.3%, respectively. Similar to our proposed tracker, ASRCF is a recent state-of-the-art correlation-convolutional visual tracker that uses ECO as a baseline method. Our most significant improvements in comparison with ASRCF occur in low resolution and scale variation scenarios, which show improvements of 2% and 1.6% in precision and 1.4% and 2.4% in success, respectively. In comparison with our baseline tracker, our precision improvement reaches 9.1%, 12.0%, and 11.5% in low resolution, scale variation, and partial occlusion scenarios, respectively. In terms of the success metric, our improvement in such scenarios reaches 7.5%, 8.8%, and 10.7% in comparison with ECO.

7.2.2 OTB100 evaluation

Fig.s 7.7 and 7.8 illustrate precision and success plots of a quantitative OPE assessment of our proposed approach in comparison with eight state-of-the-art trackers whose results in the OTB100 dataset are publicly available: ASRCF, ECO, MDNet [36], HDT [43], HCFT, FRDCFdecon [33], CREST [34] and CNN-SVM [35]. On the overall evaluation of the precision and success metrics considering the entire dataset, our tracker shows improvements of approximately 0.6% and 1.6% in comparison with the second best tracker

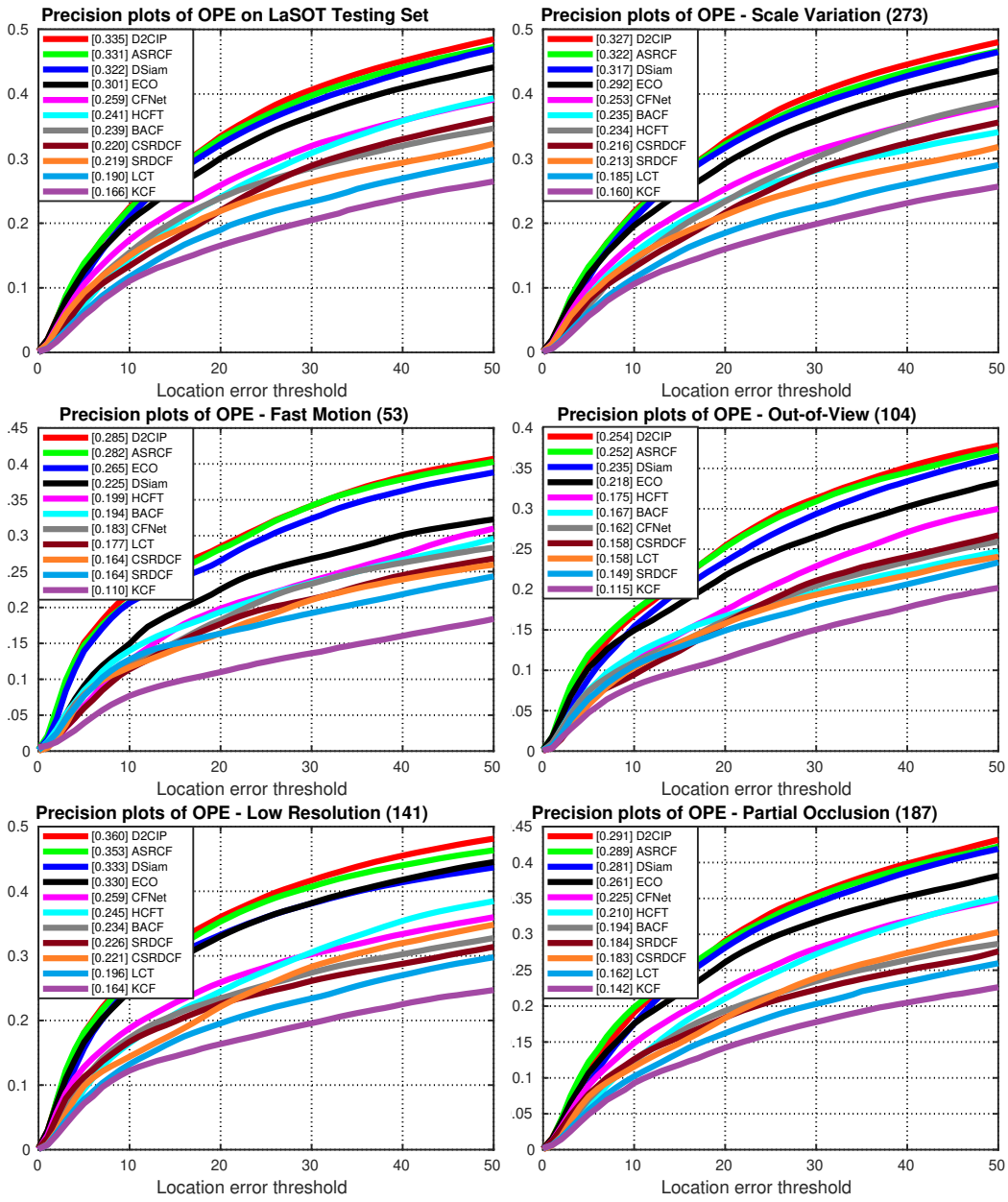


Figure 7.5: Quantitative assessment of the performance of our tracker in comparison with state-of-the-art trackers using a one-pass evaluation and precision plots on the LaSOT benchmark dataset.

ASRCF. Our precision and success improvements in comparison with ASRCF reach 3.6% and 4.6% in fast motion scenarios, 1.9% and 4.4% in scale variation scenarios, 1.5% and 1.8% in out of view conditions, 2.5% and 3.9% when illumination variations are present

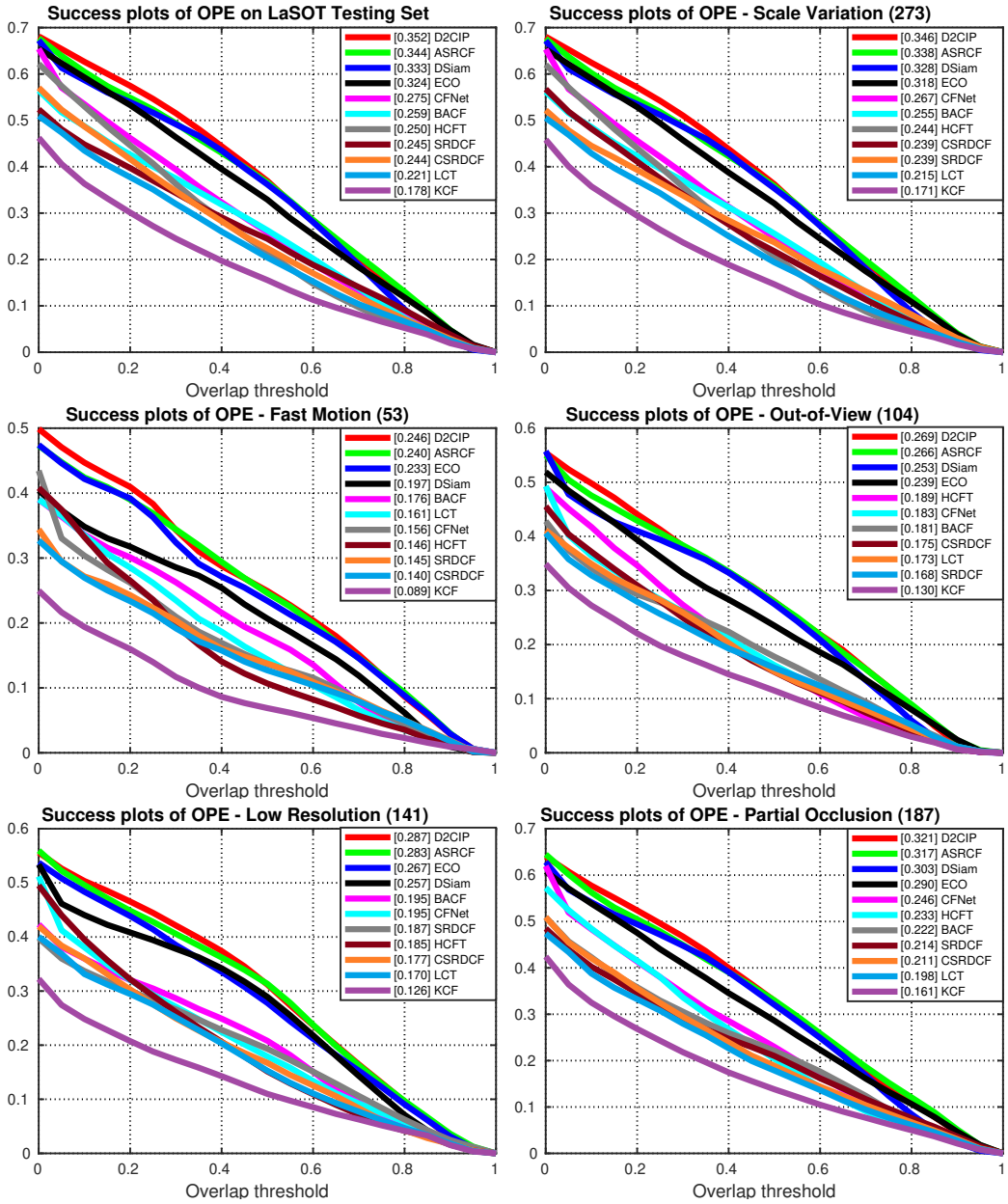


Figure 7.6: Quantitative assessment of the performance of our tracker in comparison with state-of-the-art trackers using a one-pass evaluation and success plots on the LaSOT benchmark dataset.

and 4.4% and 1.9% in scenes including occlusion. In some challenging scenarios in the OTB100 dataset, ASRCF is outperformed by ECO and MDNet. ECO is the second best tracker in the success metric for fast motion, scale variation, and illumination variation sce-

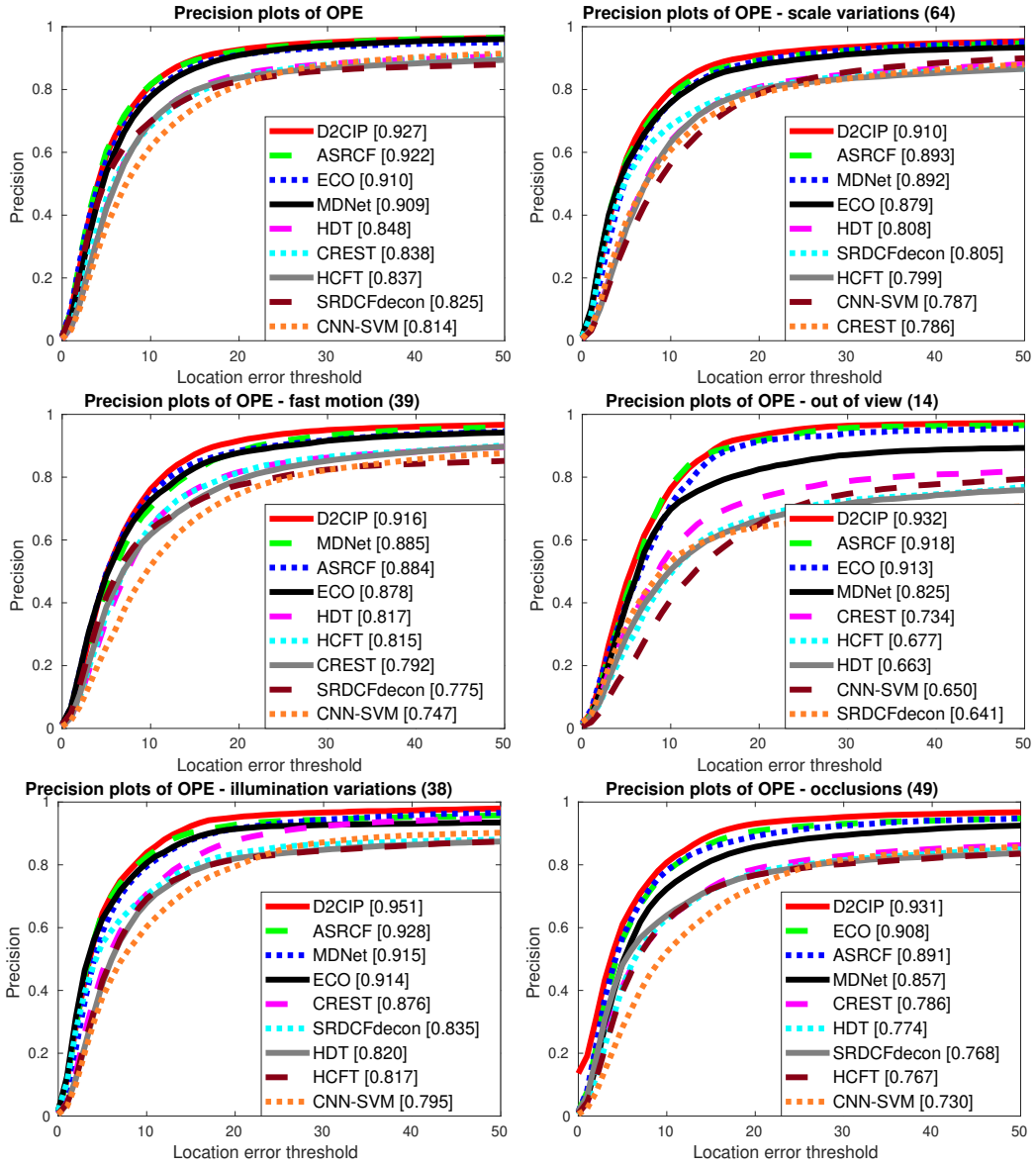


Figure 7.7: Quantitative performance assessment of our tracker in comparison with eight state-of-the-art trackers using a one-pass evaluation and precision plots on the OTB100 benchmark dataset.

narios, as well as in both metrics for occlusion scenarios. Our performance improvements with respect to ECO in these scenarios are 2.8%, 3.8%, 3.4%, 2.5%, and 1.8%, respectively. MDNet only outperforms ASRCF in the precision metric for fast motion scenarios. In that case, our performance improvement with respect to MDNet is 3.5%.

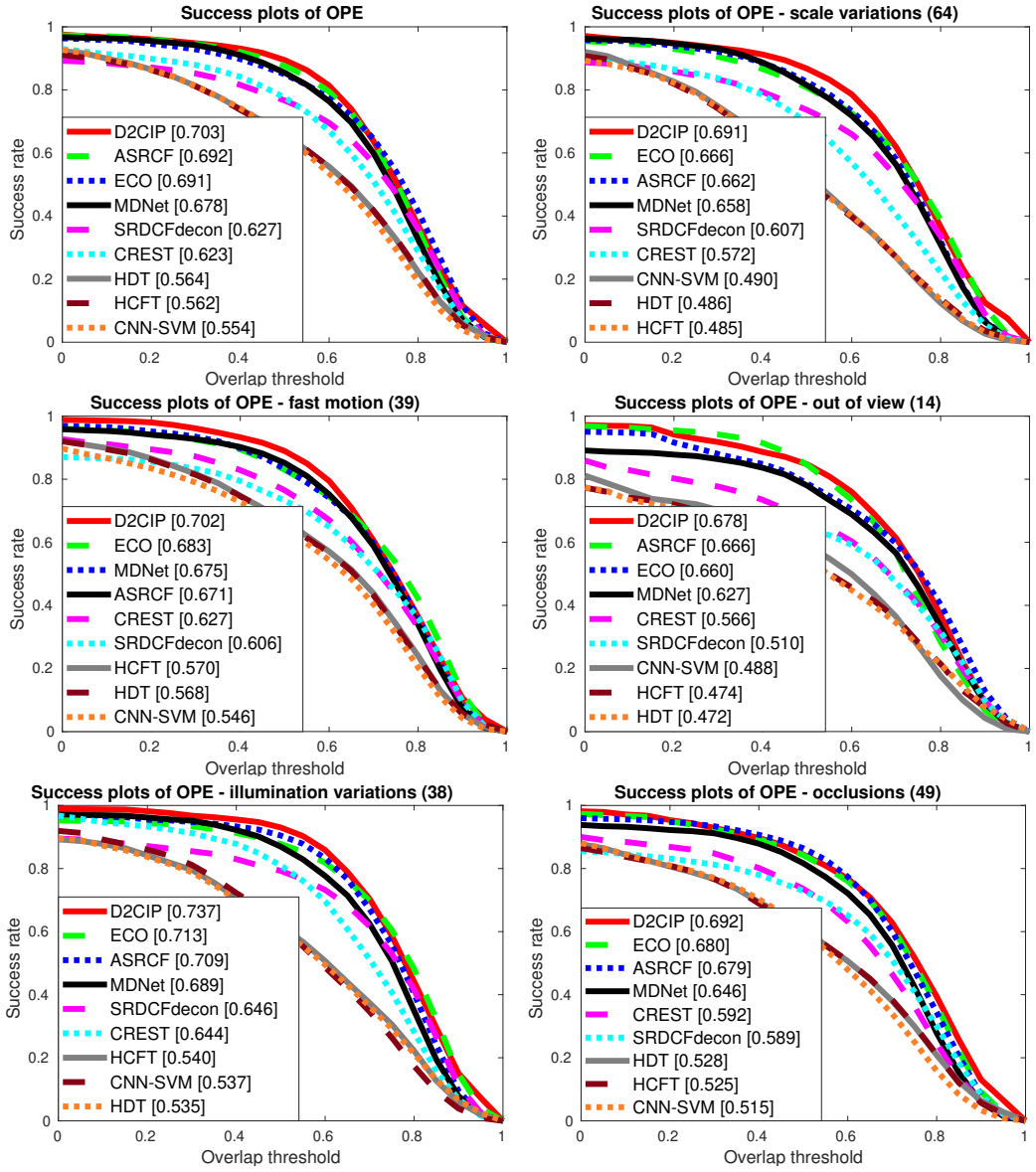


Figure 7.8: Quantitative performance assessment of our tracker in comparison with eight state-of-the-art trackers using a one-pass evaluation and success plots on the OTB100 benchmark dataset.

Fig. 7.9 presents a qualitative assessment of our tracker in comparison with ASRCF, ECO, and HCFT. In the first row, the other trackers fail because of a relatively long occlusion period, which causes not only tracking loss but also incorrect model updates. Our tracker, on the other hand, finds multiple potential clusters during partial occlusion

Figure 7.9: Qualitative evaluation of our tracker in comparison with *ASRCF*, *ECO* and *HCFT* on three challenging sequences of the OTB100 (top row, *Girl2* sequence) and LaSOT datasets (middle and bottom rows, *Train-1* and *Goldfish-4* sequences, respectively).

and maintains one model per cluster. When the target becomes visible again, this enables our tracker to sample from distributions closer to the target and assess the corresponding updated models. In the second row, the other trackers fail because of a period of fast motion by the target. Our particle filter enables our tracker to sample particles over a wider search area and the iterative particle refinement allows it to reach the best location for the target. In the third row, the presence of several similar objects causes failures in the other trackers, whereas our novel method for evaluating the particle likelihoods locates the correct target accurately.

Table 7.1: Ablative analysis of the components of the proposed visual tracker. PF corresponds to the integration of the baseline tracker with the particle filter model. IPF incorporates the iterative particle refinement procedure. IPFK includes the clustering of the likelihood distributions. D2CIP corresponds to our complete algorithm, which further incorporates maintaining one target model for each predicted mode. The numbers within parentheses indicate the relative performance gains over the baseline tracker.

Benchmark	PF	IPF	IPFK	D2CIP	Precision	Success
LaSOT	✓	✗	✗	✗	30.9% (+2.6%)	33.1% (+2.3%)
	✓	✓	✗	✗	32.3% (+7.3%)	34.5% (+6.4%)
	✓	✓	✓	✗	33.2% (+10.4%)	34.9% (+7.9%)
	✓	✓	✓	✓	33.5% (+11.3%)	35.2% (+8.7%)
OTB100	✓	✗	✗	✗	91.5% (+0.5%)	69.6% (+0.7%)
	✓	✓	✗	✗	92.2% (+1.3%)	69.9% (+1.2%)
	✓	✓	✓	✗	92.6% (+1.8%)	70.2% (+1.6%)
	✓	✓	✓	✓	92.7% (+1.9%)	70.3% (+1.7%)

7.2.3 Ablative analysis

Table 7.1 shows the performance improvement introduced by the different components of our proposed method and compares them with ECO [69], our baseline tracker. As seen in Table 7.1, our proposed method improves the performance of ECO by 11.3% in terms of precision and 8.7% on the success metric for LaSOT. Since the OTB100 dataset contains less complex sequences, the performance of the baseline tracker is already relatively high for both metrics. Nonetheless, our tracker still provides up to 1.9% and 1.7% relative improvements in precision and success.

CHAPTER 8
ADAPTIVE TARGET MODEL UPDATE USING SHORT-TERM MEMORY

To increase the robustness of the method presented in the previous chapter to model drift, we propose a new model update strategy based on the confidence scores of the correlation response maps. Inspired by the approach proposed in [96], we employ the confidence score vector in a finite state machine that assigns three different states to the tracking algorithm: *target found*, *partially lost*, and *fully lost*. The target model is updated based on the state of the tracker. In the *partially lost* state, our tracker then employs a short-term memory mechanism to save previous models, including the model corresponding to the last frame in which the tracker was in the *target found* state. No model update is performed in the *fully lost* state. We evaluated the performance of our tracker on OTB100 [6] and LaSOT [4] and observed significant improvement against state-of-the-art methods.

8.1 Proposed Algorithm

After selecting the best particle for the current frame using our iterative particle filter as explained in Section 7.1.2, we use its correlation response map to determine the target state. We calculate its mean and negative entropy according to

$$\mu_t = \frac{1}{M \times Q} \sum_{m=1}^{M} \sum_{q=1}^{Q} R^{*}_{t(m,q)}, \tag{8.1}$$

$$E_t = \frac{1}{M \times Q} \sum_{m=1}^{M} \sum_{q=1}^{Q} R^{*}_{t(m,q)} \log(R^{**}_{t(m,q)}), \tag{8.2}$$

where $R^{*}_{t(m,q)} \in \mathbb{R}^{M \times Q}$ represents the correlation response map corresponding to the best particle and $R^{**}_{t(m,q)}$ is its normalized version. We then compute the moving weighted average of $G_k = [\mu_k, E_k]^T$ over the last β frames

$$W_t = \frac{\sum_{k=t-\beta-1}^{t} G_k k}{\sum_{k=t-\beta-1}^{t} k}. \tag{8.3}$$

Finally, as proposed in [96] the confidence score vector of the current frame $S_t = [S_{\mu_t}, S_{E_t}]^T$ is given by

$$S_t = \frac{W_t - G_t}{(W_t + G_t)/2}. \tag{8.4}$$

We use this confidence score vector in a finite state machine which includes three different states: *target found*, *partially lost*, and *fully lost*. The finite state machine is initialized in the first frame in the *target found* state. The corresponding confidence scores are updated only in the *target found* state while the weighted averages are updated only in the two other states. The reason behind this design is that we do not want the frames with weak correlation response map affect on the expected mean and entropy [96]. The confidence scores of the current frame S_{μ_t} and S_{E_t} are compared with a mean threshold μ_{tr}, and two entropy thresholds E_{pt} and E_{ft}, which determine the transitions to the partially lost and fully lost states. If

$$S_{\mu_t} > \mu_{tr} \vee S_{E_t} > E_{pt}, \tag{8.5}$$

the tracker remains in the *target found* state, meaning that the confidence score is sufficiently high to allow the tracker to operate normally. In this scenario, the best particle is selected as the target state of the current frame, and the model is updated using the method described in Section 7.1.1.2. When the following conditions are satisfied

$$S_{\mu_t} \leq \mu_{tr} \wedge S_{E_t} \leq E_{pt} \wedge S_{E_t} > E_{ft}, \tag{8.6}$$

the tracker transitions to the *partially lost* state, meaning that the tracker has lower confidence on its estimate because part of the target might be occluded or the target appearance temporarily changed because of fast motion or other challenging scenarios. In this state, the size and position of the target are determined by the correlation response map of the best particle as in the *target found* state. However, once the tracker transitions to this state, we initiate a short-term memory mechanism to update its correlation filter based on models generated in previous frames.

Figure 8.1: Using the model of frame before starting the partially lost state caused by a partial occlusion helps to find the target after finishing the partially lost state.

Fig. 8.1 illustrates a scenario where the target is temporarily partially occluded, which puts the tracker in the *partially lost* state. In that case, our short-term memory mechanism uses the best model obtained in the last frame in which the tracker was in the *target found* state to update the correlation filter. This model allows us to find the target once it is no longer occluded. In other challenging scenarios, such as those caused by fast target motion, we need a different approach. By incorporating all of the best models obtained over the previous β in *partially lost* state, we are able to keep track of the target until its appearance returns to normal. Fig. 8.2 illustrates our short-term memory method. At time t, we evaluate the target likelihood using the N models generated for all high-likelihood particles at $t-1$, as in the *target found* state. Additionally, we evaluate the best model obtained in the *target found* state (frame $t-6$ in the figure) and the best models from the $\beta = 3$ previous frames in *partially lost* state. In the example shown in Fig. 8.2, the model generated at time $t-3$ is the most similar to the target because it contains the least amount of motion blur. Algorithm 8.2 explain our short-term memory.

In scenarios where the confidence score of the correlation response map is significantly lower than average, i.e., when the following conditions are satisfied

$$S_{\mu_t} \leq \mu_{tr} \wedge S_{E_t} \leq E_{pt} \wedge S_{E_t} \leq E_{ft}, \tag{8.7}$$

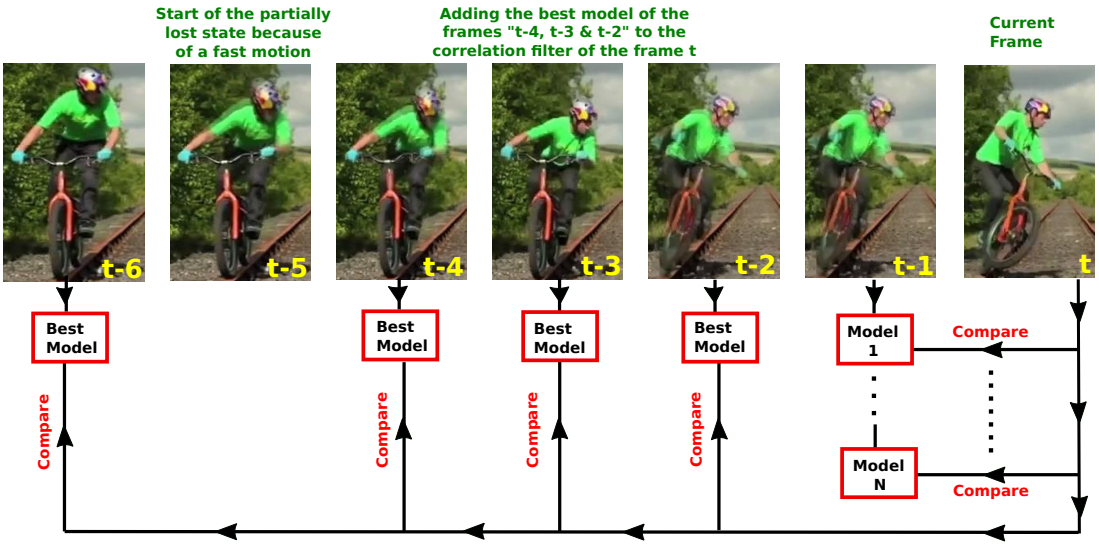


Figure 8.2: Illustration of the short-term memory mechanism used in the *partially lost* state.

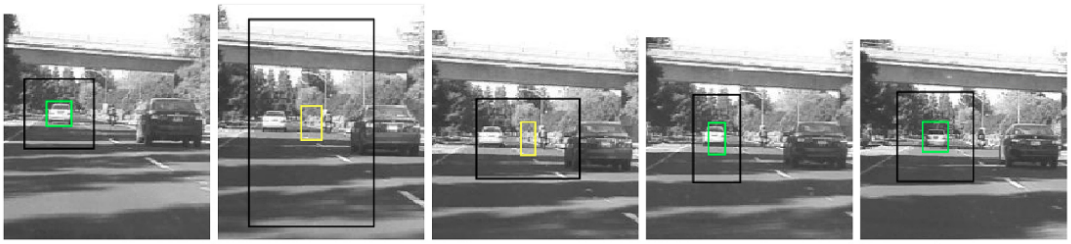


Figure 8.3: Illustration of how the short-term memory mechanism helps tracking. Black, green and yellow squares show the search areas, target found and partially lost states.

the target is likely lost due to complete occlusion or other challenging scenarios. In that case, tracker transitions to the *fully lost* state. In this state, model updates are not performed to avoid model drift. In addition, since no information is available regarding the likelihood of the target, the tracker considers the previous target state as its current estimate. Algorithm 8.1 explain our proposed framework.

Algorithm 8.1 Confidence Score Computation

Input: Correlation response map R_t^* and target state x_t^* of the best particle of the current frame t, previous target state x_{t-1} and previous correlation filter

Output: Final target state x_t of the current frame t, method of generating the target models used in the next frame $t+1$ and search area size W_s

Compute the confidence score according to Eq. 8.1 to Eq. 8.4
if tracker state = target found based on Eq. 8.5 **then**
    Find x_t and update the model as normal
end if
if tracker state = partially lost based on Eq. 8.6 **then**
    Find x_t as normal but use short-term memory to update the models based on Algorithm 8.2
end if
if tracker state = fully lost based on Eq. 8.7 **then**
    Use x_{t-1} as x_t and stop updating the models
end if
Determine the search area size W according to Eq. 8.8 to Eq. 8.10

Figure 8.4: Illustration of the three states of the tracker. Black, green, yellow and red squares show the search areas, target found, partially lost and fully lost states.

The confidence score vector can also be used to scale the search area size W_s. That is, the lower the confidence of the tracker on its estimated target position, the larger the search area it should consider. Again, similar to the strategy proposed in [96], we define the search window size as

$$W_s = W_h \left[\left(\frac{\sqrt{E_n \mu_n}}{2} \right) S_t^T T_r \right] + W_l, \tag{8.8}$$

where $T_r = [\mu_t^{-1}, E_{ft}^{-1}]$, E_n and μ_n are the normalized constants such that the following relationship holds

$$0 \leq S_t^T T_r \leq 1, \tag{8.9}$$

and W_h and W_l determine the acceptable range of search area sizes such that

$$W_l \leq W_s \leq W_l + W_h. \tag{8.10}$$

Algorithm 8.2 Short-term Memory

Input: Previous model set, best model generated in frame $t-1$ and estimated tracker state in frame t

Output: Updated model set to be employed in frame $t+1$

```
Generate the models for all high likelihood particles in frame t
if previous tracker state = fully found then
    add the best model of frame t − 1 to our model set
end if
if previous tracker state = partially lost then
    Add the best model of frame t − 1 to our model models
    if the best model of frame t − 4 is available in our model set then
        Remove it
    end if
end if
```

8.2 Results and Discussion

We evaluate our algorithm on OTB100. In this visual tracker, we use HCFT [44] as the baseline of our framework instead of ECO [69] used in the previous chapter. Based on the experiments, we consider $\mu_{tr} = 0.28$, $E_{pt} = 0.17$, $E_{ft} = 0.57$, $W_l = 1$ and $W_h = 3.5$. In Fig 8.5, we provide a quantitative evaluation of our proposed approach (DCPF_3) on a one-pass evaluation (OPE) in comparison with 9 state-of-the-art trackers: CREST [34], CFNet-conv3 [90], SiameseFC [91], SINT [92], FRDCFdecon [33], LCT [31] and CNN-SVM [35], HDT [43] and HCFT [44]. On the overall evaluation of the

precision and success plots, our tracker shows improvements of 4.5% and 3.5% in comparison with HDT, the second-best tracker in the precision metric and SRDCFdecon, the second-best tracker according to the success metric. In comparison with the well-known HCFT, the corresponding improvements are 5.5% and 15%. For the challenging scenarios of fast motion, deformation, low resolution, background clutter and out-of-plane rotation, our tracker shows improvements of approximately 7%, 6.5%, 6%, 4.5% and 4%, respectively, in comparison with the second-best tracker.

In Fig. 8.3, we further illustrate how the short-term memory contributes to our tracker's performance. After a tracking failure, the short-term memory mechanism helps the tracker to find the correct target and resume tracking it. In Fig. 8.4 we provide an example that incorporates all three states of the algorithm. Using the previous position and stopping the model update in the *fully lost* states help the tracker refrain from following the man instead of the girl.

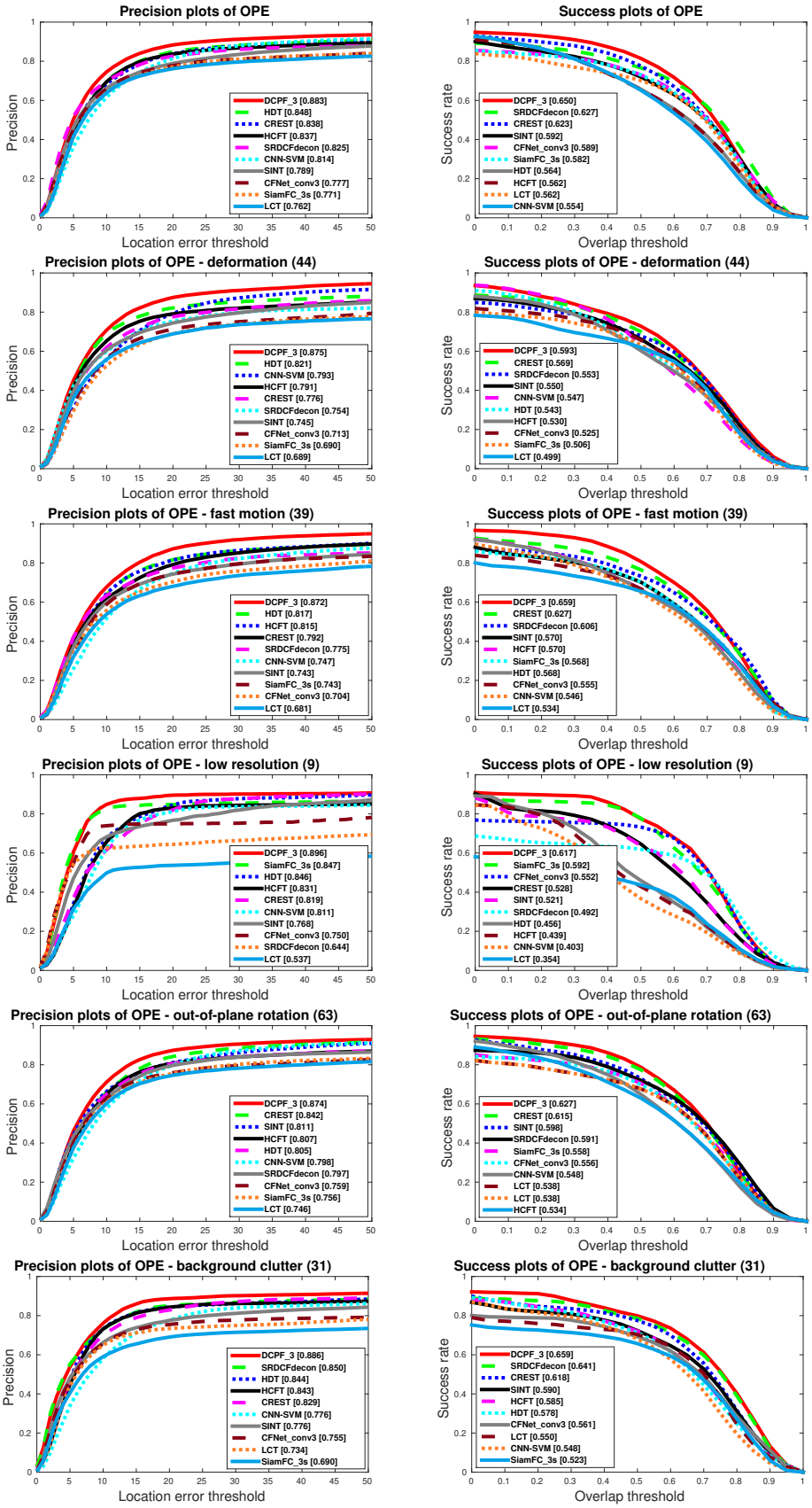


Figure 8.5: OPE quantitative evaluation of our tracker in comparison with nine state-of-the-art trackers on OTB100.

CHAPTER 9
CONCLUSION

This final chapter summarizes the main findings and contributions of this dissertation and discusses some possible future research directions related to this dissertation.

9.1 Summary

This dissertation provides several contributions to improve the performance of visual tracking algorithms. All six visual tracking techniques proposed in this dissertation resort to the integration of a novel particle filter, a deep convolutional neural network, and a correlation filter. The visual tracking algorithm described in Chapter 1 is one of the first visual trackers to use particle filters in conjunction with deep learning techniques. Sequential Monte Carlo methods such as particle filters have been used to incorporate temporal information and hence develop robust object tracking algorithms that build on features that range from simple color histograms to support vector machines. It is only natural to integrate such temporal robustness with recent state-of-the-art, highly discriminative feature extraction techniques such as correlation filter-based deep convolutional neural networks. We use the response maps generated by a correlation filter for each particle to calculate its corresponding weight. However, our initial framework, similar to other correlation visual trackers, can not find the target sizes. Thus, we extend our particle filter to sample particles for both the location and size of the target.

The main drawback of particle filters is their computational cost. So, in Chapter 5, we propose an adaptive particle filter which decrease the number of the particles especially in simple frames where locating the target is not challenging. We then present a likelihood particle filter which estimates a likelihood distribution as the proposal density for a particle filter based on correlation response maps. Correlation response maps provide an initial estimate of the target location, which results in more accurate particles. Furthermore, the

resulting likelihood distribution has a wider variance in challenging scenarios such as in the presence of fast motion and motion blur. In our fifth approach, we propose an iterative particle filter which refines the particles by considering the peak estimated from the correlation response map as a new particle. This process continues until the distance between the particle and its peak is smaller than a threshold. This iterative procedure leads most particles to converge to only a few final positions. By iteratively updating the particle likelihoods, our method also addresses the problem of calculating the posterior distribution over the correct support points in particle-correlation trackers.

This dissertation also provides a number of contributions towards the improvement of the target model generated by correlation filters. In our second visual tracker, we find all of the high-likelihood particles and calculate a model for each of them, because the common correlation filters are heavily dependent upon the estimated target position. In our third approach, we generate multiple target models in each frame by applying different update rates to the models created by the high-likelihood particles. For challenging frames, we use lower update rates, which means we rely more on previous target models. In Chapter 6, we present a mechanism based on a Gaussian mixture model that allows our algorithm to accommodate multi-modal likelihoods by clustering particles according to the distribution of their correlation response maps. Our particle filter then generates a likelihood distribution for each correlation map cluster in difficult scenarios such as those involving target occlusions. We then introduce an iterative approach that allows the particles to converge to a small number of locations closer to the actual target position in Chapter 7. In our sixth strategy, discussed in Chapter 8, We define a confidence score calculated from the correlation response map of the best particle to overcome challenging scenarios such as those involving occlusion, fast motion, or target deformation. We use the confidence score vector in a finite state machine comprised of three different states: *target found*, *partially lost*, and *fully lost*. In the *partially lost* state, a short-term memory is used to save target models from

previous frames in *partially lost* state and the model of the last frame in *found state*. In the *fully lost* state, the target models is not updated.

The evaluation of our proposed strategies on different benchmarks demonstrates that our methods substantially outperform several state-of-the-art techniques.

9.2 Future work

The recursive Bayesian estimation methods and models presented in this dissertation shows that sequential Monte Carlo methods can be successfully used to improve the performance of visual tracking algorithms. However, as is common in such frameworks, several parameters of our algorithms must be determined empirically. Identifying strategies to determine these parameters automatically is a promising direction for future research. In particular, more effective mode finding strategies such as mean-shift [97] might allow us to better accommodate multi-modal likelihoods while resorting to fewer heuristic strategies to determine the number of modes in the distribution. As previously mentioned, computational cost is also a drawback in particle filters. Improving the likelihood models using smaller but more accurate deep neural networks would allow us to reduce the number of particles needed by our algorithms. Recently, many new visual tracking benchmarks have been proposed such as [98]. These benchmark datasets could be used for training the networks to improve the accuracy of our likelihood models. Our short-term memory mechanism explained in Chapter 8 can be combined with our another contribution in applying different adjusting rates explained in Chapter 5. For example, variable target model update rates could be used according to the state of the tracker. We can also define new tracker states that account for the different modes of the likelihood distribution. This strategy would allow us to account for rapid target appearance changes without allowing the model to drift because of potentially confusing background regions.

BIBLIOGRAPHY

[1] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, "What your images reveal: Exploiting visual contents for point-of-interest recommendation," in *26th International World Wide Web Conference, WWW 2017*, 2017, pp. 391–400.

[2] K. Patel. (2020) Architecture comparison of alexnet, vggnet, resnet, inception, densenet. [Online]. Available: https://towardsdatascience.com/architecture-comparison-of-alexnet-vggnet-resnet-inception-densenet-beb8b116866d

[3] T. Li, S. Sun, T. P. Sattar, and J. M. Corchado, "Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches," *Expert Systems with Applications*, vol. 41, pp. 3944–3954, 2014.

[4] H. Fan, L. Lin, Y. Fan, P. Chu, G. Deng, S. Yu, and H. Bai, "LaSOT: A high-quality benchmark for large-scale single object tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] M. Ma, H. Fan, and K. M. Kitani, "Going deeper into first-person activity recognition," in *International Conference on Computer Vision (ICCV)*, 2016.

[6] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[7] J. Janai, F. GÃŒney, A. Behl, and A. Geiger, *Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art*. Now Foundations and Trends, 2020.

[8] G. Sreenu and M. A. Saleem Durai, "Intelligent video surveillance: a review through deep learning techniques for crowd analysis," *Journal of Big Data*, vol. 6, no. 48, 2019.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR (Presented at International Conference on Learning Representations, 2015)*, vol. abs/1409.1556, 2014.

[11] C. Bishop, *Pattern recognition and machine learning*. New York, NY, USA: Springer, 2006.

[12] E. Alpaydin, *Introduction to machine learning*. Cambridge, MA, USA: MIT Press, 2014.

[13] L. Li Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, pp. 197–387, 2014.

[14] X. Wang, Y. Zhao, and F. Pourpanah, "Recent advances in deep learning," *International Journal of Machine Learning and Cybernetics*, vol. 11, pp. 747–750, 2020.

[15] S. N. Gowda, "Human activity recognition using combinatorial deep belief networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[16] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Young Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[17] Q. Wang, C. Yuan, J. Wang, and W. Zeng, "Learning attentional recurrent neural network for visual tracking," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 930–942, 2019.

[18] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[19] Y. Liu, J. Han, Z. Qiang, and S. Caifeng, "Deep salient object detection with contextual information guidance," *IEEE Transactions on Image Processing*, vol. 29, pp. 360–374, 2020.

[20] T.-Y. Lin, P. Dollar, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, 2016.

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015, pp. 1–10.

[23] A. Siddique, R. J. Mozhdehi, and H. Medeiros, "Deep heterogeneous autoencoder for subspace clustering of sequential data," *arXiv:2007.07175*, 2020.

[24] F. S. Bashiri, E. LaRose, J. C. Badger, R. M. D'Souza, Z. Yu, and P. Peissig, "Object detection to assist visually impaired people: A deep neural network adventure," in *13th International Symposium on Visual Computing (ISVC)*, 2018.

[25] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Conference on Neural Information Processing Systems (NIPS)*, 2013.

[26] K. Dai, D. Wang, L. Huchuan, C. Sun, and J. LI, "Visual tracking via adaptive spatially-regularized correlation filters," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[27] Q. Guo, W. Feng, C. zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *International Conference on Computer Vision (ICCV)*, 2017.

[28] J. Valmadre, L. Bertinetto, J. F. Henriques, and P. H. S. Vedaldi, Andrea Torr, "End-to-end representation learning for correlation filter based tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[29] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *International Conference on Computer Vision (ICCV)*, 2017.

[30] A. Lukezic, T. Vojir, L. Cehovin, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5388–5396.

[32] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[33] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[34] Y. Song, C. Ma, L. Gong, J. Zhang, R. Lau, and M.-H. Yang, "CREST: Convolutional residual learning for visual tracking," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2555 – 2564.

[35] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *International Conference on Machine Learning (ICML)*, 2015.

[36] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[37] H. Fan and H. Ling, "SANet: Structure-aware network for visual tracking," in *CVPR Workshop on DeepVision: Temporal Deep Learning*, 2017.

[38] J. Choi, H. J. Chang, J. Jeong, Y. Demiris, and J. Y. Choi, "Visual tracking using attention-modulated disintegration and integration," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[39] M. Tang and J. Feng, "Multi-kernel correlation filter for visual tracking," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 3038–3046.

[40] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[41] K. Dai, D. Wang, H. Lu, C. Sun, and J. Li, "Visual tracking via adaptive spatially-regularized correlation filters," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4670–4679.

[42] M. Zhang, Q. Wang, J. Xing, J. Gao, P. Peng, W. Hu, and S. Maybank, "Visual tracking via spatially aligned correlation filters network," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 469–485.

[43] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M. H. Yang, "Hedged deep tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 4303–4311.

[44] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[45] R. J. Mozhdehi and H. Medeiros, "Deep convolutional particle filter for visual tracking," in *IEEE International Conference on Image Processing (ICIP)*, 2017.

[46] H. Fu, Y. Zhang, Z. Wuneng, W. Xiaofeng, and H. Zhang, "Learning reliable-spatial and spatial-variation regularization correlation filters for visual tracking," *Image and Vision Computing*, vol. 94, p. 103869, 2020.

[47] P. M. Raju, D. Mishra, and P. Mukherjee, "DA-SACOT: Domain adaptive-segmentation guided attention for correlation based object tracking," *Image and Vision Computing*, p. 104215, 2021.

[48] R. J. Mozhdehi, Y. Reznichenko, A. Siddique, and H. Medeiros, "Deep convolutional particle filter with adaptive correlation maps for visual tracking," in *IEEE International Conference on Image Processing (ICIP)*, 2018.

[49] R. J. Mozhdehi and H. Medeiros, "Deep convolutional correlation iterative particle filter for visual tracking," *arXiv preprint arXiv:2107.02984*, 2021.

[50] R. J. Mozhdehi, Y. Reznichenko, A. Siddique, and H. Medeiros, "Convolutional adaptive particle filter with multiple models for visual tracking," in *13th International Symposium on Visual Computing (ISVC)*, 2018.

[51] R. J. Mozhdehi and H. Medeiros, "Deep convolutional likelihood particle filter for visual tracking," in *24th International Conference on Image Processing, Computer Vision, & Pattern Recognition (IPCV)*, 2020.

[52] C. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[53] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[54] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[55] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[56] P. Ambrozio Dias, "Stochastic methods for fine-grained image segmentation and uncertainty estimation in computer vision," Ph.D. dissertation, Marquette University, 2020.

[57] V. V. Bhagavatula and V. Naresh Boddeti, "Advances in correlation filters: vector features, structured prediction and shape alignment," Ph.D. dissertation, Carnegie Mellon University, 2012.

[58] M. Danelljan, F. S. Khan, M. Felsberg, and J. v. d. Weijer, "Adaptive color attributes for real-time visual tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[59] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, 2002.

[60] T. Zhang, S. Liu, and C. Xu, "Correlation particle filter for visual tracking," *IEEE Transactions on Image Processing*, vol. 27, no. 6, pp. 2676–2687, 2018.

[61] H. Li, Y. Li, and F. Porikli, "Deeptrack: Learning discriminative feature representations online for robust visual tracking," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1834–1848, April 2016.

[62] A. Shinde, A. Sahu, D. Apley, and G. Runger, "Preimages for variation patterns from kernel PCA and bagging," *IIE Transactions*, vol. 46, no. 5, pp. 429–456, 2014.

[63] H. Li, Y. Li, and F. Porikli, "Convolutional neural net bagging for online visual tracking," *Computer Vision and Image Understanding*, vol. 153, pp. 120–129, 2016.

[64] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, W. R. Lau, and M.-H. Yang, "Vital: Visual tracking via adversarial learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[65] K. Chen and W. Tao, "Once for all: a two-flow convolutional neural network for visual tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, September 2017.

[66] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015.

[67] P. Li, D. Wang, L. Wang, and H. Lu, "Deep visual tracking: Review and experimental comparison," *Pattern Recognition*, vol. 76, pp. 323–338, 2018.

[68] M. Danelljan, A. Robinson, F. Shahbaz Khan, and M. Fels-berg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *European Conference on Computer Vision (EECV)*, 2016.

[69] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[70] N. Wang, Q. Zhou, R. Tian, R. Hong, M. Wang, and H. Li, "Multi-cue correlation filters for robust visual tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[71] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang, "Learning spatial-temporal regularized correlation filters for visual tracking," in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2018, pp. 4904–4913.

[72] Z. Zhu, W. Wu, W. Zou, and J. Yan, "End-to-end flow correlation tracking with spatial-temporal attention," in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2018, pp. 548–557.

[73] C. Sun, H. Wang, H. Lu, and M.-H. Yang, "Correlation tracking via joint discrimination and reliability learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[74] F. Du, P. Liu, W. Zhao, and X. Tang, "Joint channel reliability and correlation filters learning for visual tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 6, pp. 1625–1638, 2020.

[75] T. Zhang, C. Xu, and M.-H. Yang, "Multi-task correlation particle filter for robust object tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[76] T. Zhang, S. Liu, and C. Xu, "Correlation particle filter for visual tracking," *IEEE Transactions on Image Processing*, vol. 27, no. 6, pp. 2676–2687, 2018.

[77] D. Yuan, X. Lu, D. Li, Y. Liang, and X. Zhang, "Particle filter re-detection for visual tracking via correlation filters," *Multimedia Tools and Applications*, vol. 78, no. 11, pp. 14 277–14 301, 2019.

[78] Z. Fan, H. Ji, and Y. Zhang, "Iterative particle filter for visual tracking," *Image Communication*, vol. 36, pp. 140–153, 2015.

[79] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[80] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901

[81] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *International Conference on Machine Learning (ICML)*, 2015.

[82] J. Choi, H. J. Chang, J. Jeong, Y. Demiris, and J. Y. Choi, "Visual tracking using attention-modulated disintegration and integration," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[83] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[84] W. Zhong, H. Lu, and M. H. Yang, "Robust object tracking via sparse collaborative appearance model," *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2356–2368, 2014.

[85] X. Jia, "Visual tracking via adaptive structural local sparse appearance model," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2012, pp. 1822–1829.

[86] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1269–1276.

[87] T. B. Dinh, N. Vo, and G. Medioni, "Context tracker: Exploring supporters and distracters in unconstrained environments," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[88] J. Kwon and K. M. Lee, "Tracking by sampling trackers," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 1195–1202.

[89] H. Y. Cheng and J. N. Hwang, "Adaptive particle sampling and adaptive appearance for multiple video object tracking," *Signal Processing*, vol. 89, no. 9, pp. 1844–1849, 2009.

[90] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-end representation learning for correlation filter based tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[91] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional Siamese networks for object tracking," in *ECCV 2016 Workshops*, 2016, pp. 850–865.

[92] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[93] T. Kawabata, "Multiple subunit fitting into a low-resolution density map of a macromolecular complex using a gaussian mixture model," *Biophysical Journal*, vol. 95, no. 10, pp. 4643–4658, 2008.

[94] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp.

53–65, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377042787901257

[95] F. Wang, H.-H. Franco-Penya, J. D. Kelleher, J. Pugh, and R. Ross, "An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity," in *International Conference on Machine Learning and Data Mining in Pattern Recognition*, 2017, pp. 291–305.

[96] R. Walsh and H. Medeiros, "Detecting tracking failures from correlation response maps," in *Advances in Visual Computing: 12th International Symposium (ISVC)*, 2016, pp. 125–135.

[97] M. Trassinelli and P. Ciccodicola, "Mean shift cluster recognition method implementation in the nested sampling algorithm," *Entropy*, vol. 22, no. 2, 2020.

[98] M. Mullert, A. Bibit, S. Giancolat, S. Alsubaihi, and B. Ghanem, "Trackingnet: A large-scale dataset and benchmark for object tracking in the wild," in *European Conference on Computer Vision (ECCV)*, 2018.

CHAPTER 10
COPYRIGHT

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of Marquette University's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to `http://www.ieee.org/publications_standards/publications/rights/rights_link.html` to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.