Synthesis Lectures on Computer Vision

SYNTHESIS
COLLECTION OF TECHNOLOGY

Ning Xu · Weiyao Lin ·
Xiankai Lu · Yunchao Wei

Video Object Tracking

Tasks, Datasets, and Methods

Springer

Synthesis Lectures on Computer Vision

This series publishes on topics pertaining to computer vision and pattern recognition. The scope follows the purview of premier computer science conferences, and includes the science of scene reconstruction, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, and image restoration. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems, such as those in self-driving cars/navigation systems, medical image analysis, and industrial robots.

Ning Xu · Weiyao Lin · Xiankai Lu ·
Yunchao Wei

Video Object Tracking

Tasks, Datasets, and Methods

Springer

Ning Xu
Adobe Research
San Jose, CA, USA

Xiankai Lu
School of Software
Shandong University
Jinan City, China

Weiyao Lin
Department of Electronic Engineering
Shanghai Jiao Tong University
Shanghai, China

Yunchao Wei
School of Computer and Information
Technology
Beijing Jiaotong University
Beijing, China

ISSN 2153-1056 ISSN 2153-1064 (electronic)
Synthesis Lectures on Computer Vision
ISBN 978-3-031-44659-7 ISBN 978-3-031-44660-3 (eBook)
https://doi.org/10.1007/978-3-031-44660-3

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

Preface

With the universal access of cameras in all kinds of devices (e.g., mobile phones, surveillance cameras, and in-vehicle cameras), video data has gone through an exponential increase nowadays. For a lot of video-related applications such as autonomous driving, video editing, and augmented reality, segmenting target objects is important to help understand video content.

Video object tracking (VOT) is a fundamental task for video understanding in computer vision. VOT can be divided into different settings further, which are slightly different in the definition and focus areas. In this book, we will give an throughout introduction of the task of VOT (including different settings). In Chap. 1, we first briefly introduce the settings of VOT. Then in Chap. 2, we introduce the VOT task under three most common problem settings which are single-object tracking, multi-object tracking, and multi-object tracking and segmentation. Finally, in Chap. 3, we give introduction to our held HiEve challenge on VOT.

USA
September 2022

Ning Xu

Contents

Introduction

1

The ubiquity of cameras in various devices (e.g., mobile phones, surveillance cameras, in-vehicle cameras) has led to an exponential increase in video data. For a lot of video-related applications, being able to understand, localize, and track target objects is one of the most fundamental and important problems. For example, in the field of autonomous driving, precise localization of surrounding cars and pedestrians is crucial for vehicles to avoid collisions and take safe actions. Additionally, in the field of augmented reality (AR), to attach AR effects to some moving object in the space and to make the effect look realistic, we also need some algorithm to follow the motion of the target object in real-time.

Video object tracking (VOT), a fundamental problem for video understanding in computer vision, is the underlying technology that enables the above applications. In a formal definition, the task of VOT aims at producing tight bounding boxes around one or multiple target objects in the video. Meanwhile, VOT focuses on the ability to track target objects over a long period of time in videos. However, it will face big challenges when objects have fast motion, large appearance changes, similar instances, and heavy occlusion etc.

This book provides a comprehensive introduction to VOT. Specifically, in Chap. 2, we introduce the VOT task under three most common problem settings: single-object tracking, multi-object tracking, and multi-object tracking and segmentation. Each problem setting is first elaborated with a formal problem definition, followed by presenting well-known challenges, the most popular datasets and evaluation metrics. Then we overview different types of methods that have been proposed for the problem. Finally, we select two or three most representative and effective methods and dive deep into their idea details as well as experiments. Additionally, we also include the results of the recent impactful competition (HiEve Challenge) on VOT in Chap. 3, offering readers the methods and results of top-performing teams.

N. Xu et al., *Video Object Tracking*, Synthesis Lectures on Computer Vision,
https://doi.org/10.1007/978-3-031-44660-3_1

Table 1.1 Comparison between VOT tasks on several attributes

Task	Classification	Tracking	Detection	Segmentation	Object num
VOT					
SOT	×	√	×	×	Single
MOT	×	√	√	×	Multiple
MOTS	×	√	√	√	Multiple

Before diving into each individual task in the following chapters, we first give an overview of each task and let readers understand the goal of each task and the difference among them.

- Single object tracking (SOT) is a VOT setting where the target's bounding box in the first frame is given and algorithms need to predict the target position in the subsequent frames.
- Multi-object tracking (MOT) can be easily distinguished from SOT because it requires to extract the spatial and temporal trajectories of multiple moving objects from the video.
- Multi-object tracking and segmentation (MOTS) is a recently proposed tracking task that requires detecting, tracking, and segmenting objects belonging to a set of given classes from the video.

Besides the above differences, in Table 1.1 we list other attributes that can further understand and distinguish these tasks. First, all these VOT tasks do not require addressing the classification problem (i.e., recognizing object categories). Second, each task involves a different combination of sub-problems. For instance, the SOT and MOT do not need to solve the segmentation problem while the MOTS need. Lastly, the two tasks MOT and MOTS need to handle multiple instances in the video, while only the SOT focuses on a single target object.

Tracking

2

In this chapter, we will elaborate on the task of video object tracking (VOT), which aims at producing tight bounding boxes around one or multiple target objects in the video. The VOT task has different problem settings given different input or output requirements. Single object tracking (SOT) Sect. 2.1 gives the bounding box of object that needs to track in the first frame and requires to predict the whole tracklet of this object. Multi-object tracking (MOT) Sect. 2.2 requires to detect the objects first and then get their tracklets. Multi-object tracking and segmentation (MOTS) Sect. 2.3 need to predict segmentation further.

2.1 Single Object Tracking

2.1.1 Introduction

In this section, we introduce single object tracking, that given the target's bounding box in the first frame and predicting the target position in the subsequent frames. Video object tracking is one of the important tasks in computer vision, and has a wide range of applications in real life, such as video surveillance, visual navigation, etc. Video target tracking tasks also face many challenges, such as target occlusion, target deformation and so on. In order to solve the challenges in target tracking and achieve accurate and efficient target tracking, a large number of target tracking algorithms have appeared in recent years.

This section introduces the basic principles, improvement strategies and representative work of the two mainstream algorithm frameworks in the field of video target tracking (target tracking algorithm based on correlation filtering and Siamese network) in the past ten years. The target tracking algorithm also introduces typical solutions to various problems from the perspective of solving the challenges faced by target tracking, and summarizes the historical development and future development trend of video target tracking. This section also introduces and compares the datasets and challenges for object tracking tasks in

N. Xu et al., *Video Object Tracking*, Synthesis Lectures on Computer Vision,
https://doi.org/10.1007/978-3-031-44660-3_2

detail, and summarizes the characteristics and advantages of various video object tracking algorithms based on the data statistics of the datasets and the evaluation results of the algorithms.

SOT has been widely studied with the development of computer vision history. First, this setting is easy and flexible for many real-world applications, such as human trajectories construction in public security surveillance systems, navigation in Autonomous driving and unmanned aerial vehicles, Motion Trajectory Capture and Active Tracking in Robotics and pose tracking in human-object-interaction. In the late 90s, *generative methods* are the mainstream solution for SOT [166]. These methods are under the geometry-based methods with the strong assumption, such as optical flow (TLD) [85, 90], Kalman Filter [153], partial filter [3], Meanshift [46] with hand-craft feature, such as SIFT and SURF.

From the 2000s, *learning based* SOT methods become the mainstream [193] with the development of machine learning. These methods can be categorised into two classes: shallow learning and deep learning. Shallow learning ones combine the classical machine learning algorithms, such as SVM [68, 184], ensemble learning [7, 231], sparse coding [223] and correlation filter [75]. Tian et al. [184] considered tracking as a binary classification problem, and SVM is selected as the main classification. In object tracking, the target area defined as positive data, and the surrounding environment is defined as negative data. The goal is to train an SVM classifier that can classify positive and negative data into new frames.

In the deep learning era, early methods explored fully convolution network or deep correlation filter for target prediction [134, 172, 173, 194]. The tracking pipeline still follow into the traditional tracking paradigm by using the first frame annotation to generate many samples for network finetuning. Then, Siamese architectures [15, 54, 109, 110, 183, 227] or variants of recurrent neural networks [67] becomes the popular tracking methods. These methods tend to train a fully-convolution siamese networks with annotated videos and apply the fixed weight model to the unseen video directly [179]. With the development of graph neural network and Transformer, some researchers have apply these powerful relation modeling method for SOT, for instance, STARK [212], Keeptrack [139], TransT [26] and CSWinTT [175].

In the rest of this section, we will first introduce the common challenges of this task Sect. 2.1.2, popular benchmarks and evaluation metrics Sect. 2.1.3 as well as overview of common and effective methods Sect. 2.1.4. Then we present three representative SOT methods in detail which belong to Correlation filter methods Sect. 2.1.5, one-stage deep tracking methods Sect. 2.1.6, Siamese network methods Sect. 2.1.7, respectively.

2.1.2 Challenges

In this section we discuss a few challenging problems in SOT. As a classical task in the computer vision, SOT faces lots of challenges:

- *Appearance variation*. Appearance change is a common interference problem in target tracking. When the appearance of a moving target changes, its characteristics and appearance model will change, which may easily lead to tracking failure. For example: athletes in sports competitions, pedestrians on the road due to the fast motion and view changes.
- *Scale variation*. Scale adaptation is also a key issue in target tracking. When the target scale is reduced, since the tracking frame cannot be adaptively tracked, a lot of background information will be included, resulting in an update error of the target model: when the target scale When it increases, since the tracking frame cannot completely include the target, and the target information in the tracking frame is incomplete, the update error of the target model will also be caused. Therefore, it is very necessary to achieve scale adaptive tracking.
- *Occlusion and disappearance*. The target may be occluded or disappear briefly during the movement. When this happens, the tracking frame will easily include the occlusion clutter and background information in the tracking frame, which will lead to tracking in subsequent frames. The target drifts to the clutter. If the target is completely occluded, the tracking will fail because the corresponding model of the target cannot be found.
- *Other factors* Motion blurring Changes in light intensity, fast movement of the target, low resolution, etc. will lead to image models, especially when the moving target is similar to the background. Therefore, select effective features to distinguish the target from the background. Very necessary.

Based on these discussion, we can see that robust SOT method need to satisfy the following criteria: the learned feature representation should be variance for different scenarios, especially for occlusion case meanwhile be adaptive for appearance change. We will give more detailed explanation in the method sections.

2.1.3 Dataset and Metric

This section will detail popular public dataset as well as common evaluation metrics. Considering different tracking datsets have different evaluation metrics. Therefore, we introduce each dataset with corresponding evaluation metrics.

OTB (object tracking benchmark) is a old and classical dataset [207] which contains two version: OTB100 (100 videos) and OTB50 (50 videos). The videos are labeled with 11 attributes, namely: illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, inplane rotation, out-of-plane rotation, out-of-view, background clutters, and low resolution. There are two widely used metrics for OTB datasets: distance precision and overlap success that are computed per-frame and across video on average:

$$P^t = \|C_{tr}^t - C_{gt}^t\|_2 \quad S^t = \frac{BB_{tr} \cap BB_{ft}}{BB_{tr} \cup BB_{ft}} \tag{2.1}$$

VOT (visual object tracking) [103] is a popular tracking challenge that is hold on every year since 2013. Each year would update the benchmark, therefore, there are more than 10 benchmarks until now. There are 3 representative benchmarks, i.e. VOT-2014, VOT-2017 and VOT-2020. The recent released VOT-2022 challenge has 5 tracks: VOT-STs: RGB short-term segmentation tracking challenge, VOT-STb: RGB short-term bounding box tracking challenge (NEW), VOT-RT: RGB short-term tracking challenge, VOT-LT: RGB long-term bounding box tracking challenge, VOT-RGBD: RGB+depth bounding box tracking challenge. Also, each video in the VOT is labeled with the 6 attributes: camera motion, illumination change, occlusion, scale change, motion change and neutral attributes, respectively.

There are three metrics are used to report tracking performance: Robustness (Rob), Accuracy (Acc) and Expected Average Overlap (EAO). The robustness is the number of times the tracker failed, i.e., drifted from the target, and had to be reinitialized. The average robustness can be computed as:

$$\rho = \frac{1}{N_{rep}} \sum_{k=1}^{N_{rep}} F(k), \tag{2.2}$$

where $F()$ denotes the failure number.

The accuracy is defined as the intersection-over-union (IOU) between the prediction and the ground-truth:

$$\Phi_t = \frac{A_t^G \cap A_t^T}{A_t^G \cup A_t^T} \tag{2.3}$$

where A_t^T denotes the prediction while A_t^G denotes the ground-truth.

The EAO measures the expected no-reset overlap of a tracker run on a short-term sequence:

$$\bar{\Phi} = \frac{1}{N_{hi} - N_{lo}} \sum_{N_s = N_{lo}:N_{hi}} \bar{\Phi}_{N_s} \tag{2.4}$$

where Φ_* denotes the results in the expected average overlap curve.

UAV123 This dataset contains 123 video sequences and more than 110K frames from UAV [145]. It was captured using cameras mounted on drones and includes all bounding boxes and attribute annotations. Some tracking targets belong to the small object and undergo long term occlusion. The evaluation metrics are same as OTB.

TempleColor128 contains 128 videos that are specifically designated to evaluate color-enhanced trackers [120]. The videos are labeled with 11 similar attributes as in OTB. The evaluation metrics are same as OTB.

GOT-10k [84] contains more than 10,000 video segments of real-world moving objects and over 1.5 million manually labeled bounding boxes. Also, this dataset is backboned by WordNet and it covers a majority of 560+ classes of real-world moving objects and 80+ classes of motion patterns. The evaluation metrics are same as OTB.

Nfs4 Nfs4 [60] is the first higher frame rate benchmark for visual object tracking. This dataset includes 100 videos of 380K frames, with 75 videos recorded using the iPhone and iPad Pro and 25 videos taken from YouTube. All videos are manually labeled with 9 visual attributes: occlusion, illumination variation, scale variation, object deformation, fast motion, viewpoint change, out of view, background clutter, and low resolution.

TrackingNet TrackingNet [147] is a large-scale benchmark for tracking in the wild. It provides more than 30K videos with more than 14 million dense bounding box annotations. 15 attributes are provided for the testing set. In addition to the precision and success metric used in OTB2015, TrackingNet introduced another metric called normalized precision (NPre):

$$P_{norm} = \|W(C_{tr} - C_{gt})\|_2, W = diag(BB_x^{gt}, BB_x^{tr}), \tag{2.5}$$

where *diag* denotes diagonal matrix.

LaSOT This benchmark dataset is the largest annotated one in the tracking community [55], consisting of 280 long videos sequences. This dataset contains 123 video sequences with 70 object categories selected from 1000 classes of ImageNet. Sequences labeled with 14 attributes, including illumination variation, full occlusion, partial occlusion, deformation, motion blur, fast motion, scale variation, camera motion, rotation, background clutter, low resolution, viewpoint change, out-of-view and aspect ratio change. The evaluation metric of this benchmark is the same as the TrackingNet.

Besides, there are some infrared object tracking datasets, such as LSOTB-TIR [126] and VTUAV [221].

A comprehensive comparison of different datasets is show in Table 2.1.

2.1.4 Overview of Methods

Object tracking need to describe target effectively. Then find the area with the largest similarity to the target in each frame of the next video frame, so as to determine the position of the target in the current frame. According to using whether hand-crafted features or not, the development of SOT can be categorized into non-deep-learning based and deep-learning based methods. Although recent development on SOT mainly focus on deep-learning based methods, it is meaningful to review some non-deep-learning based methods as their key ideas are still very useful to understand the problem.

Non-Deep-Learning Based Methods

The development of non-deep learning based methods can be divided into two pres: generative based solutions and discriminative based solutions.

Table 2.1 Comparison among existing datasets. "Annotations" denotes the total number of object annotations. "Duration" denotes the total duration (in minutes) of the annotated videos. "FPS" denotes the frame rate

Dataset	Number	Frames	Average len	Total len	Class	Label	FPS
OTB-2015	100	59 K	590	32.8 mins	16	11	30 fps
TC-128	128	55 K	429	30.7 mins	27	11	30 fps
VOT-2018	60	14687	40	–	–	–	20 fps
UAV123	20	113 K	915	62.5 mins	9	12	30 fps
NfS	100	383 K	3,830	26.6 mins	17	9	240 fps
OxUvA	366	1,550 K	4,599	14.4 h	22	6	30 fps
TrackingNet	30,643	14,431 K	498	141.3 h	27	15	30 fps
LaSOT	1,400	3,520 K	2,506	32.5 h	70	14	30 fps
GOT-10k	10 K	1.5 M	15	41.7 h	563	6	10 fps
RGBT234	234	234 K	1,000	130 mins	–	11	30 fps

Generative Trackers

The generative appearance models mainly concentrate on how to accurately fit the data from the object class using the Bayes' theorem. With the use of a set of tracked results for describing the target appearance, generative trackers are effective in recovering target objects from tracking failures; however, since these methods neglect the contextual cues, they usually fail to track target objects with cluttered background, especially in the presence of distracters. The early generative methods mainly include two ideas: (1) Depends on the target appearance model. By modeling the target appearance model, and then finding the target in the subsequent frames. For example: optical flow method. (2) Do not depend on the target appearance model.

- **Optical flow** based method. The concept of optical flow method (Lucas-Kanade) was first proposed in 1950. It operates on the pixels in the video sequence for the appearance model. By using the pixel relationship between adjacent frames of the video sequence, finding the displacement change of the pixels Determine the motion state of the target and realize the tracking of the moving target.
  In [125], Harris corner detection and optical flow method are combined, and the pixels in the optical flow method are replaced by Harris feature points for target tracking, which reduces the number of tracking pixels as well as the complexity of the algorithm, and accelerate the processing speed. At the same time, compared with ordinary pixels, Harris corners describe the target better and have strong robustness. In [161], the concept of artificial optical flow was first proposed, the real optical flow was compared with the artificial optical flow, and the motion of the target was judged by the vector difference.

The computation of optical flow is slow, also is sensitive to the occlusion.

- **Kernel Method** The kernel based methods involve to Meanshift and Camshift [50] algorithms, which directly use the principle of the steepest descent method, gradually iterate the target template in the direction of gradient descent until it reaches the optimal position. Its core is to iteratively find the best point. During tracking, these methods find the candidate with the largest similarity degree value.

$$\begin{aligned} M_h(x) &= \frac{1}{t} \sum_{x_i \in S_h} (x_i - x) \\ S_h(x) &= \{y : (y-x)^\top (y-x) \leq r^2\} \end{aligned} \tag{2.6}$$

As shown in the first line of Eq. 2.6, Meanshift first averages the vector difference between the sample point and the center point. Then, the direction of the vector sum is the direction of increasing probability density, move the vector along the direction of increasing probability density, and iterate step by step until the optimal solution is found. However, this search method has drawback, it only the computes the sampled points, and the contribution of all points is the same regardless of the distance from the center point. When the target is occluded or motion blurred, the features of the outer layer are easily affected by the background thus leading to accuracy reduce.

- **Sparse coding based method**
  Sparse representation [205], which is also well known as compressive sensing, has drawn considerable attention in the tracking community in the last decade. Mei and Ling [142] first introduce the sparse representation framework for visual object tracking. The target is represented by a linear combination of a set of target templates and trivial templates. For each candidate, the sparse coefficient is computed by solving an L1 regularized least square object. A candidate with smaller reconstruction error is more likely to be the target. The sparse coding tracker shows promising performance in handling heavy occlusion as the trivial templates model noise and occluded pixels favorably. To further accelerate the computational efficiency of the sparse coding tracker, Bao et al. [9] exploit the reconstruction error bound of L2 norm to exclude some inferior particle samples. This scheme efficiently accelerates the L1 tracker.
  Furthermore, Zhang et al. [224] introduce multiple task learning into sparse coding.

Discriminative trackers

Unlike generative trackers, discriminative trackers usually consider tracking as a classification problem that discriminates the target from the background. Here, we review several representative and classical object tracking methods: SVM based methods, multiple instance learning based methods and correlation filter based method.

- **SVM** Support vector machines (SVMs) have long been applied for a wide range of computer vision problems [117]. One representative work is SVT (support vector tracking) [6,

184] algorithm by exploiting both the offline trained SVM classifier and the optical flow method. In contrast to previous optical flow trackers that minimize the displacement loss function between consecutive frames, SVT iteratively maximizes the SVM classification score in a coarse-to-fine fashion. By taking the advantages of temporal smoothness from optical flow and the discriminative strength from SVM classifiers, the SVT performs well in tracking moving vehicles.

Then, milestone work is Struck [68]. Struck presents a framework for adaptive visual object tracking based on structured output prediction. Through explicitly allowing the output space to express the needs of the tracker, struck is able to avoid the need for an intermediate classification step. For model training, instead of using binary labels for SVM training, struck considers the spatial distribution of samples as labels.

- **Multiple instance learning** MIL belongs to the scope of boosting learning. Numerous efforts have been made to improve the tracking performance using the idea of boosting learning. For instance, Collins et al. [32] introduced online feature selection for tracking, where in each frame the most discriminative features are chosen to compute likelihoods. One representative work is ensemble tracking [184], it takes a similar approach by combining a small number of weak classifiers using AdaBoost. Sometimes, single AdaBoost classifier can not be sufficient to capture multi-modal data. Therefore, one approach has been to cluster the data and train separate classifiers. Grabner et al. [66] build a tracker upon AdaBoost and the ensemble tracking framework.

  Lately, in [7], to handle the label ambiguity in object tracking, MIL is introduced into object tracking framework. The MIL tracker not only takes advantages of traditional boosting methods in feature selection, but also overcomes the sampling ambiguity of assigning spatial overlapped samples with binary labels by collecting all the promising positive samples into a single positive bag.
- **Randomized Learning based Trackers**

  These methods tend to use ensemble learning method, such as random forest for tracking. Kalal et al. [90] decompose the tracking task into tracking, learning and detection, where tracking and detection facilitates each other, i.e., the results from the tracker provide training data to update the detector, and the detector re-initializes the tracker when it fails. Also, thet employ P-N learning to update the tracker.

  This strategy has been widely used for long-term tracking, especially for tracking failure cases [178].
- **Correlation Filter**

  The target tracking algorithm based on correlation filtering (CF) theory has made remarkable progress in target tracking tasks, and has obvious advantages in tracking accuracy and running speed. It is one of the mainstream frameworks for video target tracking tasks in recent years. The application of CF in computer vision can be traced back to the 1980s, and it was first used in the field of object detection. In 2010, the MOSSE algorithm [19] applied CF to the video target tracking task for the first time, and achieved excellent accuracy and super speed at that time. The good performance of correlation filtering

theory in target tracking is mainly due to the following two reasons: (1) The CF target tracking method implicitly uses the circular translation operation to augment the training samples, thus greatly enriching the diversity of training samples , which improves the robustness and accuracy of the algorithm; (2) Fast Fourier Transform (FFT) accelerates the calculation of complex convolution operations in the frequency domain, reduces the amount of calculation, and increases the efficiency of model solving. The CSK [196] method builds on illumination intensity features and is further improved by using HOG features in the KCF [74] tracking algorithm.

The basis theory of CF for object tracking can be summarized as follows [74]:

Step 1 search area: Since the moving range of the target in the adjacent two frames is limited, the tracking position of the t–1th frame is used to obtain by appropriately expanding t–1 target search area, and perform target window search within the above search area of in the t–th frame image of the video.

Step 2 feature extraction: Based on the search area in the t-th frame obtained in step 1, CF performs feature extraction on the image in this area to obtain the feature map H.

Step 3 localization: CF applies the filter F on the feature map H to obtain the response map C:

$$C = H * F, \tag{2.7}$$

where $*$ denotes the Circulant convolution. The maximum response location of map C corresponds to the target position. Then, based on the target location in the current frame, CF performs image region crop.

Step 4 CF model updating: Using the current tracking result, take the target as the center point to intercept the sub-image, extract the feature map similar to step 2, and then solve the correlation filter by minimizing the object function:

Step 5 Repeat Step 1–4 until the last frame.

$$E(F) = \|X * F - Y\|^2 + \|F\|^2. \tag{2.8}$$

Here, Y is a 2-D Gaussian distribution map with the highest value at the center of space. The above optimization problem can be solved closed using the Fast Fourier Transform (FFT) method.

Based on KCF, there are lots of improved algorithms from the feature representation view or algorithm optimization view, we list some representative ones. Danelljan et al. [40] incorporate color features (RGB, LAB) into Color Names (CN) features. Ma et al. [135] propose to combine histogram feature of intensity with HoG feature for tracking. SAMF [119] further concatenated CN, intensity and HOG feature together for object tracking.

Also, to handle the scale variation of the object across temporal dimension, Danelljan et al. [35] presented a joint translation scale tracking method based on a three-dimensional scale spatial correlation filter is proposed. When designing the filter, a one-dimensional scale is added on the basis of the two-dimensional position translation. The two filters

work independently of the target localization and scale estimation, so it is convenient to port this algorithm to other algorithms.

For correlation filter improvement, SRDCF [39] handle the boundary effects caused by enlarging the search window size and introduced a spatial regularization component in the learning to penalize correlation filter coefficients depending on their spatial location. Another strategy is Correlation filters with limited boundaries [95] and Background-Aware Correlation Filters (BACF) [94]. CFLB and BACF directly perform zero-padding operation on the filter edge through the clipping matrix to obtain a correlation filter with a small scope. Furthermore, ASRCF [208] proposed an adaptive spatial regularization algorithm for correlation filtering target tracking. The spatial regularization weight map used by ASRCF can be adaptively and dynamically adjusted according to the characteristics of the target during the tracking process.

Deep-Learning Based Methods

Recently, deep learning has been applied extensively on the SOT task and improved the performance greatly compared to those non-deep-learning-based methods. Based on the deep feature leveraged in the methods, we can categorize them into *deep feature module*, *one-stage*, *Siamese methods* and *Anchor-free methods*. Next we introduce each of the methods in detail.

Deep Learning Module

At the early stage of deep learning era, some researchers made an early effort to replace the feature extraction or classification module with deep neural networks. For example, C-COT (Continuous Convolution Operators for Visual Tracking) [41] utilizes the depth features of multiple convolutional layers and innovatively proposes continuous spatial domain interpolation operations to deal with the problem of different feature resolutions in different convolutional layers. C-COT interpolates the feature map to the continuous domain through the implicit interpolation operation in the frequency domain, which is used to integrate features of different resolutions. The algorithm characterizes and calculates the correlation filter in the continuous domain, and obtains a higher target. positioning accuracy. Meanwhile, Ma et al. proposed HCFT [136] that uses three layers convolution features and train three correlation filters independently.Then, considering the deep features have different spatial resolution and semantic information, HCFT performs coarse-to-fine location from the three response maps generated from the CF. Furthermore, ECO (Efficient Convolution Operators for Tracking) [34] reduces the model parameters by convolution factorization operation, generates a sample space model to reduce the number of samples, and the model sparse update strategy reduces the update frequency, which effectively improves the running speed of the algorithm.

One-stage Trackers

The one-stage regression tracking framework takes the whole search area as input and directly outputs a response map through a regression model, which learns a mapping between input features and the target labels usually generated by a Gaussian function. One-stage regression trackers are typically on the basis of convolutional regression networks. The FCNT [194], STCT [121], GOTURN [73], and CREST [172] trackers belong to this category. The FCNT makes the first effort to learn regression networks over two convolutional layers. The output response maps from different layers are switched according to their confidence to locate target objects. The STCT algorithm exploits ensemble learning to select CNN feature channels. GOTURN [73] uses fully connected layers to directly regress input features to the apexes of the target bounding boxes rather than soft labels. Similar to GOTURN, Gao et al. [63] train an object part level regression network to predict the part center and translation. CREST [172] learns a base network as well as a residual network on the output of last convolutional layer. The output maps of the base and the residual networks are fused to infer target positions. We note that current deep regression trackers do not perform as well as DCFs trackers. We identify the main bottleneck as the data imbalance issue in regression learning. By balancing the importance of training data, the performance of one-stage deep regression trackers can be significantly improved over DCFs trackers.

Siamese Networks

In contrast to the above one-stage regression trackers using soft labels to represent the target states, the one-stage Siamese trackers [15, 187] first binarize the soft labels and then employ a binary cross entropy loss for classification learning. Bertinetto et al. [15] substituted the fully connected layer in GOTURN with a cross correlation layer and proposed a novel fully-convolutional Siamese network, namely SiameseFC, which consists of a template branch and a search branch. The CFNet [187] tracker adds a correlation filter to the template branch, yielding a shallow Siamese tracker. More recently, many works [43, 45, 62, 67, 70, 114, 118, 118, 220, 227] learn to update SiameseFC to adapt appearance changes, such as extra correlation filter [67], target-specific network [114, 118, 220] and graph neural network [62]. Note that the Siamese trackers perform template matching, which requires a large search region as input. This complicates learning classifiers by introducing imbalanced data from the background.

The two-stage classification tracking performs tracking over two steps. The first stage generates a set of candidate target samples around the previously estimated location using random sampling, regularly dense sampling [52, 141, 198, 218], or region proposal [81, 222, 232]. The second stage classifies each candidate sample as the target object or as the background. Numerous efforts have been made to learn a discriminative boundary between positive and negative samples [225]. Examples include the multiple instance learning (MIL) [7] and Struck [68, 152] methods. Recent deep trackers, such as DeepTrack [112], SANet [53] and CNN-SVM [78], all belong to the two-stage classification framework. Recently, the SiamRPN [110] tracker extends the one-stage SiameseFC algorithm with an additional

region proposal network (RPN). SiamRPN first takes the template and search images as input and outputs proposals, based on which SiamRPN outputs classification scores as well as the estimated bounding boxes in one forward pass. Fan et al. [55] further exploited a cascaded RPN to mine multiple scales of bounding boxes across different convolutional layers to improve the discriminative power of the learned classifier. UpdateNet [220] updates the target template in the Siamese network to incorporate the temporal information. Despite the favorable performance on the challenging object tracking benchmarks [100, 207], it is noted that two-stage deep trackers suffer from heavy class imbalance due to a limited number of positive samples in the tracking scenario. Moreover, the positive samples are spatially correlated and redundant.

Anchor-free Networks

Despite region proposal based networks have achieved superior performance, however, these methods also suffer some detection issues. Thus, another common idea is to treat the SOT task as a anchor-free task.

MAML [192] transfers anchor-free object detectors, such as FCOS [185], to trackers via meta-learning and incrementally updates the network weights to adapt the target-special sequences. OceanNet [228] directly predicted the position and scale of target objects in an anchor-free fashion instead of refining the reference anchor boxes. SiamBAN [27] introduced a simple yet effective visual tracking framework by exploiting the expressive power of the fully convolutional network (FCN). SiamBAN classifies objects and regresses their bounding boxes in a unified anchor-free FCN. Recently, Wang et al. [199] utilize the powerful sequential data modeling ability to object tracking task.

2.1.5 Correlation-Filter Tracking

Motivation

In this section, we first introduce an ensemble of correlation filters which are trained with different features and region proposals to address the challenging problems of drifting and scale estimation for robust visual tracking. Specifically, we integrate adaptive region proposals into a two-stream tracking framework. For the tracking stream, we learn two-stage cascade correlation filters to locate targets and to determine whether tracking failures occur. Specifically, we aggressively learn correlation filters on a large searching window with surrounding context for precise localization. With the estimated position, we conservatively learn another correlation filter on the target area to maintain a long-term memory of the target appearance. For the detection stream, we apply the long-term memory filter rather than additional classifiers over instance-aware region proposals to re-detect target objects when tracking failures occur. The ensemble of correlation filters is combined using an optimization function. To handle scale variation, we apply an additional correlation filter over

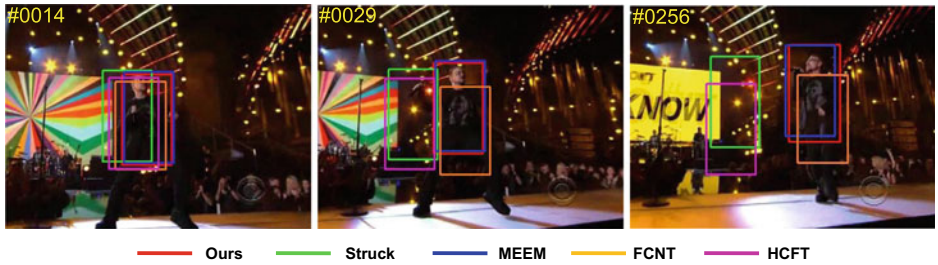


Fig. 2.1 Tracking results on the *Singer2* sequence [207] by our tracker and the Struck [68], MEEM [217], FCNT [194] and HCFT [134] trackers. Our method employs adaptive region proposals for re-targeting the target in the 29th frame and uses scale-aware region proposals to estimate the scale changes in the 256th frame. Our tracker performs favorably against the state-of-the-art methods

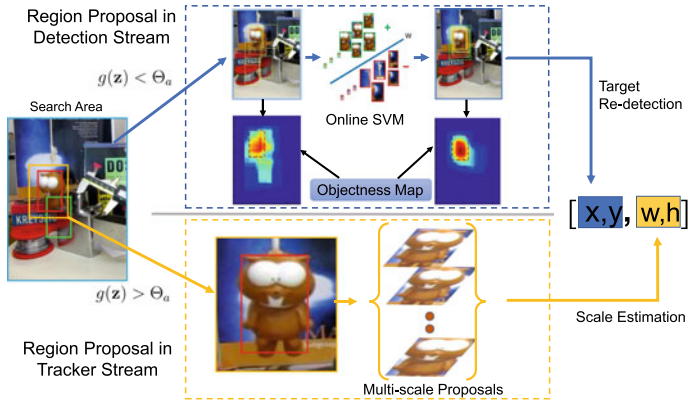


Fig. 2.2 Pipeline of the proposed tracking algorithm. For each tracking result (green), we crop a patch $\mathbf{z}$ centered at the estimated position and use a long-term memory correlation filter to determine whether a tracking failure occurs (e.g., when $g(\mathbf{z})$ is smaller than a given threshold Θ_a); then, we activate detection over instance-aware region proposals to re-detect target objects. Once the confidence score of the tracking result (red) is larger than the threshold Θ_a, we generate scale-aware proposals for scale estimation in a slightly larger window (orange). Region proposals take the objectness [1] into account and provide fewer, yet better, candidate bounding boxes than the samples generated in a random or sliding window fashion. These two types of region proposals effectively alleviate the challenges of tracking failure and scale estimation

scale-aware region proposals for scale estimation. These two types of region proposals take objectness into account and generate high-quality candidate bounding boxes to re-detect target objects and address scale changes (Figs. 2.1 and 2.2).

Method

A correlation filter can be viewed as a template of the target object. Given an image patch $\mathbf{z}$ and learned correlation filter $\mathbf{w}$, the correlation response, $f(\mathbf{z}) = \mathbf{w}^{\mathrm{T}}\mathbf{z}$, indicates the similarity between the image patch $\mathbf{z}$ and the filter $\mathbf{w}$. Learning correlation filter $\mathbf{w}$ can be posed as solving a ridge regression problem as:

$$\min_{\mathbf{w}} \sum_{k=1}^{T} \left\| \sum_{d=1}^{D} x_k^d * w^d - y_k \right\|^2 + \lambda \sum_{d=1}^{D} \|w^d\|^2, \tag{2.9}$$

where $*$ represents circular correlation, x_k^d denotes k-th sample in $\mathbf{x}$ with D-dimension channels. Each shifted sample $x_k, k \in \{1, \ldots, T\}$ means a training set with T samples, has a Gaussian function label $y_k \in \mathbf{y}$, w_d denotes the d-th channel of $\mathbf{w}$. The regularization parameter λ is subject to $\lambda > 0$. Since spatial correlation can be computed in the Fourier domain as an element-wise product, Eq. 2.9 can be solved as:

$$\mathbf{w} = \frac{\mathbf{x}^{\mathrm{T}}\mathbf{y}}{\mathbf{x}^{\mathrm{T}}\mathbf{x} + \lambda}, \tag{2.10}$$

For non-linear regression, the kernel trick is applied to obtain a more powerful regression model. Expressing the solution $\mathbf{w}$ as a linear combination of the samples:

$$\mathbf{w} = \sum_i \alpha_i \phi(x_i) \tag{2.11}$$

As a result, we can optimize $\boldsymbol{\alpha}$ instead of $\mathbf{w}$ via mapping to a non-linear feature-space $\phi(\mathbf{x})$. The kernel function satisfies: $\phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{x}') = \mathbf{k}(\mathbf{x}, \mathbf{x}')$ and kernel matrix $K_{i,j} = \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$. Thus, the solution to kernelized version of regression can be formulated as:

$$\boldsymbol{\alpha} = (K + \lambda I)^{-1}\mathbf{y} \tag{2.12}$$

further, Eq. 2.12 can be diagonalized via discrete Fourier transform (DFT) as:

$$\hat{\boldsymbol{\alpha}} = \frac{\hat{\mathbf{y}}}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda}. \tag{2.13}$$

Here, $\hat{}$ denotes the DFT transformation. To detect the target location within a patch $\mathbf{z}$, the prediction function is:

$$\hat{f}(\mathbf{z}) = \hat{\mathbf{k}}^{\mathbf{xz}}\hat{\boldsymbol{\alpha}} \tag{2.14}$$

$\hat{\mathbf{k}}^{\mathbf{xz}}$ in Eq. 2.14 is the DFT of $\mathbf{k}^{\mathbf{xz}}$, which denotes the kernel correlation of $\mathbf{x}$ and $\mathbf{z}$. The Gaussian kernel can be represented as:

$$\mathbf{k}^{\mathbf{xz}} = \exp\left(-\frac{1}{\sigma^2}\left(\|\mathbf{x}\|^2 + \|\mathbf{z}\|^2 - 2\mathcal{F}^{-1}\left(\sum_c \hat{\mathbf{x}}_c^* \cdot \hat{\mathbf{z}}_c\right)\right)\right). \tag{2.15}$$

Here, σ denotes the bandwidth of the Gaussian kernel. $\mathcal{F}^{-1}$ denotes inverse DFT, $*$ refers to complex conjugation, and c means the c-th channel of the image feature. As a result, during the tracking procedure, we first crop an image patch $\mathbf{z}$ centered at the location in the previous frame and compute the response map of f using the learned target template $\bar{x}$ in the Fourier domain:

$$f(\mathbf{z}) = \mathcal{F}^{-1}(\hat{f}(\mathbf{z})) = \mathcal{F}^{-1}(\hat{\mathbf{k}}^{\bar{\mathbf{x}}\mathbf{z}}\hat{\boldsymbol{\alpha}}) \tag{2.16}$$

In contrast to translation estimation, when we perform detection over region proposals, it is not necessary to compute the full correlation response map, as in Eq. 2.16. Instead, we only compute the similarity between the learned filter and the proposal in the spatial domain. For each input patch $\mathbf{z}$ specified by a proposal, we compute its confidence score of being the target as:

$$g(\mathbf{z}) = sum(\mathbf{k}^{\bar{\mathbf{x}}\mathbf{z}} \cdot \boldsymbol{\alpha}), \tag{2.17}$$

where $sum(\cdot)$ means the summation of all the elements in a matrix, $\bar{\mathbf{x}}$ denotes the current target appearance, the output $g(\mathbf{z})$ is a scalar value rather than a matrix, as in Eq. 2.16. We apply Eq. 2.17 for both re-detecting targets and estimating scale changes.

In this work, we employ two-stage correlation filters to estimate translation and scale changes in a similar spirit to [137]. The main differences lie in that we learn translation filter $\mathbf{f}_T$ with linear kernel over deep convolutional features [170], and we employ a conservatively learned Gaussian kernel correlation filter $\mathbf{f}_L$ to re-detect target objects rather than an additional classifier. Meanwhile, we leverage Gaussian kernel correlation filter $\mathbf{f}_S$ to handle scale changes.

Instance-Aware Region Proposals

For each frame, we crop a patch $\mathbf{z}$ centered at the estimated position to compute the tracking confidence. We set a pre-defined activation threshold Θ_a to compare with $g(\mathbf{z})$ to determine whether tracking fails (also see Fig. 2.13). Generally, deciding whether a tracker fails is critical to alleviating the drifting problem. We set the activation threshold Θ_a to 0.1 according to our observation in Eq. 2.17, with an example shown in Fig. 2.3. If the confidence score $g(\mathbf{z})$ is smaller than Θ_a, we activate the long-term memory correlation filter to perform detection over the instance-aware region proposals, which are most likely to be target objects. Otherwise, we just leverage translation filter $\mathbf{f}_T$ to estimate the target location. In this paper, each proposal s specifies a candidate bounding box with the form (x_s, y_s, w_s, h_s), where (x_s, y_s) denotes the center of the bounding box and (w_s, h_s) is the width and height. In the rest of this paper, region proposals always correspond to candidate bounding boxes.

First, we build our region proposal algorithm upon the EdgeBox [235] method due to its computational efficiency and portability. Since EdgeBox is a generic proposal generator, it is prone to generating false positive proposals under cluttered background conditions. Thus, we adopt a classifier to obtain instance-aware proposals. Specifically, we incrementally train an online support vector machine (SVM) to filter the proposals from the background which

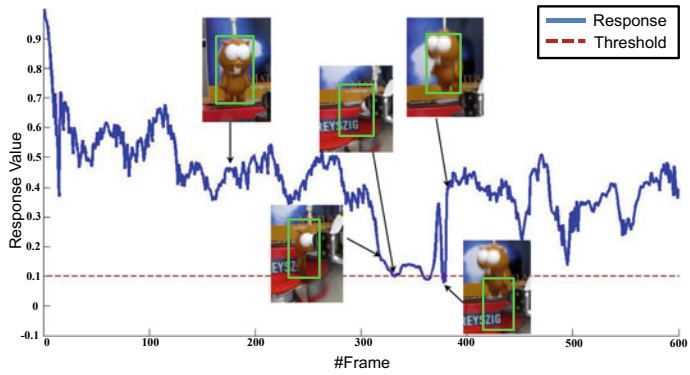


Fig. 2.3 Tracking confidence response values on the Lemming sequence. The output response values of Eq. 2.17 are generally larger than 0.1 throughout the tracking process. When tracking failure occurs, the output value drops below 0.1 rapidly

is introduced in Section C. Our method performs favorably against the recently proposed tracker [232] as we retain a small set of proposals denoted by $\mathbf{S} = \{s_1, s_2, ..., s_n\}$ for detection rather than only one proposal with the highest ranking score based on the SVM classifier in [232].

We apply a conservative scheme to adopt the re-detected proposal as it would re-initiate the tracking process. As a result, we maximize the value of $g(\mathbf{z})$ (e.g., $g(\mathbf{z}) > 1.5\Theta_a$) to decide whether the re-detected proposal is the tracking target or not. In addition, we take the motion constraint into account to prevent temporal fluctuation of the tracked bounding boxes as:

$$d(p_s, p_{i-1}) = \frac{1}{2b} \exp\left(-\frac{1}{b}|p_s - p_{i-1}|\right) \tag{2.18}$$

where p_s is the center of proposal $s \in \mathbf{S}$, and $p_{i-1} = (x_{i-1}, y_{i-1})$ is the estimated target position in the $(i-1)$th frame. We set b to the diagonal length of the searching window size.

Taking these two factors together, we select the optimal proposal as the detection result using the following objective function:

$$s^* = \arg\max_{s \in \mathbf{S}} g(\mathbf{z}_s) + \zeta d(p_s, p_{i-1}). \tag{2.19}$$

In Eq. 2.19, ζ is a weight factor used to strike a balance between the two parts. After solving Eq. 2.19, we can obtain the optimal proposal position (x_i^p, y_i^p) in frame i. The new position is updated by:

$$(x_{new}, y_{new}) = (x_{old}, y_{old}) + \gamma_1((x_i^p, y_i^p) - (x_{old}, x_{old})). \tag{2.20}$$

Here, (x_{old}, y_{old}) denotes the origin position before re-detection and γ_1 is a factor to balance the weights between the last estimated position and current position.

Adaptive Instance-Aware Proposals

In the initial frame, we use the generated proposals to train the online SVM. We assign the binary labels to these proposals according to whether their intersection over union (IoU) with the ground truth is larger than 0.5. Note that these proposals contain fewer candidate bounding boxes, which partially mix the target area and surrounding context, than the samples generated in a random and sliding window fashion. Hence, the sampling ambiguity is effectively alleviated, even though we use far fewer proposals. To incrementally update the SVM classifier, we generate instance-aware proposals around the estimated position as training data. We use the same update scheme to that in [198]. To maintain the model stability, we conservatively update the SVM classifier only when the output of the long-term memory filter $g(\mathbf{z})$ is larger than a given threshold Θ_s.

Scale-Aware Region Proposals

For tracking results with high confidence scores, we present how we address the scale estimation problem. Given the estimated position p_i, we first train a scale filter $\mathbf{f}_S$; then, we apply the Edgebox to generate scale-aware proposals centered around p_i. First, we record the maximum score of $\mathbf{f}_S$ using Eq. 2.14. We then generate scale-aware region proposals centered around p_i. We further apply the proposal rejection technique [82] to filter the proposals whose IoU with the target bounding box (centered at the estimated position with the scale of the last frame) is smaller than 0.6 or larger than 0.9. The motivation is that proposals above the upper threshold of 0.9 are those with sizes close to the current detected target, and those below 0.6 are very likely to be false proposals or to contain other objects. For each image patch specified by the remaining proposals, we resize it to a fixed size and use Eq. 2.17 to compute the correlation score with $\mathbf{f}_S$ in the spatial domain. If this score is larger than the maximum value of $\mathbf{f}_S$, which means that the target undergoes significant scale size or ratio changes, we update the target size $w_i \times h_i$ as:

$$(w_i, h_i) = (w_{i-1}, h_{i-1}) + \gamma_2((w_i^p, h_i^p) - (w_{i-1}, h_{i-1})). \tag{2.21}$$

Here, w_i^p and h_i^p denote the width and height of proposal which has highest correlation score with $\mathbf{f}_S$, respectively. We use the damping factor γ_2 to make the target size change smoothly.

Channel Regularization

Considering the convolutional features extracted from a generic convolution network (e.g., AlexNet, VGGNet [170]) pre-trained on ImageNet, the responses of all channels are varied. Specifically, as shown in Fig. 2.4, some channels are sensitive to foreground object(s) while

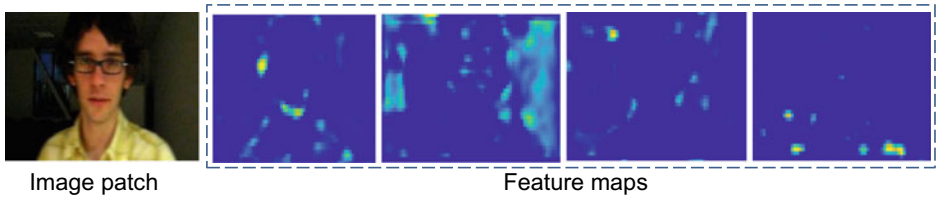


Fig. 2.4 Channel-wise response values of convolutional features on the David sequence. Given the image patch, feature x^d from different channels has different response values: the first feature map contains more information about the foreground, the second feature map contains more information about background while the rest feature maps are noisy

some channels are sensitive to background or noise. As a result, to suppress the underlying channel-wise redundancy and noise, it is meaningful to learn a weight vector $\mathbf{c} = [c^1, ..., c^D]$ for each channel:

$$\min_{\mathbf{w},\mathbf{c}} \sum_{k=1}^{T} \left\| \sum_{d=1}^{D} x_k^d \cdot c^d * w^d - y_k \right\|^2 + \lambda \sum_{d=1}^{D} \|w^d\|^2 + \xi \sum_{d=1}^{D} \left| c^d \right| \tag{2.22}$$

where $|\cdot|$ means l_1 norm which ensures the sparsity of $\mathbf{c}$, ξ is a regular hyper-parameter. Equation 2.22 can be solved by alternatively minimizing $\mathbf{w}$ and $\mathbf{c}$: we first initialize $\mathbf{c}$ to a constant vector (i.e., $\mathbf{1}$) and optimize $\mathbf{w}$ as:

$$\min_{\mathbf{w}} \sum_{k=1}^{T} \left\| \sum_{d=1}^{D} x_k^d \cdot c^d * w^d - y_k \right\|^2 + \lambda \sum_{d=1}^{D} \|w^d\|^2, \tag{2.23}$$

the solution is:

$$\mathbf{w} = \frac{(\mathbf{diag}(\mathbf{c})\mathbf{x})^T \mathbf{y}}{\mathbf{x}^{\mathrm{T}} \mathbf{diag}(\mathbf{c^2})\mathbf{x} + \lambda}, \tag{2.24}$$

where $\mathbf{diag}(\mathbf{c})$ is the diagonal matrix generated from the vector $\mathbf{c}$. Then, we fix $\mathbf{w}$ and optimize $\mathbf{c}$:

$$\begin{aligned} L &= \sum_{k=1}^{T} \left\| \sum_{d=1}^{D} (x_k^d * w^d) \cdot c^d - y_k \right\|^2 + \xi \, |\mathbf{c}| \\ &= \left\| \sum_{d=1}^{D} \mathbf{x}^d \mathbf{w}^d \cdot c^d - \mathbf{y} \right\|^2 + \xi \, |\mathbf{c}| \end{aligned} \tag{2.25}$$

which can be solve by Accelerated Proximal Gradient [186].

Model Update

Correlation filters must be updated over time to adapt to target appearance changes. In this work, we use a moving average scheme to update the correlation filters learned over both CNN features and HOG features. Given the input feature vector $\mathbf{x}^t$ generated from the image patch centered at the estimated position in the t-th frame, the learned filter $\hat{\boldsymbol{\alpha}}_N$ in the current frame can be formulated as:

$$\hat{\boldsymbol{\alpha}}_N = \frac{\hat{\boldsymbol{y}}}{\hat{\mathbf{k}}^{\mathbf{x}^t\mathbf{x}^t} + \lambda}. \tag{2.26}$$

We use a same rate η to perform interpolation between the current states ($\mathbf{x}^t$ and $\hat{\boldsymbol{\alpha}}^t$) and past states ($\bar{\mathbf{x}}^{t-1}$ and $\hat{\boldsymbol{\alpha}}^{t-1}$) as follows:

$$\begin{aligned} \hat{\boldsymbol{\alpha}}^t &= \eta\hat{\boldsymbol{\alpha}}_N + (1-\eta)\hat{\boldsymbol{\alpha}}^{t-1}, \\ \bar{\mathbf{x}}^t &= \eta\mathbf{x}^t + (1-\eta)\bar{\mathbf{x}}^{t-1}. \end{aligned} \tag{2.27}$$

Experiments

Implementation Details

We tailor the EdgeBox [235] toolbox to generate two types of region proposals. One type is the instance-aware proposals for re-detecting target objects. To ensure the proposals have high recall rates for the target objects, we incrementally learn an additional linear SVM classifier as a filter. The other type is scale-aware region proposals. We set a small step size α and a small non-maximum suppress (NMS) threshold β to center the proposals around the estimated position. We summarize the schemes for generating these two types of proposals below:

- We set α to 0.85 and β to 0.8 to generate instance-aware proposals, whereas we set α to 0.65 and β to 0.75 to generate scale-aware proposals.
- Let $A_{i-1} = w_{i-1} \times h_{i-1}$ be the target size in the previous frame. We generate instance-aware proposals of size between 0.7 and 1.3 times that of A_{i-1}. The size of the scale-aware proposals ranges from 0.3 to 1.4 times that of A_{i-1}.
- We use a fixed aspect ratio in the last frame to generate instance-aware proposals. For scale-aware region proposals, we set the aspect ratio to $1.5 \times \max(\frac{w_{i-1}}{h_{i-1}}, \frac{h_{i-1}}{w_{i-1}})$.

The stability threshold Θ_s is 0.3. We set the learning rate η to 0.01 and the weight factor ζ in Eq. 2.19 to 0.1. The damping factors γ_1 in Eq. 2.20 and γ_2 in Eq. 2.21 are set to 0.8 and 0.6, respectively. The regression target $\mathbf{y}$ simply follows a two-dimensional Gaussian function with a kernel width of 0.1. The size of searching windows is 3 times of the target size as [170]. We exploit two types of features for learning the correlation filters. For translation estimation, we use the outputs of the conv5-4 (512), conv4-4 (512), and conv3-4 (256) layers from the VGGNet-19 model [170] as features. We learn the correlation filter over

Algorithm 1 Proposed tracking algorithm

Require: Previous target position (x_{t-1}, y_{t-1}) and size (w_{t-1}, h_{t-1}), translation filter $\mathbf{f}_{\mathrm{T}}$, scale filter $\mathbf{f}_{\mathrm{S}}$, and long-term memory filter $\mathbf{f}_{\mathrm{L}}$;
Ensure: Estimated position (x_t, y_t) and size (w_t, h_t).
repeat
  Crop out the searching window in frame t centered at (x_{t-1}, y_{t-1});
  Estimate new position (x_t, y_t) from correlation response map using $\mathbf{f}_{\mathrm{T}}$ over deep convolutional features;
  Crop target patch $\mathbf{z}$ centered at (x_t, y_t) and compute correlation score $g(\mathbf{z})$ over HOG features using $\mathbf{f}_{\mathrm{L}}$;
  if $g(\mathbf{z}) < \Theta_a$ **then**
    Perform detection over instance-aware proposals S;
    if $\max\{g(\mathbf{z}_s)|s \in \mathbf{S}\} > 1.5\Theta_a$ **then**
      Refine (x_t, y_t) using the optimal proposal and Eq. 2.19;
    end if
  else
    Estimate scale changes over scale-aware proposals using the filter $\mathbf{f}_{\mathrm{S}}$;
    Update target size (w_t, h_t) using Eq. 2.21;
  end if
  Update $\mathbf{f}_{\mathrm{T}}$, $\mathbf{f}_{\mathrm{S}}$;
  if $g(\mathbf{z}) > \Theta_s$ **then**
    Update $\mathbf{f}_{\mathrm{L}}$;
  end if
until End of video sequences.

each convolutional layer. The channel regularization hyper-parameter ξ is set to 0.01. We use the weighted sum scheme on multiple correlation response maps to infer target position. By contrast, we learn the long-term correlation filter and scale estimation correlation filter over HOG features.

The algorithmic steps are summarized in Algorithm 1.

Experimental Settings and Evaluation Metrics

For convenience, we name the proposed tracker CFRP (correlation filter with region proposal). We implement the proposed CFRP in MATLAB and utilize the MatConvNet toolbox [189] to build the CNN. Our implementation runs at 6.5 frames per second on a computer with an Intel Xeron 2.4 GHz CPU, 64GB RAM. We extract CNN features using an Nvidia K20 GPU.

We validate our CFRP tracker on five large-scale tracking benchmark datasets: two Object Tracking Benchmark (OTB) datasets [206, 207], the VOT-2015 dataset [49], VOT-2015 dataset [101] and UAV-123 dataset [145]. The OTB-2015 dataset [207] extends OTB-2013 [206] with an additional 50 videos, and it is composed of 100 videos with more than 58,000 frames in total. OTB-2015 is more challenging as it contains abundant shaking and dramatic focal changing video sequences.

For OTB and UAV validation, we follow the benchmark evaluation prototype [206] and fix all the parameters for all the test sequences. We report our one-pass evaluation (OPE) results in terms of the distance precision plot and the overlap success plot. The distance precision plot is computed as the percentage of frames in which the distances of the centers between tracking results and ground truths is below a predefined pixel threshold. The overlap success plot is computed as $\frac{area(R_T \cap R_G)}{area(R_t \cup T_G)}$, which represents the extent of region overlapping between tracking results R_T and ground truths R_G over all frames. We report the area under the curve (AUC) of each overlap success plot to evaluate the tracking algorithm. Three performance measures are used for VOT2015 dataset: (i) accuracy which measures how well the predicted bounding box overlaps with the ground truth bounding box; (ii) robustness measures the frequency of tracking failures; (iii) expected average overlap (EAO) which computes the average overlaps on a set of sequences.

Overall Performance

Results on OTB

On the OTB dataset, we compare CFRP with 15 state-of-the-art trackers. These trackers mainly fall into the following three categories: (i) Deep learning trackers, including ECO [34], CCOT [41], MDNet [149], CREST [172], HCFT [134] and FCNT [194]; (ii) Correlation-filter-based trackers, including SRDCF [39], LCT [137], MUSTer [79], DSST [35] and KCF [74]; and (iii) representative tracking by detection algorithms, including TGPR [61], MEEM [217], Struck [68] and TLD [90].

Figure 2.5 shows the overall results under OPE using the distance precision rate and overlap success rate. We report the results on both the OTB-2013 [206] and OTB-2015 [207] datasets. The proposed CFRP tracker generally performs favorably against the state-of-the-art trackers. Among the compared trackers, the MDNet tracker achieves the best distance precision, but our CFRP tracker achieves distance precision rates that are 4.5% and 5.0% greater than those of HCFT on OTB-2013 and OTB-2015, respectively. With respect to the overlap success rate, the proposed method performs favorably against the recently proposed

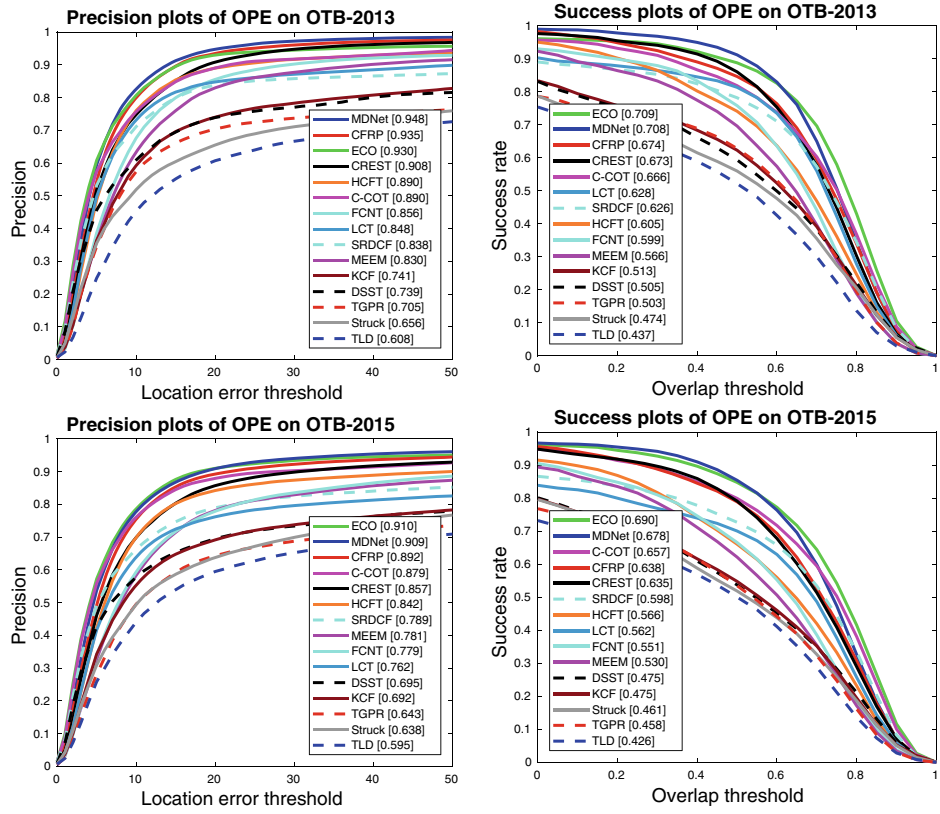


Fig. 2.5 Overall performance on the OTB-2013 [206] and OTB-2015 [207] datasets using one-pass evaluation (OPE). The legend of the distance precision plots contains the thresholded scores at 20 pixels, whereas the legend of the overlap success plots contains area-under-the-curve (AUC) scores for each tracker. The proposed CFRP tracker performs well against the state-of-the-art trackers

CREST [172], HCFT [134], FCNT [194], LCT [137], and SRDCF [39] trackers because we employ scale-aware region proposals to provide better candidate bounding boxes for scale estimation rather than searching for the scale space using brute force, as the FCNT [194], LCT [137], SRDCF [39] trackers do. The HCFT tracker could not handle scale changes at all. Our method is slightly inferior to MDNet [149] and ECO [34] but superior to CCOT [41]. We attribute the performance gap to that MDNet takes advantage of domain transfer learning from labaled videos while ECO uses more complex updating scheme.

We further compare the tracking performance for different video attributes (e.g., scale variation, occlusion, fast motion) annotated in the benchmark [207]. Figure 2.6 illustrates that our tracker performs well the against state-of-the-art methods in terms of the distance precision rate on all 100 benchmark sequences. Our tracker uniformly achieves state-of-the-art performance with occlusion (86.1%), out of view (75.9%), scale variation (87.0%),

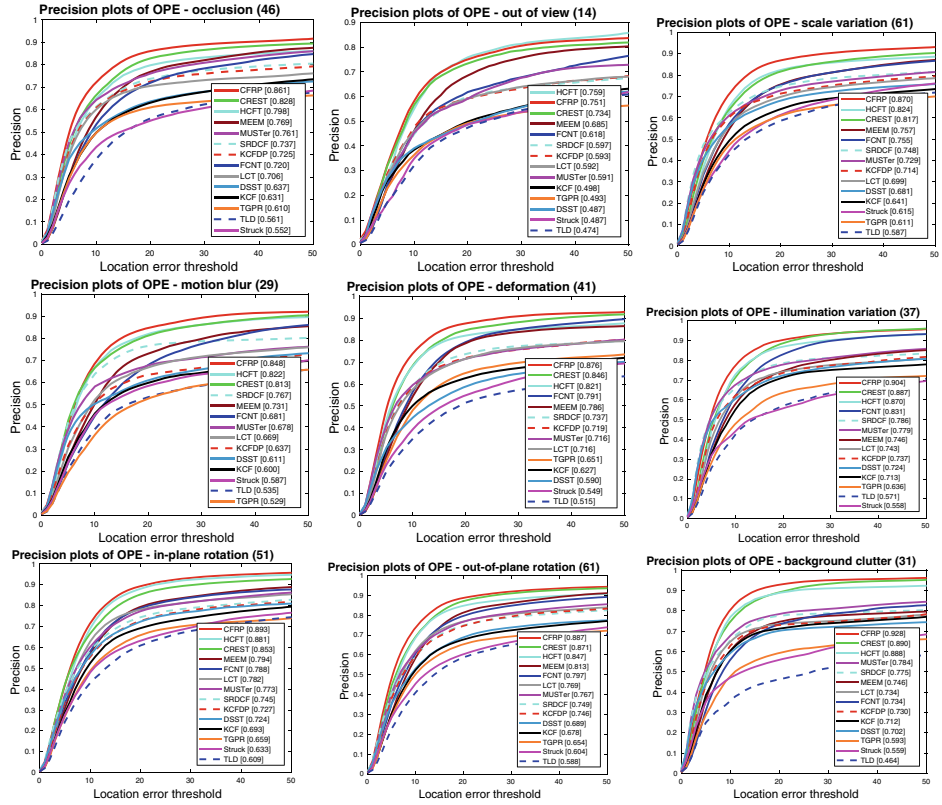


Fig. 2.6 Quantitative results on 9 challenging attributes on the OTB-2015 [207] dataset using one-pass evaluation (OPE). The legend of the distance precision plots contains the thresholded scores at 20 pixels, whereas the legend of the overlap success plots contains area-under-the-curve (AUC) scores for each tracker. The proposed CFRP generally performs better than the state-of-the-art trackers

motion blur (84.8%), deformation (87.6 %), illumination variation (90.4%), in-plane rotation (89.3%), out-of-plane rotation (88.7%) and background clutter (92.8%). These results suggest that region proposals are effective and efficient for handling various challenges, especially tracking failures caused by occlusion and out-of-view movement, as well as for addressing scale variation.

Qualitative Evaluation

We qualitatively validate the proposed tracker on the six most challenging sequences and compare the performance to that of the top-performing trackers, FCNT [194], HCFT [134], Struck [68], and MEEM [217], in Fig. 2.7. For the challenging sequence *MotorRolling*, with attributes of deformation and rotation, the Struck and MEEM trackers, which rely on hand-

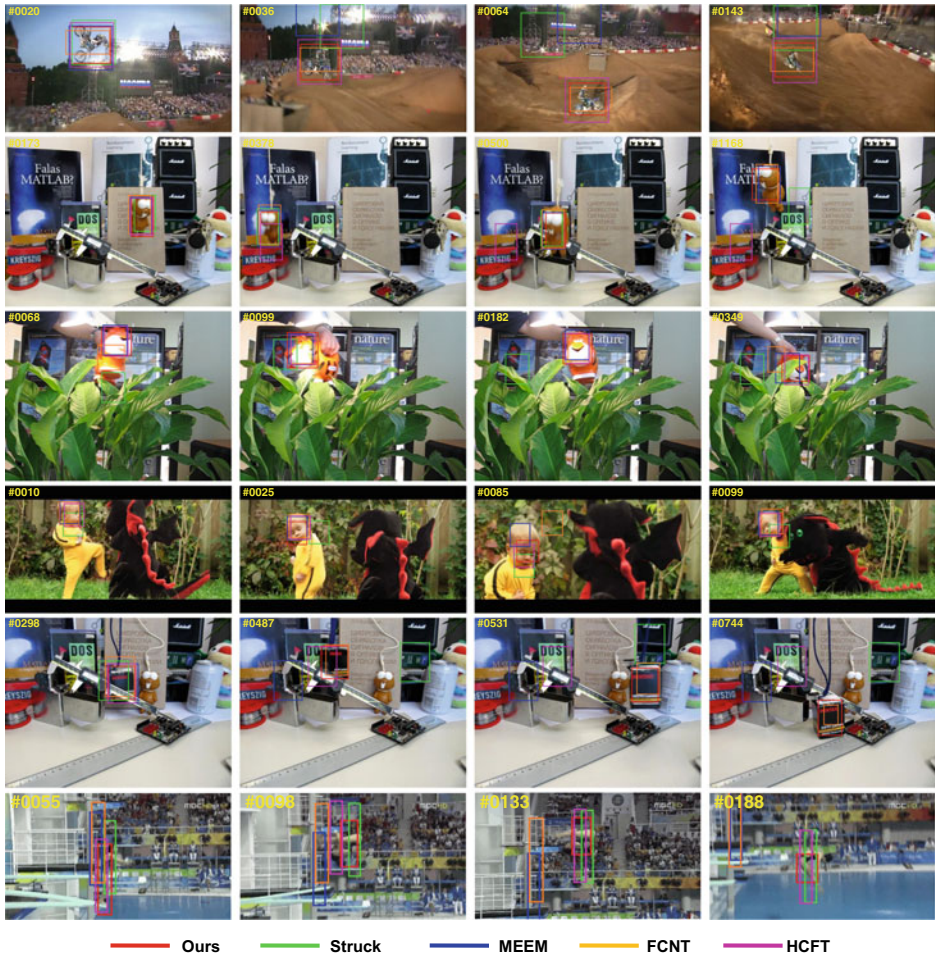


Fig. 2.7 Tracking results on five challenging benchmark sequences by our tracker and the HCFT [134], FCNT [194], MEEM [217] and Struck [68] trackers. Our tracker performs well against the state-of-the-art trackers in terms of precise localization and scale estimation. From the first row to the last: *Motorrolling*, *Lemming*, *Tiger1*, *DragonBaby*, *Box* and *Diving*

crafted features, both fail at the beginning. Although convolutional feature [170] facilitate the FCNT and HCFT trackers, they do not perform well for scale estimation. Our CFRP tracker exploits two types of region proposals and effectively alleviates the challenging scale estimation issue. Similar results are obtained for the *Diving* sequence.

In the presence of partial occlusion (*Tiger1*), the Struck, MEEM and FCNT trackers drift quickly, whereas the proposed CFRP tracker performs well due to the effectiveness of the detection scheme. On the *Box* sequence, the target is under heavy occlusion between the 460th and 480th frames. The Struck and HCFT trackers are too adaptive to re-detect the

Table 2.2 The performance of top 15 trackers on the VOT2015 dataset. We report the average ranks under accuracy, robustness and expected average overlap. Red, blue and green colors denote the first, second and third best results

	Accuracy	Robustness	EAO
MDNet	0.5977	0.7749	0.3926
CFRP	0.5617	0.8631	0.3809
DeepSRDCF	0.5774	0.8728	0.3701
SRDCF	0.4833	0.9538	0.3386
LDP	0.4605	1.0559	0.3212
sPST	0.4705	1.3759	0.3082
SAMF	0.4023	1.3329	0.2826
Struck	0.4711	1.7320	0.2516
SODLT	0.4360	2.1677	0.2768
MEEM	0.4790	2.0750	0.2170
MCT	0.4591	2.1489	0.1840
MUSTer	0.4790	2.7501	0.1612
TGPR	0.4501	2.8543	0.1600
DSST	0.4290	3.1750	0.1543
KCFDP	0.4318	3.4750	0.1503
IVT	0.3521	4.1448	0.1484
OGT	0.3470	3.6309	0.1389
MIL	0.3590	4.1130	0.1360
CT	0.3266	7.1179	0.0722

target after the 480th frame, whereas the proposed CFRP tracker performs well in estimating both the translation and scale variation. For the *Box* and *Lemming* sequences, which are characterized by both occlusion and background clutter, the MEEM tracker succeeds in re-detecting the target on the *Lemming* sequence but fails on the *Box* sequence. This shows that densely drawn samples do not take the objectness information into account and are less effective in re-detecting targets after tracking failures. The *DragonBaby* sequence has attributes of scale variation, occlusion, motion blur, fast motion, rotation and out-of-view movement; the proposed CFRP performs favorably against the state-of-the-art trackers due to the effectiveness of region proposals for detection and scale estimation (Table 2.2).

Results on VOT-2015 and VOT-2018

We report the evaluation results of CFRP on the VOT-2015 dataset [49], which contains 60 challenging sequences. Each sequence is labeled with six attributes, including camera

Table 2.3 Overall performance on VOT-2018 in comparison to the state-of-the-art trackers. EAO: Expected average overlap. A Accuracy, R: Robustness. Best result for each item is labeled in **bold**

	CFRP+	CFRP	LADCF	SiamRPN	DSLTpp	SA_Siam_R	CPT	ECO
	Ours	**Ours**	[208]	[110]	[195]	[69]	[101]	[34]
EAO ↑	**0.392**	0.299	0.389	0.352	0.325	0.337	0.339	0.280
A ↑	**0.509**	0.486	0.503	0.566	0.543	0.566	0.506	0.484
R ↓	**0.153**	0.264	0.159	0.276	0.224	0.258	0.239	0.276
	DeepSTRCF	LSART	CSRDCF	SRCT	CFTR			
	[111]	[176]	[132]	[101]	[17]			
EAO ↑	0.345	0.323	0.256	0.310	0.300			
A ↑	0.523	0.495	0.491	0.520	0.505			
R ↓	0.290	0.258	0.378	0.159	0.184			

motion, illumination change, motion change, occlusion, size change and no degradation. Following the VOT2015 challenge, we use the average accuracy, robustness rank and expected average overlap (EAO) as our evaluation criteria. We compare all the trackers that participate in the VOT-2015 challenge fairly.

Table 2.6 shows the top 18 trackers on the VOT-2015 datasets, including convolutional feature based trackers: MDNet [149], DeepSRDCF [38], LDP [133], SODLT [131] and correlation-filter based trackers: SRDCF [39], MUSTer [79], sPST [81], DSST [35], SAMF [119], KCF [74] and traditional trackers: Struck [68], MCT [151], IVT [163], OGT [150], MIL [7], CT [219] and NCC [89]. The proposed CFRP method ranks second in terms of accuracy, robustness and overall evaluations which is slightly superior to MDNet [149]. It is worth mentioning that MDNet utilizes the video from similar datasets (e.g., OTB) to pre-train the network. However, our method only adopts the first frame to train the model and achieves better performance than all other comparison methods. Note than DeepSRDCF employ a spatial regularization term to handle the boundary effects in correlation filter, as a result, achieves slightly better performance according to the accuracy. Our CFRP performs much better than the rest of all compared methods.

To further prove the effectiveness of the proposed adaptive region proposal scheme, we evaluate the proposed method on VOT-2018 dataset [101]. We take the LADCF [208] as the baseline and extend it with the proposed adaptive region proposal scheme as the new method (CFRP+) on VOT-2018 dataset. From Table 2.7, we can see that our CFRP+ achieves superior performance than all compared methods including the baseline LADCF. Overall results demonstrate the effectiveness of the proposed scheme (Table 2.3).

Results on UAV-123

The UAV-123 dataset is a recently released long term dataset which contains 123 tracking targets. The shortest video contains at least 1717 frames, and the longest one contains more than

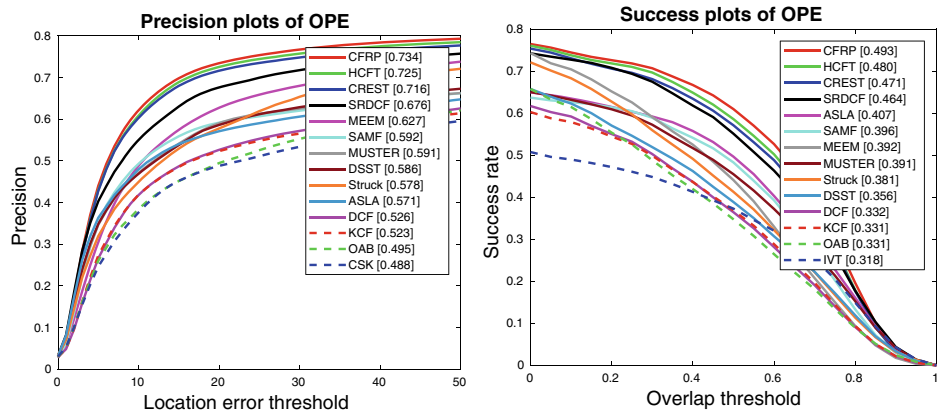


Fig. 2.8 Overall performance on the UAV-123 dataset [145] using one-pass evaluation (OPE)

5000 frames. Lots of tracking targets undergo occlusion and out of view which makes this dataset is very challenging. Our CFRP is compared with state-of-the-art trackers including CREST [172], HCFT [134], MEEM [217], SRDCF [39], MUSTer [79], DSST [35], SAMF [119], KCF [74] and TLD [90] (Fig. 2.8).

Figure 2.20 shows the whole tracking performances of all compared methods. CFRP achieves the best performance on both the distance precision rate (73.4%) and the success overlap rate (49.3%). Specifically, compared with HCFT method, our CFRP achieves better performance on this long term tracking benchmark which proves the effectiveness of the proposed adaptive region proposal.

Ablation Studies

To validate the effectiveness of the tracking components of the proposed algorithm, we conduct the following ablation studies. We first validate the effectiveness of the adaptivity for the instance-aware proposals, i.e., we remove the online SVM module to disable the adaptivity of the region proposals. We then validate the importance of the correlation filter detector by replacing it with the online SVM classifier. To verify the effectiveness of region proposals for scale estimation, we disable the scale estimation module. Furthermore, we disable the region proposals from both the tracker and detection streams. Finally, we remove the channel regularization module. Figure 2.9 shows the distance precision rate and overlap success rate of these baseline implementations on the OTB-2015 dataset.

Figure 2.9 provides the following observations. When we disable the online SVM classifier for instance-aware proposals, the performance of CFRPnl (none-learning) drops quickly ($0.892 \rightarrow 0.825, 0.638 \rightarrow 0.570$). The main reason is that without online SVM, the proposals from the background contaminate the detection results easily. CFRPnd (none-detection) performs slightly worse than the proposed CFRP tracker ($0.892 \rightarrow 0.871, 0.638 \rightarrow 0.592$),

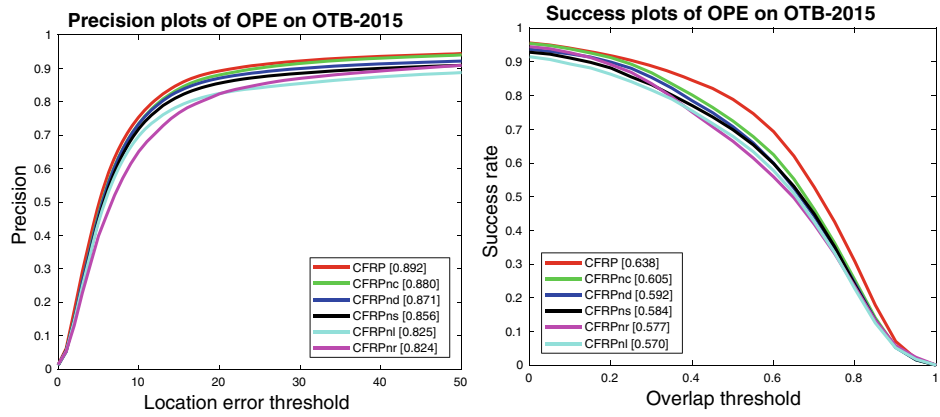


Fig. 2.9 Ablation studies. The CFRPnd (non-detection) method disables the detection module. CFRPnl (non-learning) disables the online SVM classifier, and CFRPns (non-scale) disables the scale estimation module. For the CFRPnr (non-region proposal) method, both the redetection and scale estimation modules are removed. For CFRPnc, channel regularization module is removed. The proposed CFRP tracker takes all these components into account and performs the best in these ablation studies

demonstrating the importance of integrating a detector into the correlation filters to improve the tracking accuracy. When we disable the scale estimation module, the overlap success rate drops rapidly ($0.892 \rightarrow 0.856$, $0.638 \rightarrow 0.584$). In this case, the tracker has no ability to handle the scale variation which confirms the effectiveness of our scale-aware region proposal to address the challenging scale estimation problem. Moreover, channel regularization helps to reduce redundancy information for location estimation, as a result, remove channel regularization has slight effect on final performance ($0.892 \rightarrow 0.880$, $0.638 \rightarrow 0.605$). Finally, we disable both the detection and scale estimation modules for CFRPnr (none region proposal), which performs the worst among the five baseline trackers with respect to the distance precision ($0.892 \rightarrow 0.824$, $0.638 \rightarrow 0.577$) due to both the occlusion and the scale variation aggravate the tracking performance.

To evaluate the effectiveness of the region proposals, we conduct ablation studies by replacing the region proposal module with traditional sampling methods, including the particle filter [3] and dense sampling [198]. Figure 2.10 illustrates the overall tracking performance on the OTB-2015 dataset. Note that the region proposal scheme demonstrates superior performance compared to the particle filter (CFRP_pf) and dense sampling (CFRP_ds) methods, indicating that region proposal ensures a higher recall rate of the target objects than those of the alternative approaches. The proposed CFRP tracker combines all these components together and achieves the best results.

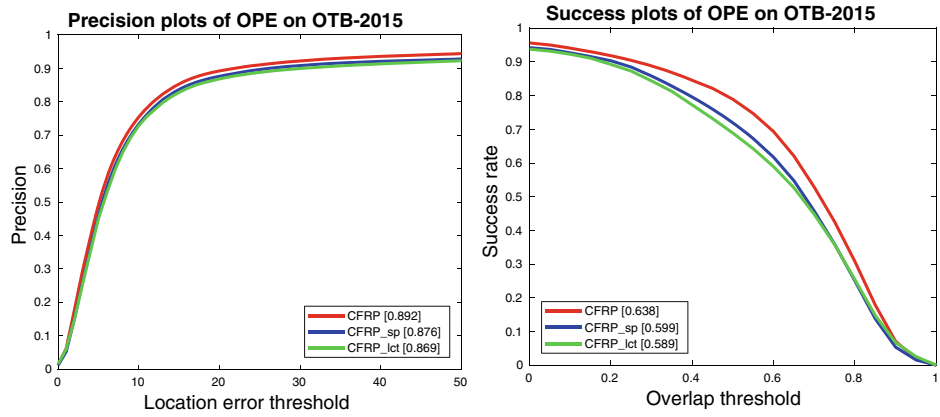


Fig. 2.10 Ablation study for long-term memory filter with a classifier as in [137] and scale estimation on the OTB-2015 dataset

Stability of Parameter Variation

This subsection validates the effectiveness of several key parameters θ_a, α and β in our CFRP tracker. To quantitatively analyze the influence of θ_a to the final tracking performance, we provide the corresponding tracking results on OTB2015 dataset under different selections of θ_a. Figure 2.11 shows the influence of θ_a with different values 0, 0.05, 0.1, 0.2, 0.5 on overlap rate (OR) and distance precision (DP). We see that if θ_a is too small, the proposed tracking method fails to judge tracking failure. If θ_a is too large, it will lead to frequent re-detection during tracking procedure. As a result, we set the value of θ_a to 0.1 in this paper.

Parameters α and β influence the quality of the generated proposals in Edgebox [36]. The quality of the generated proposals is most related to the scale estimation. We verify

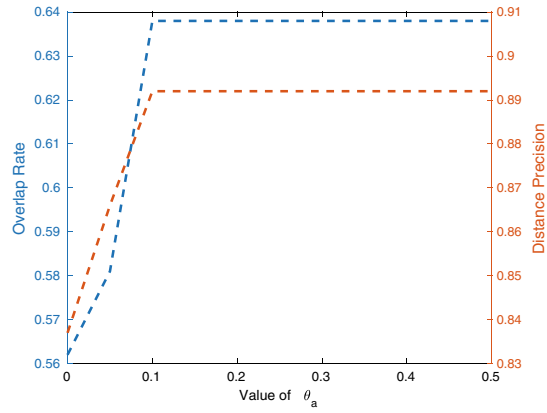


Fig. 2.11 Tracking performance with varying values of θ_a on OTB-2015 dataset. The average overlap rate and average distance precision versus the value of θ_a are illustrated

Table 2.4 Evaluation of different values of α and β on OTB-2015 dataset which is measured by overlap rate (OR) and distance precision (DP)

	OR (%)	DP (%)
$\alpha = 0.55, \beta = 0.55$	88.7	63.4
$\alpha = 0.65, \beta = 0.65$	89.2	63.8
$\alpha = 0.65, \beta = 0.75$	89.2	63.8
$\alpha = 0.75, \beta = 0.75$	89.2	63.8
$\alpha = 0.85, \beta = 0.55$	88.5	63.0
ResNet18	89.5	63.9
MobileNet	88.4	62.8

Table 2.5 Tracking speed comparison with state-of-the-art methods on GTX-2080

Method	CFRP (Ours)	MDNet [149]	HCFT [134]	ECO [34]	CCOT [41]	CREST [172]
Speed (FPS)	22.8	1.3	24.6	20.9	16.1	18.7

the stability of the setting of α and β. The corresponding results of different values of α and β are reported in Table 2.4. It can be observed that slight change the values of α and β have little effect on the tracking performance. We also study the performance on various network architectures, we take the well-known light-weighted networks (e.g., ResNet and MobileNet) as the examples. From Table 2.4, we can see different light-weight network architectures have a slight influence on the final tracking performance.

From Table 2.5, we can see that the speed of our method is close to real-time which is faster than representative correlation filter-based methods: ECO [34] and CCOT [41], meanwhile, much faster than MDNet [149].

Conclusion

This paper presents an effective algorithm for visual tracking. We integrate ensemble correlation filters with the region proposal scheme into a two-stream tracking framework. We employ adaptive region proposals to generate fewer, yet better, candidate bounding boxes to facilitate target object re-detection after tracking failure. Since region proposals naturally take the objectness information into account, we employ scale-aware region proposals to address the challenging scale estimation issue. In addition, to handle the channel redundancy and noise of the convolutional feature, we apply regularization to feature channels during the filter learning. The proposed tracker performs favorably against state-of-the-art trackers in terms of accuracy and effectiveness.

2.1.6 Deep Regression Tracking

Motivation

This section introduces DSLT that paid to data imbalance in visual tracking. For the deep regression trackers that directly learn a dense mapping from input images of target objects to soft response maps, we identify their performance is limited by the extremely imbalanced pixel-to-pixel differences when computing regression loss. This prevents existing end-to-end learnable deep regression trackers from performing as well as discriminative correlation filters (DCFs) trackers. For the deep classification trackers that draw positive and negative samples to learn discriminative classifiers, there exists heavy class imbalance due to a limited number of positive samples when compared to the number of negative samples. To balance training data, DSLT proposed a novel shrinkage loss to penalize the importance of easy training data mostly coming from the background, which facilitates both deep regression and classification trackers to better distinguish target objects from the background (Fig. 2.12).

Method

To demonstrate the effectiveness of the proposed shrinkage loss in handling data imbalance, we start by developing our tracker within the one-stage regression framework owing to the convenience in implementation. Figure 2.13 shows an overview of the proposed regression network. In the following (Sect. 2.1.6), we first briefly revisit learning deep regression networks. We then present the proposed shrinkage loss in the context of regression learning in detail (Sect. 2.1.6). To facilitate regression learning with shrinkage loss, we exploit multi-level semantics across convolutional layers with residual connections in Sect. 2.1.6. Finally, we show the generalization ability of our shrinkage loss to classification learning based object tracking (Sect. 2.1.6).

Convolutional Regression

Convolutional regression networks regress a dense sampling of inputs to soft labels which are usually generated by a Gaussian function. Here, we formulate the regression network as

Fig. 2.12 Qualitative results of different loss functions for learning one-stage regression trackers on the *Bolt2* sequence [207]. The proposed regression tracker (DSLT*) with shrinkage loss performs much better than that with the L2 and L3 losses

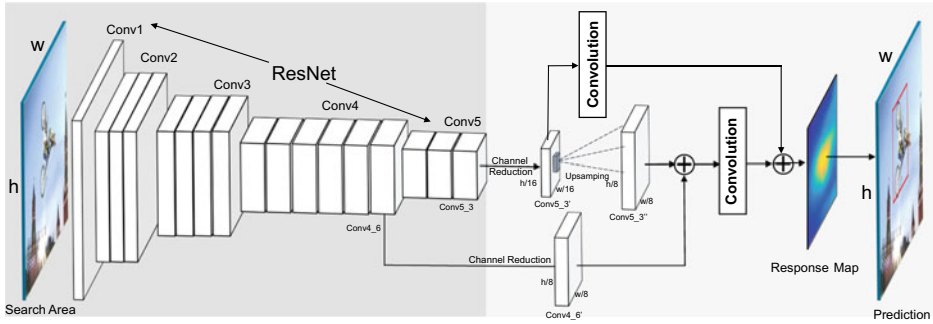


Fig. 2.13 Overview of the proposed deep regression network for tracking. Left: Fixed feature extractor (ResNet50 [72]). Right: Regression network trained in the first frame and updated frame-by-frame. We apply residual connections to both convolutional layers and output response maps, and use a bilinear interpolation layer for upsampling. The proposed network effectively exploits multi-level semantic abstraction across convolutional layers (Sect. 2.1.6). Our shrinkage loss helps to break the performance bottleneck of one-stage regression trackers caused by data imbalance and accelerates the convergence of network training (Sect. 2.1.6)

one convolutional layer. Formally, learning the weights of the regression network involves solving the following minimization problem:

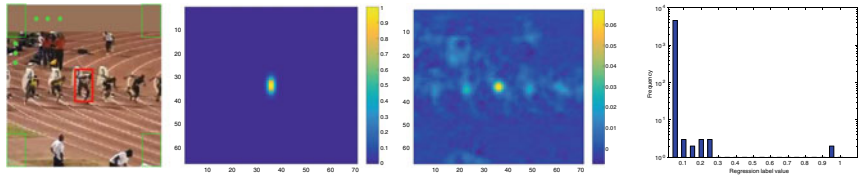
$$\arg\min_{\mathbf{W}} \|\mathbf{W} * \mathbf{X} - \mathbf{Y}\|^2 + \lambda \|\mathbf{W}\|^2, \tag{2.28}$$

where $*$ denotes the convolution operation and $\mathbf{W}$ denotes the kernel weight of the convolutional layer. Note that there is no bias term in Eq. 2.28 as we set the bias parameters to 0. $\mathbf{X}$ means the input features. $\mathbf{Y}$ is the matrix of soft labels, and each label $Y_{i,j} \in \mathbf{Y}$ ranges from 0 to 1. λ is the regularization weight. We estimate the target translation by searching for the location of the maximum value of the output response map. The size of the convolution kernel $\mathbf{W}$ is either fixed (e.g., 5×5) or proportional to the size of the input features $\mathbf{X}$. Let η be the learning rate, we iteratively optimize $\mathbf{W}$ by minimizing the square loss:

$$\begin{aligned} \mathcal{L}(\mathbf{W}) &= \|\mathbf{W} * \mathbf{X} - \mathbf{Y}\|^2 + \lambda \|\mathbf{W}\|^2 \\ \mathbf{W}_t &= \mathbf{W}_{t-1} - \eta \frac{\partial L}{\partial \mathbf{W}}. \end{aligned} \tag{2.29}$$

Regression Tracking with Shrinkage Loss

In order to learn convolutional regression networks, the input search area has to contain a large body of background surrounding target objects (Fig. 2.14a). As the surrounding background contains valuable context information, a large area of the background helps strengthen the discriminative power of the target objects from the background. However,



(a) Input patch (b) Soft labels $\mathbf{Y}$ (c) Outputs $\mathbf{P}$ (d) Histogram of $|\mathbf{P} - \mathbf{Y}|$

Fig. 2.14 **a** Input patch. **b** The corresponding soft labels $\mathbf{Y}$ generated by Gaussian function for training. **c** The output regression map $\mathbf{P}$. **d** The histogram of the absolute difference $|\mathbf{P} - \mathbf{Y}|$. Note that easy samples with small absolute difference scores dominate the training data. See Sect. 2.1.6 for details

this increases the number of easy samples from the background as well. These easy samples produce a large loss in total to make the learning process unaware of the valuable samples close to targets. Formally, we denote the response map in every iteration by $\mathbf{P}$, which is a matrix of size $m \times n$. $P_{i,j} \in \mathbf{P}$ indicates the probability of the position $i \in [1, m]$, $j \in [1, n]$ being the target object. Let l be the absolute difference between the estimated probability $P_{i,j}$ and its corresponding soft label y, i.e., $l = |P_{i,j} - Y_{i,j}|$. Note that, when the absolute difference l is larger, the sample at the location (i, j) is more likely to be the hard sample and vice versa. Figure 2.14d shows the histogram of the absolute differences. Note that easy samples with small absolute difference scores dominate the training data.

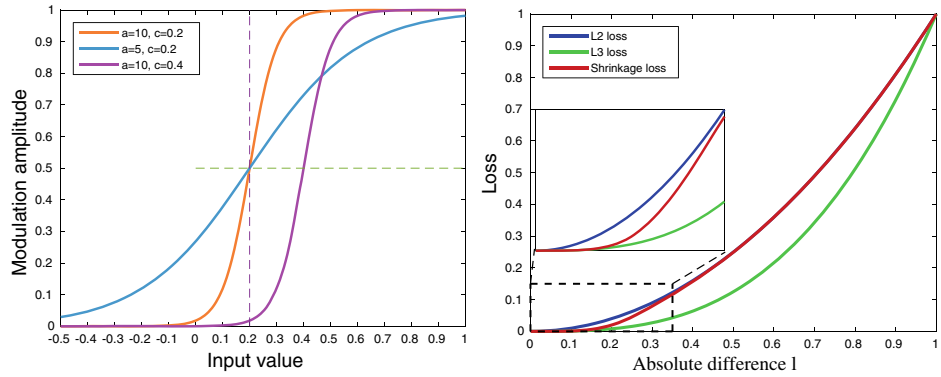
In terms of the absolute difference l, the square loss in regression learning can be formulated as:

$$\mathcal{L}_2 = |P_{i,j} - Y_{i,j}|^2 = l^2. \tag{2.30}$$

It is worth noting that, as the output probability $P_{i,j}$ learns to regress the ground truth $Y_{i,j} \in [0, 1]$, $P_{i,j}$ almost ranges between 0 and 1 during the training process. Hence the absolute difference l almost ranges between 0 and 1 as well. The recent work on dense object detection [122] shows that adding a modulating factor to the cross entropy loss helps alleviate the data imbalance issue. The modulating factor is a function of the output probability with the goal to decrease the loss from easy samples. In regression learning, this amounts to re-weighting the square loss using an exponential form of the absolute difference term l as follows:

$$\mathcal{L}_F = l^\gamma \cdot \mathcal{L}_2 = l^{2+\gamma}. \tag{2.31}$$

For simplicity, we set the parameter γ to 1 as we observe that the performance is not sensitive to this parameter, i.e., $\mathcal{L}_F = l^3$. Note that, the weight not only penalizes easy samples (i.e., $l < 0.5$) but also penalizes hard samples (i.e., $l > 0.5$). By revisiting the shrinkage estimator [33] and the cost-sensitive weighting strategy [106] in learning regression networks, instead of using the absolute difference l as weight, we propose a modulating factor with respect to l to re-weight the square loss to penalize easy samples only. The modulating function is with the shape of a Sigmoid-like function as:



(a) Modulating factor (b) Loss comparison

Fig. 2.15 **a** Modulating factors in Eq. 2.32 with different hyper-parameters. **b** Comparison between the square loss (L_2), L_3 loss and the proposed shrinkage loss for regression learning. The proposed shrinkage loss only decreases the loss from easy samples ($l < 0.5$) and keeps the loss from hard samples ($l > 0.5$) unchanged. See Sect. 2.1.6 for details

$$f(l) = \frac{1}{1 + \exp\left(a \cdot (c - l)\right)}, \tag{2.32}$$

where a and c are hyper-parameters controlling the shrinkage speed and the localization respectively. Figure 2.15a shows the shapes of the modulating function with different hyper-parameters. When applying the modulating factor to weight the square loss, we obtain the proposed shrinkage loss as:

$$\mathcal{L}_S = \frac{l^2}{1 + \exp\left(a \cdot (c - l)\right)}. \tag{2.33}$$

As shown in Fig. 2.15b, the proposed shrinkage loss only penalizes the importance of easy samples ($l < 0.5$) and keeps the loss of hard samples almost unchanged ($l > 0.5$) when compared to the square loss (L_2). In contrast, the L_3 loss penalizes both the easy and hard samples.

When applying the shrinkage loss to Eq. 2.28, we employ the cost-sensitive weighting strategy [106] and utilize the values of soft labels as an importance factor, e.g., $\exp(\mathbf{Y})$, to highlight the valuable rare samples. In summary, we rewrite Eq. 2.28 with the shrinkage loss for learning regression networks as:

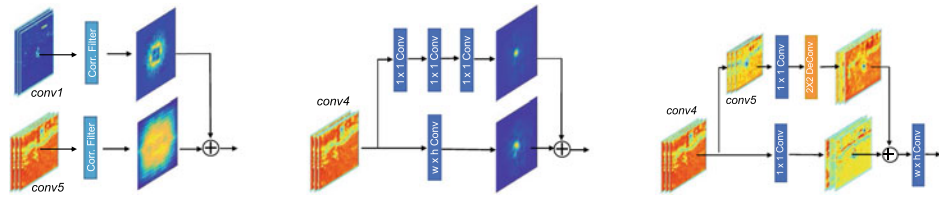
$$\mathcal{L}_S(\mathbf{W}) = \frac{\exp(\mathbf{Y}) \cdot \|\mathbf{W} * \mathbf{X} - \mathbf{Y}\|^2}{1 + \exp(a \cdot (c - |\mathbf{W} * \mathbf{X} - \mathbf{Y}|))} + \lambda \|\mathbf{W}\|^2. \tag{2.34}$$

We set the value of a to be 10 to shrink the weight function quickly and the value of c to be 0.2 to adapt to the distribution of l, which ranges from 0 to 1. Extensive comparison with the

other losses shows that the proposed shrinkage loss not only improves the tracking accuracy but also accelerates the training speed (see Sect. 2.1.6).

Convolutional Layer Connection

It is well known that CNN models consist of multiple convolutional layers emphasizing different levels of semantic abstraction. For visual tracking, early layers with fine-grained spatial details are helpful in precisely locating target objects; while the later layers maintain semantic abstraction that are robust to significant appearance changes. To exploit both merits, existing deep trackers [34, 134, 194] develop independent models over multiple convolutional layers and integrate the corresponding output response maps with empirical weights. When learning regression networks, we observe that semantic abstraction plays a more important role than spatial detail in dealing with appearance changes. The FCNT exploits both the *conv4* and *conv5* layers and CREST [172] merely uses the *conv4* layer. Our studies in Sect. 2.1.6 also suggest that regression trackers perform well when using the *conv4* and *conv5* layers as the feature backbone. To integrate the response maps generated over convolutional layers, we use a residual connection block to make full use of multiple-level semantic abstraction of target objects. In Fig. 2.16, we compare our scheme with the ECO [34] and CREST [172] methods. The DCFs tracker ECO [34] *independently* learns correlation filters over the *conv1* and *conv5* layers. CREST [172] learns a base and a residual regression network over the *conv4* layer. The proposed method in Fig. 2.16c fuses the *conv4* and *conv5* layers before learning the regression networks. Here we use the deconvolution operation to upsample the $conv5$ layer before connection. We reduce feature channels to ease the computational load as in [40, 72]. Our connection scheme resembles the Option C of constructing the residual network [72]. Ablation studies affirm the effectiveness of this scheme to facilitate regression learning (see Sect. 2.1.6).



(a) ECO (b) CREST (c) Ours

Fig. 2.16 Different schemes to fuse convolutional layers (Sect. 2.1.6). ECO [34] independently learns *correlation filters* over multiple convolutional layers. CREST [172] learns a base and a residual regression network over a single convolutional layer. We first fuse multiple convolutional layers using residual connection and then perform regression learning. Our regression network makes full use of multi-level semantics across multiple convolutional layers rather than merely integrating response maps as ECO and CREST

Classification Tracking with Shrinkage Loss

To further demonstrate the effectiveness of the proposed shrinkage loss for classification learning, we apply our shrinkage loss to the well-known Siamese tracking framework [183]. We take the representative SiameseFC [15], SiamRPN [110], SiamRPN++ [109] and MDNet [88] trackers as baseline due to their clear architectures and state-of-the-art performances.

SiameseFC. The SiameseFC tracker formulates object tracking as a frame-by-frame matching problem. Let z' denote the template (i. e., target to be tracked) given in the first image and x' denote the search area in the subsequent frames. The matching similarity score can be computed by a cross correlation operation between two streams with the feature embedding $\phi(\cdot)$:

$$g(z', x') = \phi(z') \star \phi(x') + b, \tag{2.35}$$

where $\star$ means cross correlation and b means the bias. As ϕ is implemented via fully convolution networks, the output $g(z', x')$ preserves the spatial information in which each element reflects the similarity score between the target image and the search region. As a result, the position of the maximum score relative to the center of the score map reflects the displacement of the target frame-by-frame. To determine whether the target matches with the search region, SiameseFC learns a binary classifier for each element. The binary cross entropy (BCE) loss between the label $y \in [0, 1]$ and the prediction score p is formulated as:

$$\mathcal{L}_\phi(\mathbf{Y}, \mathbf{P}) = \sum_{i \in S} -\left(y_i \log(p_i) + (1 - y_i) \log(1 - p_i)\right), \tag{2.36}$$

where $p = \text{sigmoid}(\cdot)$ is the network final prediction.

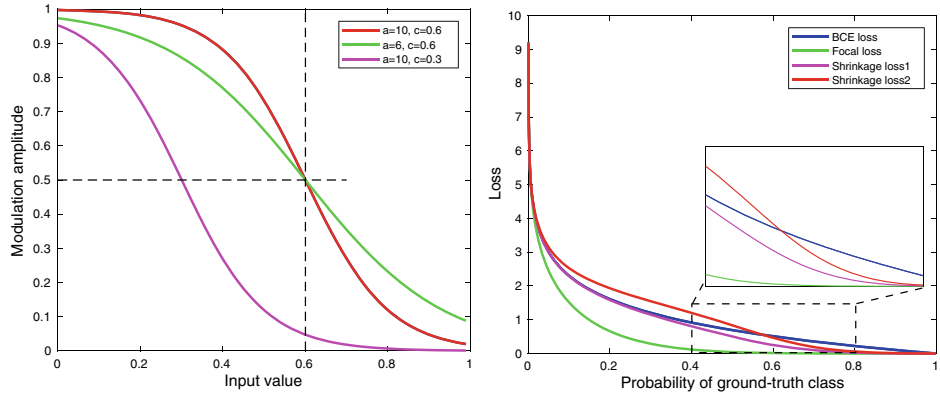
Figure 2.18 shows that there exists class imbalance between the positive and negative labels. To alleviate this issue, SiameseFC uses the class ratio as weights to balance the loss:

$$\mathcal{L}_\phi(\mathbf{Y}, \mathbf{P}) = \sum_{i \in S} -\left(y_i \beta_1 \log(p_i) + (1 - y_i) \beta_2 \log(1 - p_i)\right), \tag{2.37}$$

where β_1 and β_2 denote the negative and positive sample ratio in each training batch. To further handle data imbalance, we first apply the recent focal loss [122] to Eq. 2.37. We reformulated the weighted BCE loss (Eq. 2.37) in the focal loss type as follows:

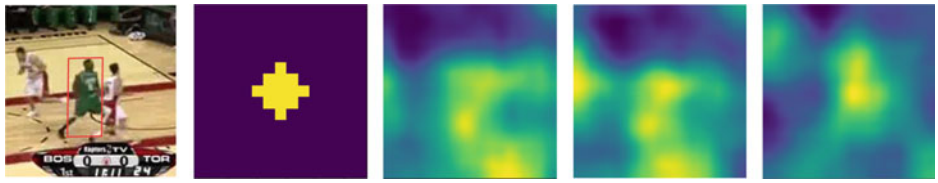
$$\mathcal{L}_F(y, p) = -\alpha(1-p_t)^\gamma \log(p_t) = \begin{cases} -\beta_1(1-p)^\gamma \log(p) & y=1 \\ -\beta_2 p^\gamma \log(1-p) & y=0. \end{cases} \tag{2.38}$$

To avoid setting an empirical value to α for controlling the positive loss and negative as the original focal loss does, we use the values of β_1 and β_2 in Eq. 2.37.



(a) Modulating factor (b) Loss comparison

Fig. 2.17 **a** Modulating factors in Eq. 2.39 with different hyper-parameters (Sect. 2.1.6). **b** Comparison between the BCE loss, focal loss and the proposed shrinkage loss for Siamese network learning. The proposed shrinkage loss only decreases the loss from easy samples ($l < 0.5$) and keeps the loss from hard samples ($l > 0.5$) almost unchanged (pink) or even greater (red) for hard samples



(a) Input (b) Binary labels (c) SiameseFC (d) SiamFC_fl (e) SiamFC_sl

Fig. 2.18 Response maps with different loss functions. **a** The region of interest. **b** Ground truth map. **c**, **d** and **e** show the response maps of SiameseFC with the original loss, focal loss and the proposed shrinkage loss. See Sect. 2.1.6 for details

Similar to Eq. 2.33, we apply the proposed shrinkage loss to the weighted BCE loss (Eq. 2.37) by multiplying the modulating function:

$$f(p_t) = \frac{1}{1 + \exp(a \cdot (p_t - c))}. \tag{2.39}$$

Figure 2.17d demonstrates that the proposed shrinkage loss helps penalize the loss of easy samples while preserving the loss of hard samples. In comparison, focal loss partially penalizes the loss of hard samples. Furthermore, we take the cost-sensitive weighting strategy and apply a weight factor to Eq. 2.39 as:

$$f(p_t) = \frac{\exp(p_t)}{1 + \exp(a \cdot (p_t - c))}. \tag{2.40}$$

In this way, the loss of easy samples can be suppressed while the loss for hard samples can be enhanced. Overall, the shrinkage loss for the BCE loss can be written as:

$$\mathcal{L}_S(y,\, p) = \begin{cases} -\frac{\beta_1 \exp(p)}{(1+\exp(a\cdot(p-c))} \log(p) & y = 1 \\ -\frac{\beta_2 \exp(1-p)}{(1+\exp(a\cdot((1-p)-c))} \log(1-p) & y = 0. \end{cases} \tag{2.41}$$

The values of a and c are set to 10 and 0.6, respectively. From Fig. 2.18, we can see that, compared to the response map of BCE loss in (c), focal loss penalizes partial contribution of hard samples as well, resulting in false large response for the distractor (d). However, equipped with the proposed shrinkage loss, SiamFC_sl can separate the targets from distractors and the background (e).

SiamRPN. The SiamRPN [110] tracker extends SiameseFC by incorporating an additional region proposal network (RPN) from the FastRCNN detector [160]. The outputs of SiamRPN include one classification branch $[\phi(x)]_{cls}$ for scoring each proposal and one regression branch $[\phi(x)]_{reg}$ for locating each proposal:

$$\begin{aligned} A^{cls}_{w\times h\times 2k} &= [\phi(x)]_{cls} \star [\phi(z)]_{cls} \\ A^{reg}_{w\times h\times 4k} &= [\phi(x)]_{reg} \star [\phi(z)]_{reg}, \end{aligned} \tag{2.42}$$

where k is the number of anchors in the RPN. Similar to deep regression trackers in Fig. 2.14, each element on the feature map corresponds to one anchor. The only difference lies in that the bounding boxes in deep regression tracking are generated with a fixed scale and aspect ratio while the anchors in the RPN are generated with varying scales and aspect ratios. As a result, for the classification branch, there exists heavy class imbalance between positive and negative samples. We apply the proposed shrinkage loss to train the classification branch as in Eq. 2.41. For the regression branch, we keep the original smooth $\mathcal{L}_1$ loss. Finally, the overall loss is composed of the classification loss $\mathcal{L}_{cls}$ and the regression loss $\mathcal{L}_{reg}$ for training the whole network:

$$\mathcal{L}_{all} = \mathcal{L}_{cls} + \eta \mathcal{L}_{reg}, \tag{2.43}$$

where $\mathcal{L}_{cls}$ denotes the proposed shrinkage loss in Eq. 2.41 and $\mathcal{L}_{reg}$ denotes the smooth $\mathcal{L}_1$ loss, and η is a hyper-parameter to balance these two losses. We name the new network as SiamRPN_sl.

MDNet. The MDNet [88, 149] tracker also adopts the binary cross entropy loss (Eq. 2.36) to optimize the whole network:

$$\mathcal{L}_{cls} = -\frac{1}{N} \sum_{i=1}^{N} (y_i \log(p_i) + (1 - y_i) \log(1 - p_i)), \tag{2.44}$$

where N is the total number of positive and negative samples, p denotes the prediction and y_i indicates a binary label. Similar to Eq. 2.41, we apply our shrinkage loss to the real-time version of MDNet [88] to handle data imbalance more effectively. Formally, we formulate our shrinkage loss as:

$$\begin{aligned}\mathcal{L}_{cls} = -\frac{1}{N}\sum_{i=1}^{N}(&\frac{\exp(p_i)}{1+\exp(a\cdot(p_i-c))}y_i\log(p_i)+\\ &\frac{\exp(1-p_i)}{1+\exp(a\cdot(1-p_i)-c)}(1-y_i)\log(1-p_i)).\end{aligned} \tag{2.45}$$

Tracking Framework

Deep Regression Tracking. We first detail the pipeline of the proposed regression tracker. In Fig. 2.13, we show an overview of the proposed deep regression network, which consists of model initialization, target object localization, scale estimation and model update. For training, we crop a patch centered at the estimated location in the previous frame. We use the ResNet-50 [72] model as the backbone feature extractor. Specifically, we take the activation from the third and fourth convolutional blocks as features. We fuse features via residual connection and then feed them into the proposed regression network. During tracking, given a new frame, we crop a search patch centered at the estimated position in the last frame. The regression network takes this search patch as input and outputs a response map, where the location of the maximum value indicates the position of target objects. Once obtaining the estimated position, we carry out scale estimation using the scale pyramid strategy from [35]. To adapt to appearance variations, we incrementally update our regression network frame-by-frame. We use the tracked results and soft labels in the last T frames for model updating.

Deep Classification Tracking. We follow the original implementation of SiameseFC [15] and take the AlexNet [104] model to build the two-stream fully convolutional network as feature embedding. During offline training, we use the ImageNet Video dataset [165] to train the whole network. In each iteration, we crop one training pair from the same video. Each pair consists of an exemplar patch (i.e., template) and the corresponding search region. We feed the image pair into the Siamese network and optimize the feature embedding $\phi(\cdot)$ by minimizing the proposed shrinkage loss (Eq. 2.41). The exemplar patch is cropped in a fixed size of 127×127 pixels and the search region is of 255×255 pixels including partial surrounding context. During the tracking process, given the cropped target object in the initial frame, we crop a search region in the current frame and feed this image pair into the SiameseFC tracker. The maximum response of the score map indicates the location of the target object. We employ a multi-scale strategy to handle scale changes. For fair comparison, we do not use online fine-tuning as in [15]. For the SiamRPN tracker, we use AlexNet [104] as the backbone feature network. Following the protocol of focal loss [122], we use all the

proposals to train the network rather than selecting the proposals with an IOU score smaller than 0.3 or larger than 0.6 as in [110].

Experimental Results

In this section, we first elaborate the implementation details in Sect. 2.1.6. We then evaluate the proposed methods on six benchmark datasets including OTB-2013 [206], OTB-2015 [207], UAV-123 [146], VOT-2016 [100], VOT-2018 [101] and LaSOT [55] in comparison with state-of-the-art trackers in Sects. 2.1.6 and 2.1.6. Finally, in Sect. 2.1.6, we present extensive ablation studies on different types of losses as well as their effect on the convergence speed of network training.

Implementation Details

We implement the proposed deep regression tracker DSLT* in Matlab using the Caffe toolbox [86]. All experiments are performed on a PC with an Intel i7 4.0GHz CPU and an NVIDIA TITAN X GPU. We apply a 1×1 convolution layer to reduce the channels of $Res3d$ and $Res4f$ from 512 and 1024 to 128, respectively. We train the regression networks with the Adam [99] algorithm. Considering the large gap between the maximum values of the output regression maps over different layers, we set the learning rate η in the $Res4f$ and $Res3d$ layers to $8e$-7 and $2e$-8. During online updating, we decrease the learning rates to $2e$-7 and $5e$-9, respectively. The length of frames T for model updating is set to 7. The soft labels are generated by a two-dimensional Gaussian function with a kernel width proportional to (0.1) the target size. For scale estimation, we set the ratio of scale changes to 1.03 and the levels of scale pyramid to 3. The average tracking speed including all training process is 6.3 frames per second. We implement the Siamese tracking (i.e., SiamFC_sl, SiamRPN_sl and SiamRPN++) based on the open PySOT toolkit[1] and RT-MDNet on Pytorch. All the training setting is the same as the original SiameseFC and SiamRPN trackers. Specifically, we use the ILSVRC VID [165] to train SiameseFC and RT-MDNet. For SiamRPN, we use the ILSVRC VID [165], MS-lin2014microsoft [123], ILSVRC Det [165] and Youtube-bounding box [159] as training data. The average running speeds of the improved SiameseFC, SiamRPN and RT-MDNet trackers are 45.7, 80.3 and 42.7 in FPS. All the source code of this project is available at https://github.com/chaoma99/DSLT.

Overall Performance of Regression Tracking

We extensively evaluate the proposed one-stage regression tracker on five challenging tracking benchmarks. We follow the protocol of the benchmarks for fair comparison with state-of-the-art trackers. For the OTB [206, 207] datasets, we report the results of one-pass evaluation (OPE) with distance precision (DP) and overlap success (OS) plots. The legend of distance

[1] https://github.com/STVIR/pysot/.

precision plots contains the thresholded scores at 20 pixels, while the legend of overlap success plots contains area-under-the-curve (AUC) scores for each tracker.

OTB Dataset. There are two versions of this dataset. The OTB-2013 [206] dataset contains 50 challenging sequences and the OTB-2015 [207] dataset extends the OTB-2013 dataset with additional 50 video sequences. All the sequences cover a wide range of challenges including occlusion, illumination variation, rotation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutter and low resolution. We fairly compare the proposed DSLT* method with the state-of-the-art trackers, which mainly fall into three categories: (i) one-stage regression trackers including DSLT [130], CREST [172], FCNT [194], GOTURN [73], SiameseFC [15]; (ii) one-stage DCFs trackers including ECO [34], C-COT [41], BACF [94], DeepSRDCF [36], ma2015hierarchical [134], HDT [158], SRDCF [39], KCF [74], and MUSTer [80]; and (iii) two-stage trackers including SiamRPN++ [109], SiamRPN [110], MEEM [217], TGPR [61], SINT [183], and CNN-SVM [78]. For experimental completeness, we also report the tracking performance of the proposed SiamRPN_sl. As shown in Fig. 2.19, the proposed DSLT* achieves the best distance precision (94.1%) and the second best overlap success (67.6%) on OTB-2013. Our method outperforms the state-of-the-art deep regression trackers (CREST [172] and FCNT [194]) by a large margin. We attribute the favorable performance of our DSLT* to two reasons. First, the proposed shrinkage loss effectively alleviates the data imbalance issue in regression learning. As a result, the proposed method can automatically mine the most discriminative samples and eliminate the distraction caused by easy samples. Second, we exploit the residual connection scheme to fuse multiple convolutional layers to further facilitate regression learning as multi-level semantics across convolutional layers are fully exploited. As well, our DSLT* performs favorably against all DCFs trackers such as C-COT, ma2015hierarchical and DeepSRDCF. Note that ECO achieves the best results by exploring both deep features and hand-crafted features. On OTB-2015, our method ranks second in both distance precision and overlap success. Meanwhile, our SiamRPN_sl outperforms SiamRPN significantly in terms of precision (+2.3%) and AUC scores (+1.6%).

UAV-123 Dataset. This dataset [146] contains 123 video sequences obtained by unmanned aerial vehicles (UAVs). Some tracking targets belong to the small object and undergo long term occlusion. We evaluate the proposed DSLT* with several representative methods including SiamRPN++ [109], SiamRPN [110], ECO [34], DSLT [130], SRDCF [39], KCF [74], MUSTer [80], MEEM [217], TGPR [61], SAMF [119], DSST [183], CSK [75], Struck [68], and TLD [90]. Figure 2.20 shows that the performance of the proposed DSLT* is slightly superior to ECO in terms of distance precision and overlap success rate. Furthermore, equipped with the proposed shrinkage loss, our implemented SiamRPN_sl method achieves a new state-of-the-art in location precision (78.1%) and AUC (56.9%).

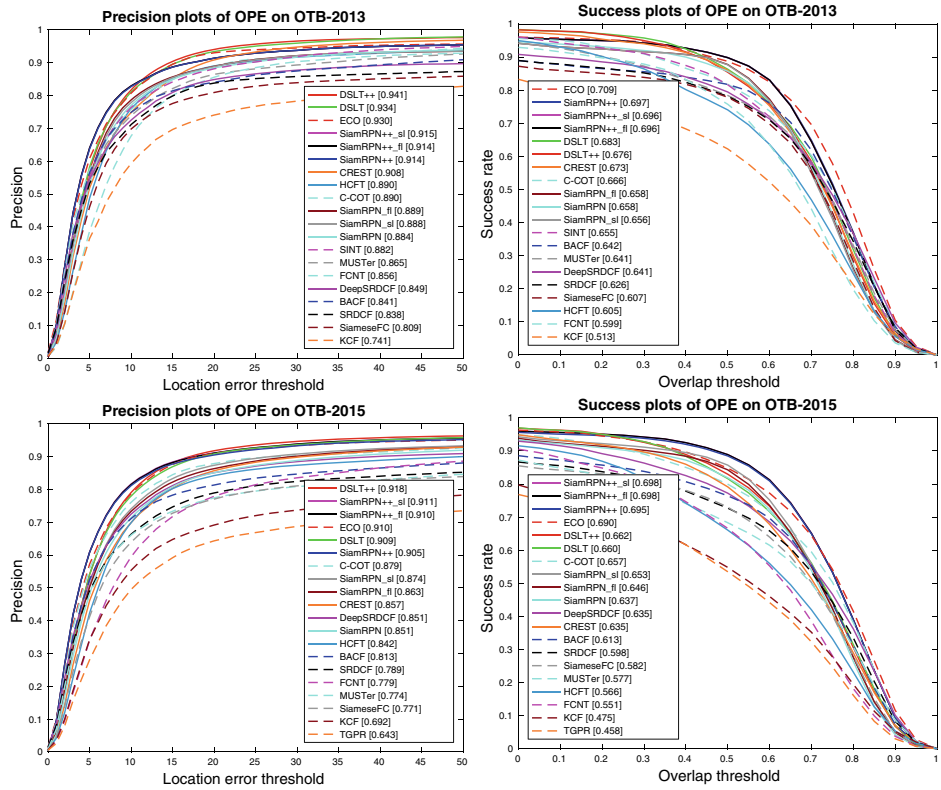


Fig. 2.19 Overall performance on the OTB-2013 [206] and OTB-2015 [207] datasets using one-pass evaluation (OPE). The legend of distance precision contains the threshold scores at 20 pixels while the legend of overlap success contains area-under-the-curve score for each tracker. Our tracker performs well against state-of-the-art methods

VOT-2016 Dataset. The VOT-2016 [100] dataset contains 60 challenging videos, which are annotated by the following attributes: occlusion, illumination change, motion change, size change, and camera motion. The overall performance is measured by the expected average overlap (EAO), accuracy (A) and failure (F). We compare our method with state-of-the-art trackers from the VOT-2016 benchmark including SiamRPN++ [109], SiamRPN [110], ECO [34], DSLT [130], C-COT [41], CREST [172], Staple [14], SRDCF [39], Deep-SRDCF [36], MDNet [149]. Table 2.6 shows that our method performs better than most compared methods such as C-COT and CREST. The VOT-2016 report [100] suggests a strict state-of-the-art bound as 0.251 with the EAO metric. The proposed DSLT* achieves a much higher EAO of 0.364. Meanwhile, our implemented SiamRPN_sl method achieves a slight performance gain when comparing to original SiamRPN. Equipped with the proposed shrinkage loss, SiamRPN++_sl yields an EAO score of 0.481 that sets a new state-of-the-art.

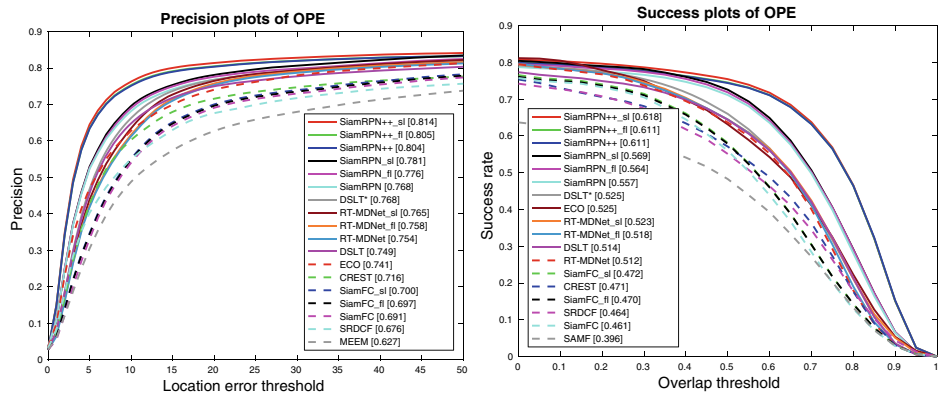


Fig. 2.20 Overall performance on the UAV-123 [146] dataset using one-pass evaluation (OPE). The legend of distance precision contains the threshold scores at 20 pixels while the legend of overlap success contains area-under-the-curve score for each tracker. The proposed DSLT* method ranks first among the regression based methods

According to the definition of the VOT report, all these trackers are state-of-the-art. Detailed comparison can be seen in Fig. 2.21.

Overall Performance of Classification Tracking

To evaluate the generalization ability of the proposed shrinkage loss, we replace the original BCE loss of SiameseFC [15] and RT-MDNet [88] with focal loss (SiamFC_fl, RT-MDnet_fl) and our shrinkage loss (SiamFC_sl, RT-MDNet_sl). We fairly compare these four approaches with the representative Siamese tracking methods: SiameseFC [15], CFNet [187]; correlation filter based trackers: Staple [14], LCT [135], DSST [112], KCF [74] and other popular classification based trackers: TGPR [61], Struck [68], CNN-SVM [78]. Figure 2.22 shows the overall tracking results on the OTB-2015 dataset. The RT-MDNet_sl method performs well among all the compared methods. Compared to original RT-MDNet, RT-MDNet_sl achieves large performance gains of distance precision (+1.9%) and overlap success (+0.9%). Meanwhile, we observe our SiamFC_sl surpasses SiameseFC across distance precision (+2.5%) and overlap success (+0.7%). As these methods adopt the same training and test protocols, we attribute the performance improvement solely to the proposed shrinkage loss which helps handle data imbalance during network learning. In contrast, focal loss causes SiamFC_fl to perform worse than the origin SiameseFC tracker with losses of −0.6% in distance precision and −0.8% in overlap success. The reason is that focal loss penalizes not only easy samples but also hard samples which are crucial to learning Siamese networks.

Table 2.6 Overall performance on the VOT-2016 [100] dataset in comparison with the top 10 trackers. EAO: Expected average overlap. A: Accuracy, F: Failure. Best results are in **bold**

Tracker	EAO ↑	Accuracy ↑	Failure ↓
SiamRPN++_sl	**0.481**	**0.634**	11.37
SiamRPN++_fl	0.478	0.632	11.93
SiamRPN++ [109]	0.477	0.632	11.58
RT-MDNet_sl	0.372	0.558	10.94
RT-MDNet_fl	0.369	0.552	11.77
RT-MDNet [88]	0.370	0.554	11.52
DSLT*	0.364	0.551	**9.253**
DSLT [130]	0.332	0.541	15.48
SiamRPN_sl	0.356	0.581	18.09
SiamRPN [110]	0.340	0.579	20.13
ECO [34]	0.374	0.546	11.67
C-COT [41]	0.331	0.539	16.58
TCNN [148]	0.325	0.554	17.93
SSAT [100]	0.321	0.577	19.27
MLDF [100]	0.311	0.490	15.04
Staple [14]	0.295	0.544	23.89
SiamRN [100]	0.277	0.547	23.99
CREST [172]	0.283	0.550	25.10
DeepSRDCF [36]	0.276	0.529	20.34
SiamFC_sl	0.242	0.534	29.55
SiamFC_fl	0.236	0.531	29.78
SiamFC [15]	0.235	0.531	29.80
MDNet [149]	0.257	0.544	21.08
SRDCF [39]	0.247	0.544	28.31

To further demonstrate the effectiveness of the proposed shrinkage loss, we replace the original loss function of the SiamRPN++ [109], SiamRPN [110] tracker with our shrinkage loss (SiamRPN++_sl and SiamRPN_sl) and focal loss (SiamRPN++_fl and SiamRPN_fl) respectively. We mainly evaluate these two approaches on both the VOT-2018 [101] and LaSOT [55] datasets.

VOT-2018 Dataset. The VOT-2018 [101] is a recent dataset for evaluating online tracking methods. It contains 60 video sequences with different challenging attributes: (1) full occlusion, (2) out-of-view motion, (3) partial occlusion, (4) camera motion, (5) fast motion, (6) scale change, (7) aspect ratio change, (8) viewpoint change, (9) similar objects. Following the

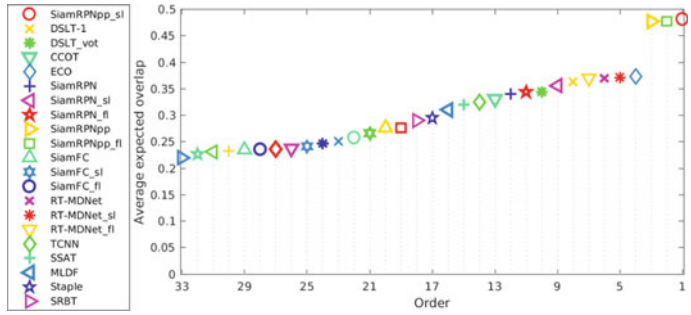


Fig. 2.21 Overall performance on the VOT-2016 [100] dataset using expected average overlap graph. SiamRPNpp denotes SiamRPN++ [109]

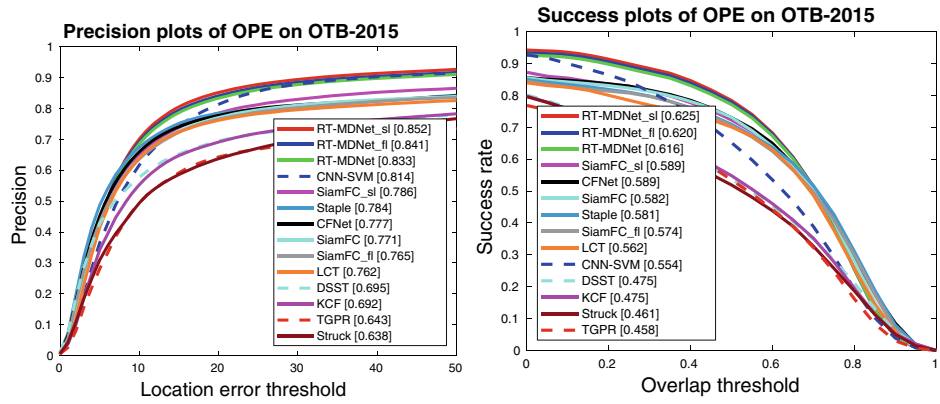


Fig. 2.22 Overall performance on the OTB-2015 [207] dataset using one-pass evaluation (OPE). Equipped with the proposed shrinkage loss, RT-MDNet_sl ranks first among the compared methods

evaluation protocol of VOT-2018, we adopt the Expected Average Overlap (EAO), Accuracy (A) and Robustness (R) as the criteria. Table 2.7 shows that SiamRPN_sl implementation significantly outperforms SiamRPN, achieving an EAO of 0.361 versus 0.352, accuracy of 0.584 versus 0.566, and robustness of 0.183 versus 0.224. Since we adopt the same network architecture, the performance gains are solely achieved by the proposed shrinkage loss, which helps penalize easy negative proposals during network training. When using focal loss, SiamRPN_fl performs better than SiamRPN, but not as well as SiamRPN_sl. This affirms that focal loss can alleviate the data imbalance issue to some extent by penalizing easy training samples, but our shrinkage loss succeeds in maintaining the loss of hard samples almost unchanged and achieves better results. Our shrinkage loss helps SiamRPN++ to achieve larger performance gains than focal loss does. With our shrinkage loss, SiamRPN++_sl sets a new state-of-the-art result on the VOT-2018 dataset. Figure 2.23 presents the EAO rank plots of all the compared trackers as well as the overlap curves. Note that the VOT-2018

Table 2.7 Overall performance on the VOT-2018 [101] dataset in comparison with the state-of-the-art trackers. EAO: expected average overlap. Best result for each item is labeled in **bold**

Tracker	EAO ↑	Accuracy ↑	Robustness ↓
SiamRPN++_sl	**0.440**	0.586	0.179
SiamRPN++_fl	0.419	**0.597**	0.198
SiamRPN++ [109]	0.412	0.592	0.234
SiamRPN_sl	0.361	0.584	0.183
SiamRPN_fl	0.358	0.582	0.194
SiamRPN [110]	0.352	0.566	0.276
DSLTpp [195]	0.325	0.543	0.224
SA_Siam_R [69]	0.337	0.566	0.258
CPT [101]	0.339	0.506	0.239
ECO [34]	0.280	0.484	0.276
DSLT*	0.274	0.500	0.279
DSLT [130]	0.263	0.489	0.281
DeepSTRCF [111]	0.345	0.523	0.290
LSART [176]	0.323	0.495	0.258
SiamFC_sl	0.190	0.502	0.582
SiamFC_fl	0.188	0.501	0.584
SiamFC [15]	0.188	0.500	0.585
CSRDCF [132]	0.256	0.491	0.378
SRCT [101]	0.310	0.520	**0.159**
RT-MDNet_sl	0.182	0.525	0.561
RT-MDNet_fl	0.179	0.522	0.566
RT-MDNet [88]	0.178	0.522	0.567
CFTR [17]	0.300	0.505	0.184

dataset is much challenging than VOT-2016, the performance of most regression based methods, such as ECO [34], CSRDCF [132] as well as our DSLT*, is inferior to the classification based methods.

LaSOT Dataset. The LaSOT [55] dataset is a recently released large-scale dataset for training and testing single object trackers. There are 1,400 videos in total containing 69 categories of objects including *airplane*, *bicycle* and *zebra*, etc. The average video length is about 2500 frames with challenging factors such as occlusion, deformation, out-of-view and motion blur. Following the OTB-2015 [207] protocol, LaSOT uses One-Pass Evaluation (OPE) with the distance precision, and success rate as the evaluation metrics. We evaluate the proposed methods on LaSOT in comparison with 30 trackers,

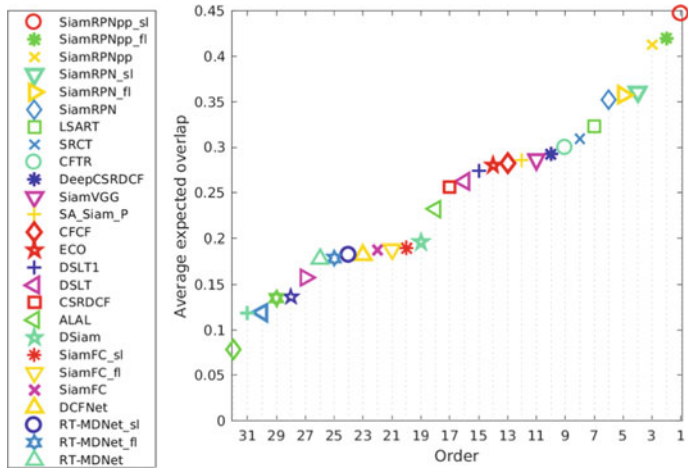


Fig. 2.23 Expected average overlap results on the VOT-2018 [101] dataset. SiamRPNpp denotes SiamRPN++

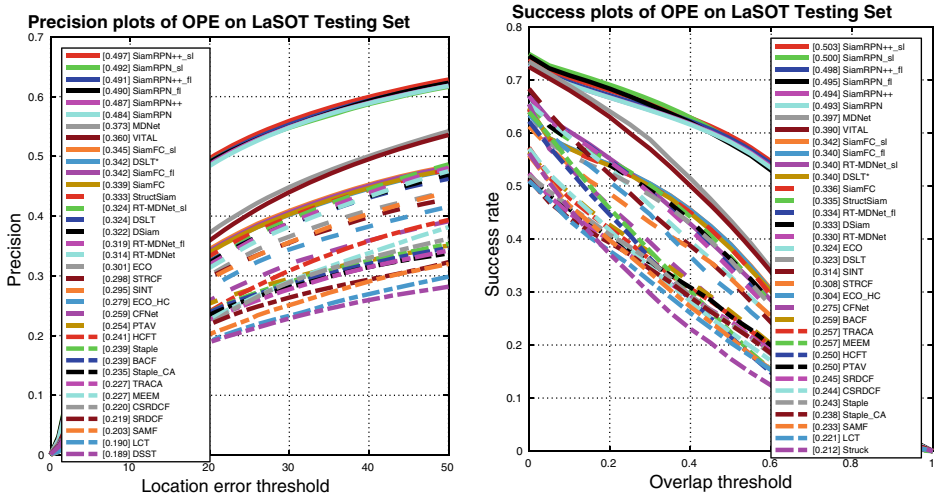


Fig. 2.24 Comparisons with the state-of-the-art methods on the LaSOT [55] dataset using one-pass evaluation (OPE). Equipped with the proposed shrinkage loss, the SiamRPN++_sl method ranks first

including SiamRPN++ [109], SiamRPN [110], VITAL [173], ECO [34], DSLT* (ours), DSLT [130], MDNet [149], SiameseFC [15], CFNet [187], BACF [94], SRDCF [36], ma2015hierarchical [134], etc. Figure 2.24 illustrates the overall tracking results. With the use of shrinkage loss, SiamRPN++_sl advances the performance of the original SiamRPN++ tracker by a significant margin (+1.0% in DP and +0.9% in AUC) and achieves the new state-of-the-art results on the LaSOT dataset (49.7% and 50.3%). Equipped with focal loss,

SiamRPN++_fl also achieve performance gains when compared to origin ones. However, the performance gap between SiamRPN++_fl and Siam-RPN++_sl clearly demonstrates our shrinkage loss is more effective in handling data imbalance. Benefiting from the offline training with large-scale annotated data in LaSOT, most offline based methods, such as SiamRPN++[109], SimaRPN[110] and VITAL[173], surpass methods without offline training (e.g., DSLT* (ours), DSLT [130] and BACF [94]).

Ablation Studies

In this section, we first analyze the contributions of the loss function and the effectiveness of the residual connection scheme. We then discuss the convergence speed of different losses in regression learning.

Shrinkage Loss Parameters Analysis. We first offer more in-depth performance analysis for our shrinkage loss. We report the comprehensive performances with different hyper-parameters on the combined pool of the OTB-2015 [207] and UAV-123 [146] datasets. Table 2.8 summarizes the ablation results. We can see that removing the importance factor $\exp(Y)$ in Eqs. 2.34 and 2.41 yields a slight performance drop (84.3→84.0, 59.9→59.7). While disabling the modulating function leads to large performance degradation (84.3→80.6, 59.9→58.1). We can draw the similar conclusion from the results of SiamRPN_sl. In addition, we study the impact of hyper-parameters (a and c) of our shrinkage loss. We observe that the parameter c, which controls the importance of hard samples, is more sensitive to the final performance than the parameter a, which controls the shrinkage speed. We present more hyper-parameter analysis of focal loss. We observe that the best performance can be obtained when $\alpha = 0.5, \gamma = 2$.

Loss Function Analysis. Next, we replace the proposed shrinkage loss with square loss ($\mathcal{L}_2$) or $\mathcal{L}_3$ loss. We evaluate the alternative implementations on the OTB-2015 [207] dataset. Overall, the proposed DSLT* method with shrinkage loss significantly advances the square loss ($\mathcal{L}_2$) and $\mathcal{L}_3$ loss by a large margin. Figure 2.26 presents the quantitative results on the OTB-2015 dataset. Note that the baseline tracker with $\mathcal{L}_2$ loss performs much better than CREST [172] in both distance precision (87.0% vs. 83.8%) and overlap success (64.2% vs. 63.2%). This clearly proves the effectiveness of the convolutional layer connection scheme, which applies residual connection to both convolutional layers and output regression maps rather than only to the output regression maps as CREST does. In addition, we implement an alternative approach using online hard negative mining (OHNM) [149, 168] to completely exclude the loss from easy samples. We empirically set the mining threshold to 0.01. Our DSLT* outperforms the OHNM method significantly. Our observation is thus well aligned to [122] that easy samples still contribute to regression learning but they should not dominate the whole gradient. In addition, the OHNM method manually sets a threshold, which is hardly applicable to all videos. Moreover, we present the qualitative results with prediction scores on

Table 2.8 Ablation studies on the combined pool of the OTB-2015 and UAV-123 datasets under the regression and classification cases. We report distance precision (DP) and AUC scores

Ablation cases of shrinkage loss		DP(%)↑	AUC(%)↑
Regression case (DSLT*)	w/o exp(Y) in Eq. 2.34	84.0	59.7
	Only exp(Y) in Eq. 2.34	80.6	58.1
	Different values of a and c in Eq. 2.34		
	a = 10, c = 0.2	**84.3**	**59.9**
	a = 6, c = 0.2	84.0	59.7
	a = 10, c = 0.6	83.7	59.6
	a = 6, c = 0.6	83.1	59.0
Classification case (SiamRPN_sl)	w/o exp(p_t) in Eq. 2.41	82.4	60.9
	Only exp(p_t) in Eq. 2.41	81.1	59.8
	Different values of a and c in Eq. 2.41		
	a = 10, c = 0.6	**82.7**	**61.1**
	a = 6, c = 0.2	82.4	60.7
	a = 10, c = 0.2	82.2	60.6
	a = 6, c = 0.6	81.9	60.4
Ablation cases of focal loss		DP(%) ↑	AUC(%) ↑
Classification case (SiamRPN_fl)	Different values of α and γ in Eq. 2.38		
	$\alpha = 0.5, \gamma = 0.5$	81.1	59.8
	$\alpha = 1, \gamma = 0.5$	81.4	59.9
	$\alpha = 0.5, \gamma = 1$	81.2	59.8
	$\alpha = 1, \gamma = 1$	81.7	60.2
	$\alpha = 0.5, \gamma = 2$	81.8	60.4
	$\alpha = 1, \gamma = 2$	**82.0**	**60.5**

the challenging Motorcycle-3 sequence [55] in Fig. 2.25. Specifically, in the 631st frame, our SiamRPN_sl yields a more accurate prediction than SiamRPN_fl and SiamRPN. In the 702nd frame, we can see that both SiamRPN with binary cross entropy loss and SiamTPN_fl fail to track the target undergoing large appearance changes, whereas the proposed SiamRPN_sl can locate the targets robustly.

Feature Analysis. We report the influence of backbone feature networks: ResNet50 [72] and VGG16 [170]. With the use of ResNet50, DSLT* achieves better performance than its

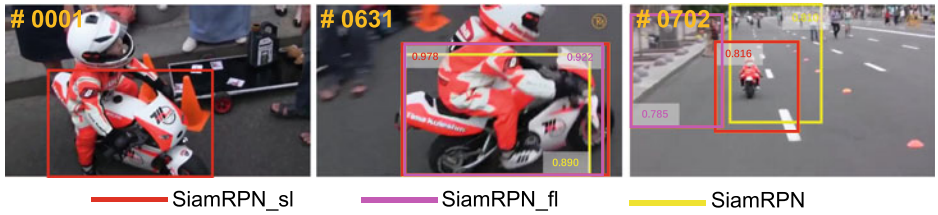


Fig. 2.25 Quantitative results on the Motorcycle-3 sequence [55]. The predictions of SiamRPN_sl, SiamRPN_fl and SiamRPN are in red, pink and yellow, respectively. SiamRPN_sl method with the proposed shrinkage loss can locate the targets more robustly than that with focal loss and BCE loss

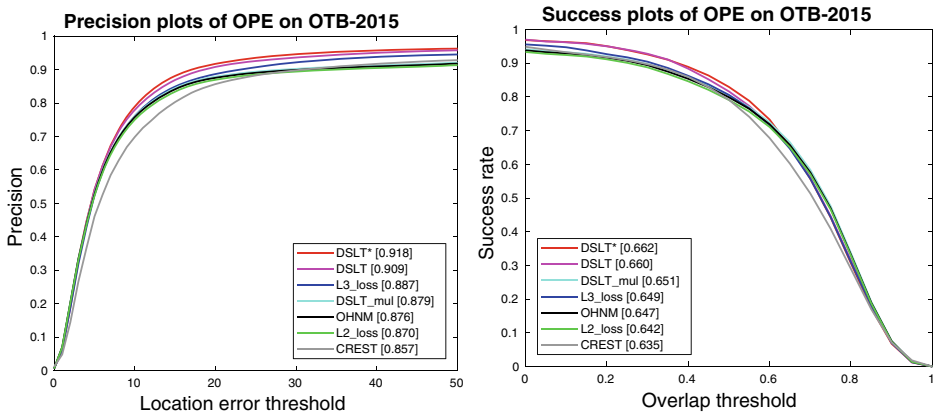


Fig. 2.26 Ablation studies with different losses and different layer connections on the OTB-2015 [207] dataset

early version DSLT [130]. We further evaluate the effectiveness of convolutional layers. For fair comparison, we use the VGG16 [170] as the backbone network. We first remove the connections between convolutional layers. The resulted DSLT_mul algorithm resembles the CREST [172]. Figure 2.26 shows that DSLT_mul has performance drops of around 0.3% (DP) and 0.1% (OS) when compared to DSLT. This affirms the importance of fusing features before regression learning.

Convergence Speed. Following the protocol in previous works [12, 127] about loss functions, Fig. 2.27 compares the convergence plots and the required training iterations using different losses on the OTB-2015 [207] dataset. We also report the AUC scores on the validation set of LaSOT [55] over the fair training with our shrinkage loss, focal loss and the origin BCE loss. Overall, the training loss using the shrinkage loss descends quickly and stably. The shrinkage loss thus requires the least iterations to converge during tracking.

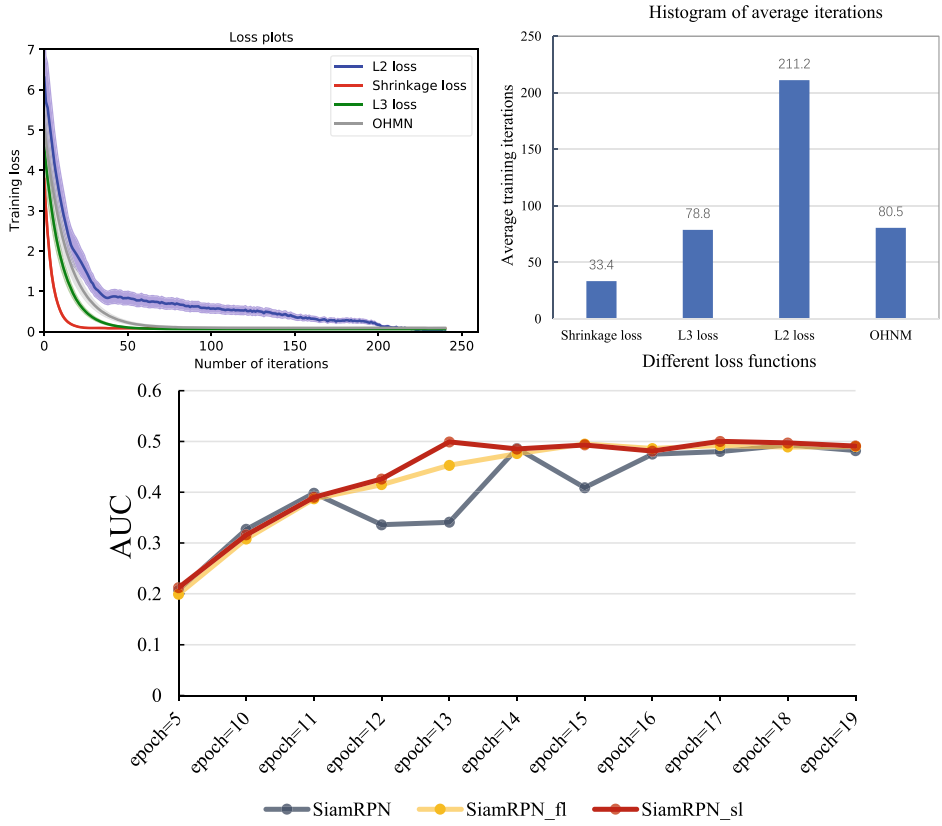


Fig. 2.27 Training loss plots in terms of average curves with deviations (top left), average training iterations per sequence (top right) and AUC plots (bottom)

Qualitative Evaluation

Figure 2.28 visualizes the tracking results of the proposed DSLT* method on six challenging sequences in comparison with the top performing trackers including ECO [34], C-COT [41], CREST [172] and ma2015hierarchical [134]. The ma2015hierarchical does not perform well in most presented sequences. It is because ma2015hierarchical empirically weights the response maps of multiple layers and does not incorporate a sample re-weighting strategy as in ECO and C-COT. For CREST, drift occurs on both the *Lemming* and *Bolt2* sequences. Despite a similar residual scheme to fuse multiple response maps, CREST cannot handle the background distractions well as the used square loss is unaware of data imbalance. Moreover, a single convolution layer in CREST is insufficient to capture the rich semantics. On the other hand, both the ECO and C-COT trackers use multiple convolutional layers as well as hand-crafted features. They integrate multiple correlation response maps and rely on a post-processing location refinement. Despite the overall favorable performance, ECO

Fig. 2.28 Qualitative evaluation. We show tracking results of the CREST [172], ma2015hierarchical [134], C-COT [41], ECO [34] and our method on six challenging video sequences (from left to right and top to bottom are *Lemming*, *Skating1*, *Girl2*, *Bolt2*, *Human9*, and *Soccer*, respectively)

and C-COT do not perform well in the presence of heavy motion blur. These methods both drift in the 374th frame of the *Lemming* sequence. The proposed DSLT* performs well on all these sequences. The proposed shrinkage loss can effectively suppress the target-similar samples from the background. This helps our DSLT* handle heavy occlusion (*Lemming*, *Skating1*) and background clutter (*Soccer*) well, let alone the motion blur (*Human9*) and deformation (*Bolt2*).

Conclusion

In this paper, we addressed the data imbalance issue for learning deep models for robust visual object tracking. We first revisited one-stage trackers based on deep regression networks and identify the performance bottleneck of one-stage regression trackers as the data imbalance issue in regression learning, which impedes one-stage regression trackers from achieving state-of-the-art results, especially when compared to DCFs trackers. To break the performance bottleneck, we proposed the novel shrinkage loss to facilitate learning regression networks with better accuracy and faster convergence speed. To further improve regression learning, we exploited multi-level semantic abstraction of target objects across multiple convolutional layers as features. We applied the residual connections to both convolutional layers and their output response maps. We succeed in narrowing the performance gap between one-stage deep regression trackers and DCFs trackers. Moreover, we showed that the proposed shrinkage loss is effective in addressing the class imbalance issue in classification learning as well. We took the Siamese trackers and RT-MDNet as baseline algorithms and investigated the class imbalance issue during the off-line learning process. We applied the proposed shrinkage loss to facilitate learning classification tracking net-

works. With the use of our shrinkage loss, the baseline Siamese trackers (SiameseFC and SiamRPN) and RT-MDNet both achieved large performance gains. Extensive experiments on six benchmark datasets: OTB-2013, OTB-2015, UAV-123, VOT-2016 as well as recent VOT-2018 and large-scale LaSOT demonstrated the effectiveness and efficiency of the proposed shrinkage loss in alleviating the data imbalance issue in deep object tracking.

2.1.7 Siamese Network Tracking

Motivation

In recent years, Siamese network based trackers have significantly advanced the state-of-the-art in real-time tracking. Despite their success, Siamese trackers tend to suffer from high memory costs, which restrict their applicability to mobile devices with tight memory budgets. To address this issue, we propose a distilled Siamese tracking framework to learn small, fast and accurate trackers (students), which capture critical knowledge from large Siamese trackers (teachers) by a teacher-students knowledge distillation model. This model is intuitively inspired by the one teacher versus multiple students learning method typically employed in schools. In particular, our model contains a single teacher-student distillation module and a student-student knowledge sharing mechanism. The former is designed using a tracking-specific distillation strategy to transfer knowledge from a teacher to students. The latter is utilized for mutual learning between students to enable in-depth knowledge understanding (Fig. 2.29).

Distilled Siamese Networks for Visual Tracking

In this section, we present the proposed Distilled Siamese Tracker (DST) framework for high-performance tracking. As shown in Fig. 2.30, the proposed framework consists of two essential stages. First, in Sect. 2.1.7, for a given teacher network, such as SiamRPN, we obtain a "dim" student with a reduced architecture via Deep Reinforcement Learning (DRL). Then, the "dim" student is further simultaneously trained with an "intelligent" student via the proposed distillation model, facilitated by a teacher-students learning mechanism (see Sect. 2.1.7).

"Dim" Student Selection

Inspired by N2N [4] originally introduced for compressing classification networks, we transfer the form of selecting a student tracker with a reduced network architecture to learn an agent with an optimal compression strategy (policy) by DRL. We introduce several new techniques to adapt the standard N2N classification method for Siamese trackers. This includes carefully designed variables in reinforcement learning and a practical tracking performance evaluation strategy for DRL.

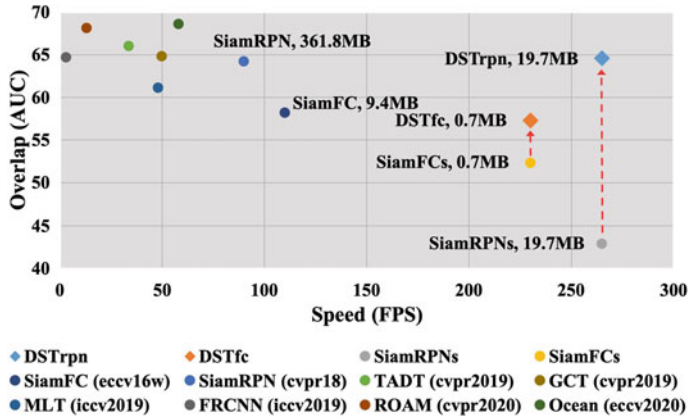


Fig. 2.29 Comparison in terms of speed (FPS) and accuracy (AUC) of state-of-the-art (SOTA) trackers on OTB-100 [207], including SiamRPN [110], SiamFC [15], TADT [118], GCT [62], MLT [28], FRCNN [83], ROAM [215], and Ocean [228]. Compared with the small models trained from scratch (SiamRPNs and SiamFCs), our models (DSTrpn and DSTfc) trained with the knowledge distillation method show significant improvements. Further, DSTrpn achieves a 3× speed, 18× memory compression rate and comparable accuracy over its teacher variant (SiamRPN [110]). Besides, both DSTrpn (SiamRPN [110] as teacher) and DSTfc (SiamFC [15] as teacher) obtain competitive accuracy while achieving the highest speed

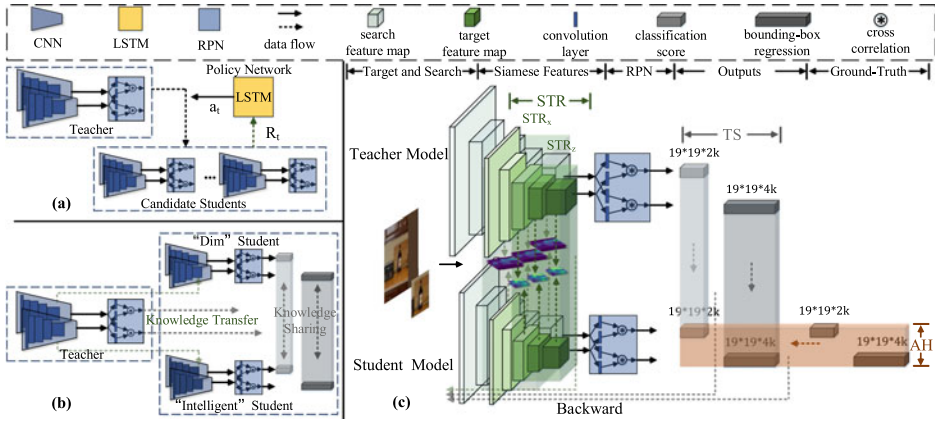


Fig. 2.30 Illustration of the proposed Distilled Siamese Tracker (DST) framework. **a** "Dim" student selection via DRL: at each step t, a policy network guides the generation of candidate students via action a_t and then updates them according to reward R_t. **b** Simplified schematization of our teacher-students knowledge distillation (TSsKD) model, where the teacher transfers knowledge to students, while students share knowledge with each other. **c** Detailed flow chart of teacher-student knowledge transfer with STR, TS and AH loss

Variables in Reinforcement Learning: In our task, the agent for selecting a small and reasonable tracker is learned from a sequential decision-making process by policy gradient DRL. The whole decision process can be modeled as a Markov Decision Process (MDP), which is defined as the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{T}, R, \gamma)$.

The state space $\mathcal{S}$ is a set of all possible reduced network architectures derived from the teacher network. $\mathcal{A}$ is the set of all actions to transform one network into another compressed one. Here, we use layer shrinkage [4] actions $a_t \in [0.1, 0.2, \ldots, 1]$ by changing the configurations of each layer, such as kernel size, padding, and number of output filters. To ensure that the size of the two networks' (teacher and student) feature maps are consistent, we only change the channel numbers of each layer. Moreover, we add a constraint on the RPN part of the SiamRPN sub-networks to force the classification and regression branches to perform the same action. This ensures that the final output channels of the networks remain unchanged after the compression operation. $\mathbb{T}: \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the state transition function, and γ is the discount factor in MDP. To maintain an equal contribution for each reward, we set γ to 1. R is the reward function, which is designed to achieve a trade-off between tracking accuracy and compression rate. The formulation is as follows:

$$R = C(2 - C) \cdot \frac{\text{acc}_s}{\text{acc}_t}, \tag{2.46}$$

where $C = 1 - \frac{S_s}{S_t}$ is the relative compression rate of a student network with size S_s compared to a teacher with size S_t, and acc_s and acc_t are the validation accuracy of the student and teacher networks.

Practical Tracking Evaluation: To obtain an accurate evaluation of the networks' performance, we propose a new tracking performance evaluation strategy. First, we select a fixed number of images from each class of a whole dataset to form a small dataset that includes a training subset and validation subset. After tuning on the training subset, the student networks are evaluated on the validation subset. Here, we define a new accuracy metric for tracking by selecting the top-N proposals with the highest confidence and calculating their overlaps with the ground-truth boxes for M image pairs from the validation subset:

$$\text{acc} = \sum_{i=1}^{M} \sum_{j=1}^{N} o(g_i, p_{ij}), \tag{2.47}$$

where $p_{ij} (j \in [1, 2, \ldots, N])$ denotes the j-th proposal of the i-th image pair, g_i is the corresponding ground-truth and o is the overlap function. At each step, the policy network outputs N_a actions and the reward is defined as the average reward of generated students:

$$R_t = \frac{1}{N_a} \sum_{i=1}^{N_a} R_{t_i}. \tag{2.48}$$

Training Solution. Given a policy network θ and the predefined MDP, we use a policy gradient network learning to compress Siamese trackers and a policy gradient algorithm to

optimize the network step by step. With the parameters of the policy network denoted as θ, our objective function is the expected reward over all the action sequences $a_{1:T}$:

$$J(\theta) = E_{a_{1:T}\sim P_\theta}(\mathbf{R}). \tag{2.49}$$

We use REINFORCE [203] to calculate the gradient of our policy network. Given the hidden state h_t, the gradient is formulated as:

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta E_{a_{1:T}\sim P_\theta}(\mathbf{R}) \\ &= \sum\nolimits_{t=1}^{T} E_{a_{1:T}\sim P_\theta}[\nabla_\theta \log P_\theta(a_t|a_{1:(t-1)})R_t] \\ &\approx \sum\nolimits_{t=1}^{T}[\nabla_\theta log P_\theta(a_t|h_t)\frac{1}{N_a}\sum\nolimits_{i=1}^{N_a} R_{t_i}], \end{aligned} \tag{2.50}$$

where $P_\theta(a_t|h_t)$ is the probability of actions controlled by the current policy network with hidden state h_t. R_{t_i} is the reward of the current k-th student model at step t. Furthermore, in order to reduce the high variance of estimated gradients, a state-independent baseline b is introduced:

$$b = \frac{1}{N_a \cdot T}\sum\nolimits_{t=1}^{T}\sum\nolimits_{i=1}^{N_a} R_{t_i}. \tag{2.51}$$

It denotes an exponential moving average of previous rewards. Finally, our policy gradient is calculated as:

$$\nabla_\theta J(\theta) \approx \sum\nolimits_{t=1}^{T}[\nabla_\theta log P_\theta(a_t|h_t)(\frac{1}{N_a}\sum\nolimits_{i=1}^{N_a} R_{t_i} - b)]. \tag{2.52}$$

We use the gradient to optimize the policy, obtaining a final policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$ and reduced student network. All the training processes in this section are based on the small dataset selected from the whole dataset, considering the time cost of training all students.

Notice that our selection process is a simple network architecture search (NAS) method, where the search space includes the networks with the same layers but fewer convolution channels. Our method aims to search a student network with relatively good performance. A more advanced NAS method may get a better network structure, and it is one effective technique for practical engineering applications. However, this is not our research focus. Thus, we only use a simple method as a baseline.

Teacher-Students Knowledge Distillation

After the network selection, we obtain a "dim" student network with poor comprehension due to the small model size. To pursue more intensive knowledge distillation and favorable tracking performance, we propose a Teacher-Students Knowledge Distillation (TSsKD) model. It encourages teacher-student knowledge transfer as well as mutual learning between students, which serves as more flexible and appropriate guidance. In Sect. 2.1.7, we elaborate

the teacher-student knowledge transfer (distillation) model. Then, in Sect. 2.1.7, we describe the student-student knowledge sharing strategy.

Teacher-Student Knowledge Transfer

In the teacher-student knowledge transfer model, we introduce a transfer learning approach to capture the knowledge in teacher networks. It contains two components: the Siamese Target Response (STR) learning and distillation loss. The former is used for transferring the middle-level feature maps without background disturbance to the student from the teacher. This provides clean critical middle-level semantic hints to the student. The latter includes two sub-loss: the Teacher Soft (TS) loss and Adaptive Hard (AH) loss, which allows the student to mimic the high-level outputs of the teacher network, such as the logits [77] in the classification model. This loss can be viewed as a variant of KD methods [23, 77], which are used to extract dark knowledge from teacher networks. Our transfer learning approach includes both classification and regression and can be incorporated into other networks by removing the corresponding part.

Siamese Target Response (STR) Learning. In order to lead a tracker to concentrate on the same target as a teacher, we propose a background-suppression Siamese Target Response (STR) learning method in our framework. Based on the assumption that the activation of a hidden neuron can indicate its importance for a specific input, we transfer the semantic interest of a teacher onto a student by forcing it to mimic the teacher's response map. We gather the feature responses of different channels into a response map by a mapping function $F : \mathbb{R}_{C\times H\times W} \rightarrow \mathbb{R}_{H\times W}$, which outputs a 2D response map from the 3D feature maps provided. We formulate this as:

$$F(U) = \sum\nolimits_{i=1}^{C} |U_i|, \tag{2.53}$$

where $U_i \in \mathbb{R}_{H\times W}$ is the ith channel of a spatial feature map, and $|\cdot|$ represents the absolute values of a matrix. In this way, we squeeze the responses of different channels into a single response map.

Siamese trackers have two weight-sharing branches with different inputs: a target patch and a larger search region. Different from the detection [23] requiring multiple responses on objects in the search region, our tracking task only needs one ideal high target response in both search region and template patch, respectively. To learn the target responses of both branches, we combine their learning process. Since we found that the surrounding noise in the search region's response will disturb the response learning of the other branch due to the existence of distractors, we set a weight on the search region's feature maps. The following multi-layer response learning loss is defined:

$$L^{\text{STR}} = L_x^{\text{STR}} + L_z^{\text{STR}}, \tag{2.54}$$

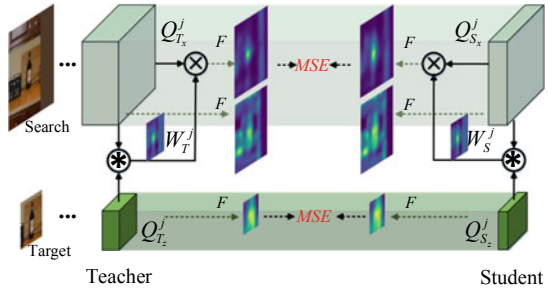


Fig. 2.31 Illustration of our Siamese Target Response (STR) learning. Take one layer as an example. For the target branch, feature maps are directly transformed into 2D activation maps. For the search branch, weights (W_T^j and W_S^j) are calculated by conducting a cross-correlation operation on the two branches' feature maps and then multiplying the result by the search feature map

$$L_x^{\text{STR}} = \sum\nolimits_{j\in\tau} \|F(W_S^j Q_{S_x}^j) - F(W_T^j Q_{T_x}^j)\|_2, \tag{2.55}$$

$$L_z^{\text{STR}} = \sum\nolimits_{j\in\tau} \|F(Q_{S_z}^j) - F(Q_{T_z}^j)\|_2, \tag{2.56}$$

where τ is the set of layers' indices conduct. $Q_{T_x}^j$ and $Q_{T_z}^j$ denote the teacher's feature map of layer j in the search and target branch, respectively. $W_T^j = Q_{T_x}^j \star Q_{T_z}^j$ is the weight on the teacher's jth feature map. The students' variables, such as W_S^j, are defined in the same way.

By introducing this weight, which rearranges the importance of different areas in the search region according to their similarities with the target, the response activation is concentrated on the target. This keeps the response maps from the two branches consistent and enhances the response learning. An example of our multi-layer Siamese target response learning is shown in Fig. 2.31. The comparison of response maps with and without weights shows that the surrounding noise is suppressed effectively. It is worth mentioning that our STR method is the first attempt to do feature learning on Siamese networks.

Distillation Loss. Our distillation loss contains two critical components: the teacher soft loss, which is used to imitate the teacher, and the adaptive hard loss which is designed to integrate the ground-truth.

(1) Teacher Soft (TS) Loss. We set C_s and B_s as the student's classification and bounding box regression outputs, respectively. C_t and B_t represent the teachers' variables. In order to incorporate the dark knowledge [77] that regularizes students by placing emphasis on the relationships learned by the teacher network across all the outputs, we need to 'soften' the output of classification. We set $P_t = \text{softmax}(C_t/\text{temp})$, where temp is a temperature parameter used to obtain a soft distribution [77]. Similarly, $P_s = \text{softmax}(C_s/\text{temp})$. Then, we formulate the TS loss as follows:

$$L^{\text{TS}} = L_{\text{cls}}^{\text{TS}}(P_s, P_t) + L_{\text{reg}}^{\text{TS}}(B_s, B_t), \tag{2.57}$$

where $L_{\text{cls}}^{\text{TS}} = \text{KL}(P_s, P_t)$ is a Kullback Leibler (KL) divergence loss on the soft outputs of the teacher and student. $L_{\text{reg}}^{\text{TS}}$ is the original regression loss of the tracking network.

(2) Adaptive Hard (AH) Loss. To make full use of the ground-truth G, we combine the outputs of the teacher network with the original hard loss of the student network. For the regression loss, we employ a *teacher bounded regression loss* [23].

$$L_{\text{reg}}^{\text{AH}}(B_s, B_t, G_{\text{reg}}) = \begin{cases} L_r(B_s, G_{\text{reg}}), & \text{if } gap < m, \\ 0, & \text{otherwise.} \end{cases} \tag{2.58}$$

where $gap = L_r(B_t, G_{\text{reg}}) - L_r(B_s, G_{\text{reg}})$ is the gap between the student's and the teacher's loss (L_r is the regression loss of the tracking network) with the ground-truth. m is a margin. This loss aims to keep the student's regression vector close to the ground-truth when its quality is worse than the teacher. However, once it is close to or outperforms the teacher network, we remove the loss for the student to avoid over-fitting. For the classification loss, there is no unbounded issue in the discrete classification task [23]. Thus, we directly use the student's original classification loss $L_{\text{cls}}^{\text{AH}}$. Finally, our AH loss is defined as follows:

$$L^{\text{AH}} = L_{\text{cls}}^{\text{AH}}(C_s, G_{\text{cls}}) + L_{\text{reg}}^{\text{AH}}(B_s, B_t, G_{\text{reg}}). \tag{2.59}$$

Then, incorporated with the TS loss, our distillation loss is formulated as,

$$L^{\text{D}} = \eta L^{\text{TS}} + \lambda L^{\text{AH}}, \tag{2.60}$$

where η and λ are balance parameters. It is worth mentioning that our L^{D} is not a simple extension of the distillation loss for object detection (denoted as L_{O}^{D}) in [23]. To closely analyze the differences, we reformulate it as the summary of the soft loss and hard loss, *i.e.* $L_{\text{O}}^{\text{D}} = L_{\text{O}}^{\text{S}} + L_{\text{O}}^{\text{H}}$, where

$$L_{\text{O}}^{\text{S}} = (1-\mu) L_{\text{cls}}^{\text{TS}}(P_s, P_t), \tag{2.61}$$

$$L_{\text{O}}^{\text{H}} = \mu L_{\text{cls}}^{\text{AH}}(C_s, G_{\text{cls}}) + L_{\text{reg}}^{\text{AH}}(B_s, B_t, G_{\text{reg}}) + \nu L_r(B_s, G_{\text{reg}}). \tag{2.62}$$

To focus on the major components of these losses, we ignore the impact of the balance parameters (μ, ν). Comparing Eq. (2.57) with (2.62), the previous soft loss L_{O}^{S} lacks the teacher regression loss $L_{\text{reg}}^{\text{TS}}$. In the detection task, the teacher may provide wrong guidance [23]. However, in visual tracking which can be regarded as a one-shot single target detection task, the teacher's regression output is accurate enough. Furthermore, lacking the teacher regression loss indicates that the L_{O}^{S} loss can not push the student network to imitate the regression branch of the teacher as much as possible or fully mine the underlying knowledge inside the teacher. Thus, we add the regression loss $L_{\text{reg}}^{\text{TS}}$ into the proposed distillation loss. Comparing Eq. (2.59) with (2.62), the previous hard loss L_{O}^{H} contains an additional regression loss $L_r^s = L_r(B_s, G_{\text{reg}})$, which is a little redundant for two reasons. Firstly, when the student performs worse than the teacher ($gap < m$), L_r^s provides the same information as $L_{\text{reg}}^{\text{AH}}$. If the student performs similar to or better than the teacher network, which means

that the student has learnt enough knowledge from the ground-truth, L_r^s will not offer additional information for training. In contrast, the strong supervision from L_r^s easily leads to over-fitting on some samples. Therefore, we remove L_r^s in our distillation loss.

Overall Loss. By combining the above STR loss and distillation loss, the overall loss for transferring knowledge from a teacher to a student is defined as follows:

$$L^{\mathrm{KT}} = \omega L^{\mathrm{STR}} + \eta L^{\mathrm{TS}} + \lambda L^{\mathrm{AH}}. \tag{2.63}$$

Student-Student Knowledge Sharing

Based on our teacher-student distillation model, we propose a student-student knowledge sharing mechanism to further narrow the gap between the teacher and the "dim" student. As an "intelligent" student with a larger model size usually learns and performs better (due to its better comprehension), sharing its knowledge is likely able to inspire the "dim" one to develop a more in-depth understanding. On the other side, the "dim" student can do better in some cases and provide some useful knowledge too. To capture the useful knowledge and reduce the "bad" knowledge from the "dim" student, we propose a conditional suppressed weighting for our knowledge sharing. It is worth mentioning that our experimental results show the proposed knowledge sharing is effective for a variety of tracking challenges, especially in case of *deformation* and *background clutter*. More details are presented in Sect. 2.1.7.

We take two students as an example and denote them as a "dim" student s1 and an "intelligent" student s2. For a true distribution $p(x)$ and the predicted distribution $q(x)$, the KL divergence on N samples is defined as:

$$\begin{aligned} D_{KL}(p||q) &= \sum_{i=1}^{N} p(x_i) log(\frac{p(x_i)}{q(x_i)}) \\ &= -H(p(x)) + (-\sum_{i=1}^{N} p(x_i) log(q(x_i))) \\ &= -H(p(x)) + L_{CE}(p, q), \end{aligned} \tag{2.64}$$

where H is the entropy and L_{CE} is the cross-entropy. We can see that, unlike the traditional cross-entropy loss, the KL divergence loss contains an entropy item of the label (true distribution). The tracking classification output can be regarded as a probability distribution function (PDF). If $p(x)$ is a hard label and thus a fixed constant, the KL divergence and cross-entropy loss are equal during training. However for the knowledge sharing, $p(x)$ is a differentiable output of the other student. Compared to the cross-entropy loss, the KL divergence (also known as relative entropy) offers a more accurate supervision signal for the training and can better measure the similarity of two PDFs. Therefore, we use KL divergence loss as the classification loss for knowledge sharing.

For a proposal d_i in a Siamese tracker, assume that the predicted probabilities of being target by s1 and s2 are $p_1(d_i)$ and $p_2(d_i)$, respectively. The predicted bounding-box regression values are $r_1(d_i)$ and $r_2(d_i)$. To improve the learning effect of s1, we obtain the knowledge shared from s2 by using its prediction as prior knowledge. The KL divergence is defined as:

$$L_{\text{cls}}^{\text{KS}}(\text{s1}||\text{s2}) = \sum_{i=1}^{N} (p_1(d_i) log \frac{p_1(d_i)}{p_2(d_i)} + (1 - p_1(d_i)) log \frac{1 - p_1(d_i)}{1 - p_2(d_i)}). \tag{2.65}$$

For regression, we use the smooth L_1 loss:

$$L_{\text{reg}}^{\text{KS}}(\text{s1}||\text{s2}) = \sum_{i=1}^{N} L_1(r_1(d_i) - r_2(d_i)). \tag{2.66}$$

The knowledge sharing loss for $s1$ can be defined as:

$$L^{\text{KS}}(\text{s1}||\text{s2}) = L_{\text{cls}}^{\text{KS}}(\text{s1}||\text{s2}) + L_{\text{reg}}^{\text{KS}}(s1||s2). \tag{2.67}$$

However, there exists some representation inaccuracy during the sharing process, especially at the first several epochs. To filter out the reliable shared knowledge, we introduce a conditional suppressed weighting function σ for L^{KS}:

$$\sigma(\text{s1}) = \begin{cases} f(e) & \text{if } L^{\text{GT}}(\text{s2}) - L^{\text{GT}}(t) < h, \\ 0 & \text{otherwise.} \end{cases} \tag{2.68}$$

Here, $L^{\text{GT}}(\text{s2})$, $L^{\text{GT}}(t)$ are the losses for s2 and the teacher, with ground-truth. h is their gap constraint. $f(e)$ is a function that decreases geometrically with current epoch e. Our knowledge transfer and sharing run simultaneously, which prevents the accumulation of the representation errors. The overall knowledge distillation loss with conditional knowledge sharing is defined as:

$$L_{\text{s1}}^{\text{KD}} = L_{\text{s1}}^{\text{KT}} + \sigma(\text{s1}) L^{\text{KS}}(\text{s1}||\text{s2}). \tag{2.69}$$

In this way, we ensure that the shared knowledge offers accurate guidance to s1 and enhances the training. For s2, the loss is similar:

$$L_{\text{s2}}^{\text{KD}} = L_{\text{s2}}^{\text{KT}} + \beta \cdot \sigma(\text{s2}) L^{\text{KS}}(\text{s2}||\text{s1}), \tag{2.70}$$

where $\sigma(\text{s2})$ is defined in the same way as $\sigma(\text{s1})$ and β is a discount factor on account of the difference in reliability of the two students. Considering the "dim" student's worse performance, we set $\beta \in (0, 1)$.

Finally, to train two students simultaneously, the loss for our TSsKD is:

$$L^{\text{KD}} = L_{\text{s1}}^{\text{KD}} + L_{\text{s2}}^{\text{KD}}. \tag{2.71}$$

Experiments

To demonstrate the effectiveness of the proposed method, we conduct experiments on SiamFC [15]), SiamRPN [110] (VOT version as in [109]) and SiamRPN++ [109]. For the simple Siamese trackers (SiamRPN [110] and SiamFC [15]), since there is no smaller classic handcrafted structure, we first search and then train a proper "dim" student via our framework. We evaluate the distilled trackers on several benchmarks and conduct an ablation study (from Sects. 2.1.7–2.1.7). Furthermore, to validate our TSsKD on well-designed handcrafted structures, we distilled SiamRPN++ trackers with different backbones (Sect. 2.1.7). Note that all experiments, unless otherwise stated, are conducted using two students. All the experiments are implemented using PyTorch with an Intel i7 CPU and four Nvidia GTX 1080ti GPUs.

Implementation Details

Reinforcement Learning Setting. In the "dim" student selection experiment, an LSTM is employed as the policy network. A small representative dataset (about 10,000 image pairs) is created by selecting images uniformly from several classes in the whole dataset to train the corresponding tracker. The policy network is updated over 50 steps. In each step, three compressed networks are generated and trained from scratch for 10 epochs. We observe heuristically that this is sufficient to compare performance. Both SiamRPN and SiamFC use the same settings.

Training Datasets. For SiamRPN, as with the teacher, we pre-process four datasets: ImageNet VID [164], YouTube-BoundingBoxes [159], COCO [123] and ImageNet Detection [164], to generate about two million image pairs with 127×127 pixel target patches and 271×271 pixel search regions. Unlike SiamRPN, our student does not need a backbone pre-trained on ImageNet and can learn good discriminative features via STR. To make the model robust to gray videos, 25% of the pairs are converted into grayscale during training. Moreover, a translation within 12 pixels and a resize operation varying from 0.85 to 1.15 are performed on each training sample to increase the diversity. Our SiamFC is trained on ImageNet VID [164] with 127×127 pixels and 255×255 pixels for the two inputs, respectively, which is consistent with SiamFC [15].

Optimization. During the teacher-students knowledge distillation, "intelligent" students are generated by halving the convolutional channels of the teachers (SiamRPN and SiamFC). SiamRPN's student networks are warmed up by training with the ground-truth for 10 epochs, and then trained for 50 epochs with the learning rate exponentially decreasing from 10^{-2} to 10^{-4}. As with the teacher, SiamFC's student networks are trained for 30 epochs with a learning rate of 10^{-2}. All the losses used in the experiments are reported in Table 2.9. The other hyperparameters are set to: $m = 0.005$, $\omega = 100$, $\eta = 1$, $\lambda = 0.1$, $temp = 1$, $h = 0.005$ and $\beta = 0.5$. We set these weights according to the empirical settings and the scale of different loss components. In practice, according to our experiments, this does not have much influence on the final results (Fig. 2.32).

Table 2.9 Losses used in the knowledge transfer stage. MSE, L_1 and KL represent Mean-Square-Error loss, smooth l_1 loss and Kullback Leibler divergence loss, respectively

	L^{AH}	L^{TS}	L^{STR}	L^{KS}
SiamFC	Logistic	KL	MSE	KL
SiamRPN	Cross-entropy+bounded	KL+L_1	MSE	KL+L_1

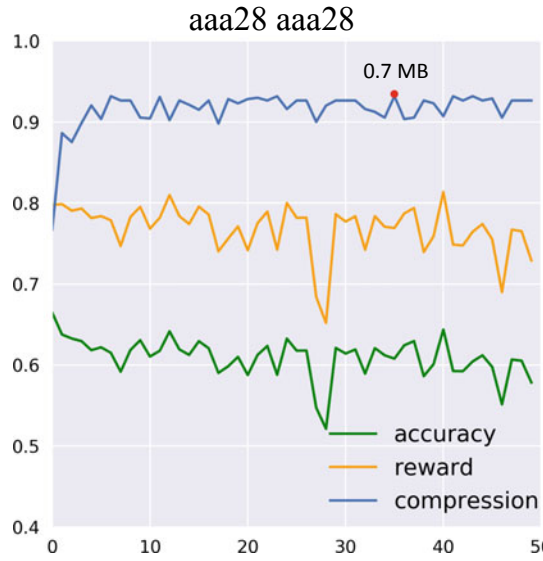


Fig. 2.32 "Dim" student selection on **a** SiamRPN and **b** SiamFC. Reward, accuracy, compression (relative compression rate C in Eq. 2.46) versus iteration

Evaluations of "Dim" Student Selection

Training Details. As shown in Fig. ??, the complicated architecture of SiamRPN caused several inappropriate SiamRPN-like networks to be generated in the top 30 iterations, leading to unstable accuracies and rewards. After five iterations, the policy network gradually converges and finally achieves a high compression rate. On the other side, the policy network of SiamFC converges quickly after several iterations due to its simple architecture (See Fig. ??). The compression results show that our method is able to generate an optimal architecture regardless of the teacher's complexity. Finally, two reduced models of size 19.7 MB and 0.7 MB for SiamRPN (361.8 MB) and SiamFC (9.4 MB) are generated. The detailed structures of the reduced models, DSTrpn and DSTfc, are shown in Table 2.10.

Loss Comparison. We also compare the losses of different student networks. As shown in Fig. 2.33, the "intelligent" students (denoted as Student1) have a lower loss than the "dim" ones (denoted as Student2) throughout the whole training and validation process, and maintain a better understanding of the training dataset. They provide additional reliable knowledge to the "dim" students which further promotes more intensive knowledge distillation and better tracking performance.

Table 2.10 Detailed convolutional structures of DSTfc and DSTrpn. The numbers denote the input and output channel numbers of the corresponding convolutional layers

	conv1	conv2	conv3	conv4	conv5	rpn_cls1/2	rpn_reg1/2
DSTfc	(3,38)	(38,64)	(64,96)	(96,96)	(96,64)	/	/
DSTrpn	(3,192)	(192,128)	(128,192)	(192,192)	(192,128)	(128,128/1280)	(128,128/2560)

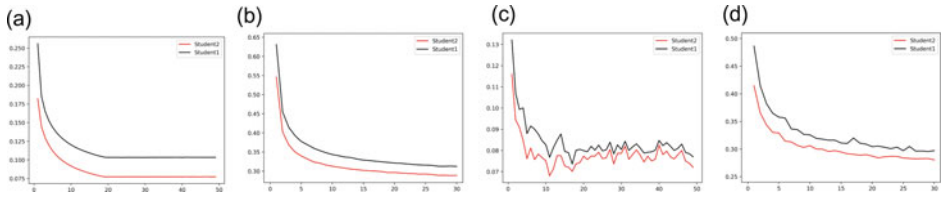


Fig. 2.33 Losses comparison, including training loss of the **a** SiamRPN and **b** SiamFC students, and validation loss of the **c** SiamRPN and **d** SiamFC students

Benchmark Results

Results on OTB-100. On the OTB-100 benchmark [207], a conventional method for evaluating trackers is one-pass evaluation (OPE). However, some trackers may be sensitive to the initialization. To measure the performance with different initializations, we use spatial robustness evaluation (SRE) and temporal robustness evaluation (TRE). SRE uses different bounding boxes in the first frame and TRE starts at different frames for initialization. We compare our DSTrpn (SiamRPN as teacher) and DSTfc (SiamFC as teacher) trackers with various recent fast trackers (more than 50 FPS), including the teacher networks SiamRPN [110] and SiamFC [15], Siam-tri [43], TRACA [29], HP [44], Cfnet2 [187], and fDSST [37]. The evaluation metrics include both precision and success plots in OPE [207], where ranks are sorted using precision scores with center error less than 20 pixels and Area-Under-the-Curve (AUC), respectively. In Fig. 2.34, our DSTrpn outperforms all the other trackers in terms of precision and success plots. As for speed, DSTrpn runs at an extremely high speed of 265 FPS, which is nearly 3× faster than SiamRPN (90 FPS) and obtains the same (even slightly better) precision and AUC scores. DSTfc runs more than 2× faster than SiamFC with comparable performance.

Results on DTB. We benchmark our method on the Drone Tracking Benchmark (DTB) [116], which includes 70 videos captured by drone cameras. We compare SiamRPN, recent Siamese works such as HP [44], and the trackers evaluated in DTB, including DAT [156], HOGLR [198], SODLT [197], DSST [35], MDNet [149], MEEM [217], and SRDCF [39]. The evaluation metrics include distance Precision (P) at a threshold of 20 pixels, Overlap Precision (OP) at an overlap threshold of 0.5, and AUC.

As shown in Table 2.11, DSTrpn performs best in terms of DP and OP. For AUC, DSTrpn ranks first (0.5557) and significantly outperforms SiamFC (0.4797). Compared

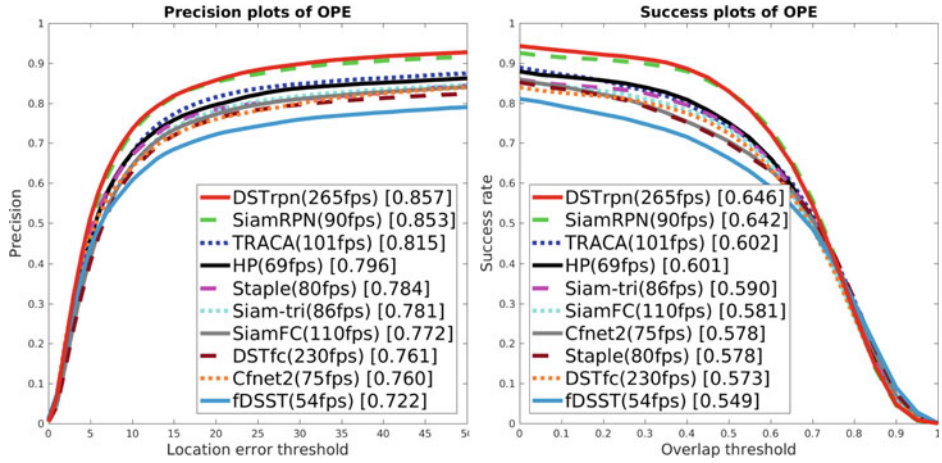


Fig. 2.34 Precision and success plots with AUC for OPE on the OTB-100 benchmark [207]

Table 2.11 Evaluation on DTB [116] by Precision (P), Overlap Precision (OP) and Area-Under-the-Curve (AUC). The first, second and third best scores are highlighted in color

	DSTrpn	DSTfc	SiamRPN [110]	SiamFC [15]	HP [44]	DAT [156]	HOGLR [198]	SODLT [197]	DSST [35]	MDNet [149]	MEEM [217]	SRDCF [39]
P	0.7965	0.7486	0.7602	0.7226	0.6959	0.4237	0.4638	0.5488	0.4037	0.6916	0.5828	0.4969
OP	0.6927	0.5741	0.6827	0.5681	0.5775	0.2650	0.3057	0.4038	0.2706	0.5328	0.3357	0.3723
AUC	0.5557	0.4909	0.5502	0.4797	0.4721	0.2652	0.3084	0.3640	0.2644	0.4559	0.3649	0.3390

with SiamRPN, our DSTrpn surpasses it in terms of both AUC and OP, while even achieving a 3.6% improvement on DP. This is particularly impressive considering that our model size is just 1/18 of the teacher SiamRPN. DSTfc outperforms SiamFC in terms of all three criterias too. On DTB [116], the metrics include P at a threshold of 20 pixels and OP at an overlap threshold of 0.5.

Results on VOT2019, LaSOT and TrackingNet. We also conduct extensive experiments on challenging and large-scale datasets, including VOT2019 [102], LaSOT [51] and TrackingNet [147], to evaluate the generalization of our method. On VOT2019, the trackers are ranked by EAO (Expected Average Overlap) while on LaSOT and TrackingNet, OP and DP are used, similar to DTB. We compare DaSiamRPN [234], ECO [34], MDNet [149], and our baselines: SiamRPN [110] and SiamFC [15].

As shown in Table 2.12 and Fig. 2.35, the model size of our DSTrpn (or DSTfc) is further smaller than its teacher model SiamRPN (or SiamFC), while the AUC scores on two large-scale datasets are very close (about 0.02 on DSTrpn) to the teacher. This strongly demonstrates the robustness of the two distilled trackers on long and various videos. Based on SiamRPN [110], DaSiamRPN introduces a distractor-aware model updating strategy, as well as a global detection module. Note that, our DSTrpn achieves better performance than

Table 2.12 Results comparison on VOT2019 [102] in terms of EAO, A (Accuracy) and R (Robustness), LaSOT [51] and TrackingNet [147] in terms of AUC, *P* (Precision)

	VOT 2019			TrackingNet		
	EAO	A	R	AUC	*P*	FPS
ECO [34]	/	/	/	0.554	0.492	8
MDNet [149]	/	/	/	0.606	0.565	1
DaSiamRPN [234]	/	/	/	0.638	0.591	160
SiamRPN [110]	0.272	0.582	0.527	0.675	0.622	90
SiamFC [15]	0.183	0.511	0.923	0.573	0.52	110
DSTrpn	0.247	0.552	0.637	0.649	0.589	265
DSTfc	0.182	0.504	0.923	0.562	0.512	230

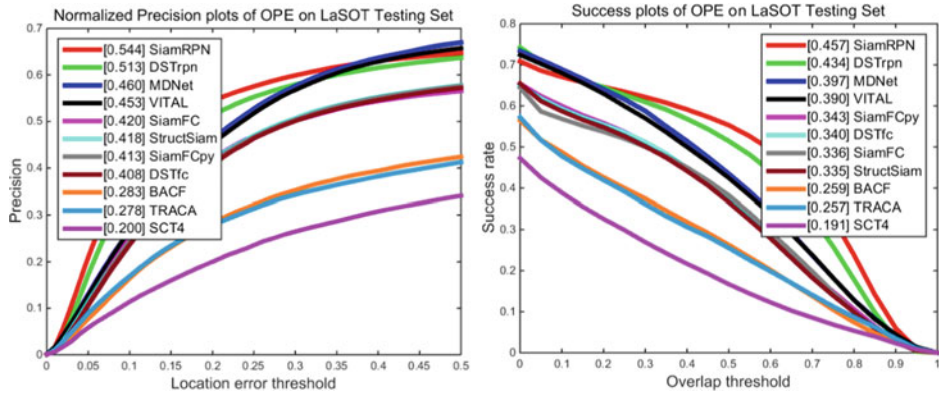


Fig. 2.35 Evaluation results of trackers on the LaSOT benchmark [51]

DaSiamRPN in both two datasets while maintaining smaller model size and not using any complex strategy or additional module. Via the proposed KD training, our trackers achieve comparable performance with few accuracy losses and much higher speeds on long and challenging videos.

Results on UAVDT and FaceTracking. Unmanned aerial vehicle tracking and face tracking are two practical applications of visual tracking. They are good scenes to evaluate tracking models in practical situations. We evaluate our methods on UAVDT [116] and FaceTracking [157], the results are shown in Table 2.13. AUC and distance precision are used on both benchmarks like TrackingNet [147]. We compare ECO [34], MDNet [149], LGT [157], and our baselines: SiamRPN [110] and SiamFC [15]. On both benchmarks, our distilled trackers obtain comparable or even slightly better performance compared with our baselines. These results further demonstrate the effectiveness and generalization of the proposed knowledge distillation method.

Table 2.13 Comparison on UAVDT [116] and FaceTracking [157] in terms of AUC and *P* (Precision)

	UAVDT		FaceTracking		
	AUC	*P*	AUC	*P*	FPS
ECO [34]	0.488	0.702	0.538	0.834	8
MDNet [149]	0.492	0.725	0.499	0.833	1
LGT [157]	/	/	0.559	0.833	4
SiamRPN [110]	0.733	0.836	0.453	0.809	90
SiamFC [15]	0.526	0.681	0.425	0.694	110
DSTrpn	0.717	0.817	0.452	0.813	265
DSTfc	0.518	0.686	0.430	0.698	230

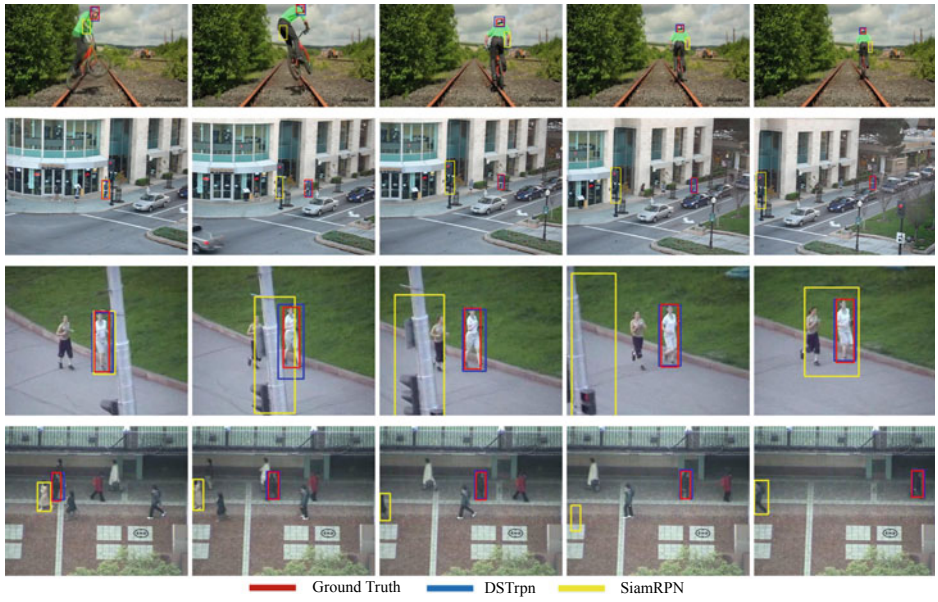


Fig. 2.36 Sample results of SiamRPN and our DSTrpn on OTB-100 [207] sequences (Biker, Human4, Jogging-2 and Subway) and DTB [116] sequences (ChasingDrones, Horse1, RcCar7 and Soccer2). On these sequences, our DSTrpn outperforms SiamRPN while running at a much faster speed

Qualitative Evaluation

We compare our DSTrpn method with SiamRPN [110] on several challenging sequences from OTB-100 [207] and DTB [116] in Fig. 2.36. The previous knowledge distillation (KD) work [162] has revealed that the student model with KD can outperform the teacher in some cases or on some data distributions. Similarly, our student model DSTrpn also achieves better

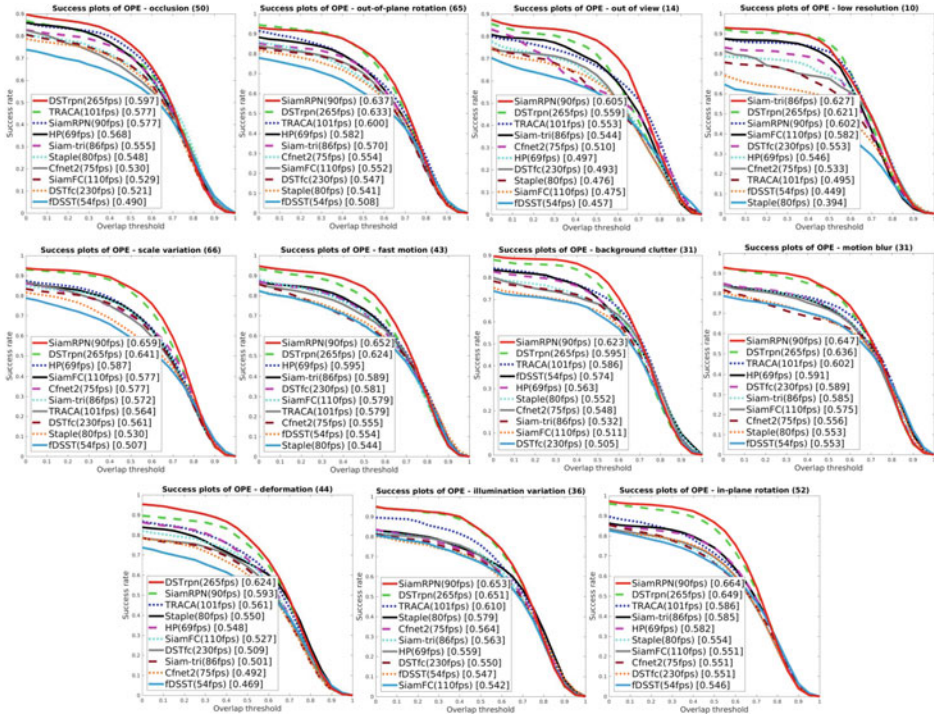


Fig. 2.37 Overlap success plots of OPE with AUC for 11 tracking challenges on OTB-100 [207] including: *Illumination Variation (IV), Scale Variation (SV), Occlusion (OCC), Deformation (DEF), Motion Blur (MB), Fast Motion (FM), In-Plane Rotation (IPR), Out-of-Plane Rotation (OPR), Out-of-View (OV), Background Clutter (BC)* and *Low Resolution (LR)*. Our method achieves the best performance

performance on several challenging sequences compared with its teacher SiamRPN. For example, on the Biker sequence (first row in Fig. 2.36), SiamRPN fails to track objects well, whereas our DSTrpn algorithm performs accurately in terms of both precision and overlap. The SiamRPN method gradually loses track of the target due to significant Deformation (DEF) and Fast Motion (FM) in Biker sequence. When Illumination Variation (IV) and Scale Variation (SV) occur, our tracker is more stable than SiamRPN as shown on the Human4 sequence. On Jogging-2 and Subway, our tracker overcomes Occlution (OCC), Out-of-Plane Rotation (OPR) and Background Clutter (BC), maintaining high tracking accuracy. On all four sequences, our DSTrpn outperforms the teacher while running much more faster. In Fig. 2.37, the overlap scores of our DSTrpn and other trackers on 11 tracking challenges are shown. Our method achieves the best results on all challenges, while running much faster than other trackers. This reveals the robustness and effectiveness of our method for challenging scenes. Further, with the shared knowledge from the other student, our students

can even outperform their teachers in certain cases and obtain better overall performance on certain benchmarks, including OTB-100 and DTB.

Ablation Study

Knowledge Transfer Components. The teacher-student knowledge transfer consists of three components: (i) AH loss, (ii) TS loss, and (iii) STR loss. We conduct an extensive ablation study by implementing a number of variants using different combinations, including (1) GT: simply using hard labels, (2) TS, (3) GT+TS, (4) AH+TS, (5) TS+STR, (6) GT+TS+STR, and (7) AH+TS+STR (the full knowledge transfer method). Table 2.14 shows our results on SiamFC and SiamRPN. For SiamRPN, we can see that the GT without any proposed loss degrades dramatically compared with the teacher, due to the lack of a pre-trained backbone. When using the TS loss to train student, we observe a significant improvement in terms of precision (15.8%) and AUC (15.7%). However, directly combining GT and TS (GT+TS) could be suboptimal due to over-fitting. By replacing GT with AH, AH+TS further boosts the performance for both metrics. Finally, by adding the STR loss, the model (AH+TS+STR) is able to close the gap between the teacher and student, outperforming other variants. In addition to the results shown in Table 2.14, we do two more studies for the final model (AH+TS+STR) to further demonstrate the impact of the newly proposed weighting strategy in STR and distillation loss (AH+TS). In the first study, we remove the weighting strategy and achieved the worse Precision (0.814) and lower AUC

Table 2.14 Results for different combinations of GT, TS, AH and STR in terms of precision and AUC on OTB-100 [207]

		GT	AH	TS	STR	Precision	AUC
SiamRPN	Student1	✓				0.638	0.429
				✓		0.796	0.586
		✓		✓		0.795	0.579
			✓	✓		0.800	0.591
				✓	✓	0.811	0.608
		✓		✓	✓	0.812	0.606
			✓	✓	✓	0.825	0.624
	Teacher	/	/	/	/	0.853	0.643
SiamFC	Student1	✓				0.707	0.523
				✓		0.711	0.535
		✓		✓		0.710	0.531
				✓	✓	0.742	0.548
		✓		✓	✓	0.741	0.557
	Teacher	/	/	/	/	0.772	0.581

Table 2.15 Ablation experiments of different learning mechanisms (NOKD, KD, TSsKD) in terms of AUC on OTB-100 [207]

		NOKD	TSKD	TSsKD	Size	FPS
SiamRPN	Student1	0.429	0.624	0.646	19.7M	265
	Student2	0.630	0.641	0.644	90.6M	160
	Teacher	0.642	/	/	361.8M	90
SiamFC	Student1	0.523	0.557	0.573	0.7M	230
	Student2	0.566	0.576	0.579	2.4M	165
	Teacher	0.581	/	/	9.4M	110

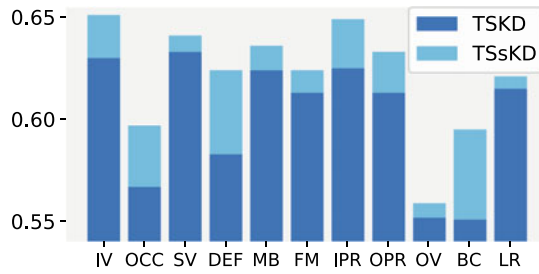


Fig. 2.38 The AUC scores of different learning mechanisms on 11 challenging attributes of OTB-100 [207]

(0.609). In the second one, we replace our distillation loss with the bounded loss [23] and got a reduced performance of 0.817 and 0.612 in terms of the Precision and AUC, respectively. SiamFC only employs the classification loss, so GT is equal to AH and we use GT here. Results show that the gaps are narrower than SiamRPN but improvements are still obvious. These results clearly demonstrate the effectiveness of each component.

Different Learning Mechanisms. To evaluate our TSsKD model, we also conduct an ablation study on different learning mechanisms : (i) NOKD: trained with hard labels, (ii) TSKD: our tracking-specific teacher-student knowledge distillation (transfer) and (iii) TSsKD. "Student1" and "Student2" represent the "dim" and "intelligent" student, respectively. Students are trained following different paradigms and results can be seen in Table 2.15. With KD, all students are improved. Moreover, with the knowledge sharing in our TSsKD, the "dim" SiamRPN student gets a performance improvement of 2.2% in terms of AUC. The "dim" SiamFC student gets a 1.6% improvement. More specifically, the shared knowledge from the "intelligent" student is effective and provides promising improvement on all challenging tracking cases. As shown in Fig. 2.38, TSsKD outperforms TSKD on all 11 challenging attributes of OTB-100 [207]. In terms of DEF (*Deformation*), and BC (*Background Clutter*), knowledge sharing obtains a significant improvement of more than 4%. The TSsKD model obviously enhances the robustness of "dim" student. Besides, the "intelligent" SiamRPN and SiamFC students get slight improvements (0.3%) as well. Fusing the knowledge from

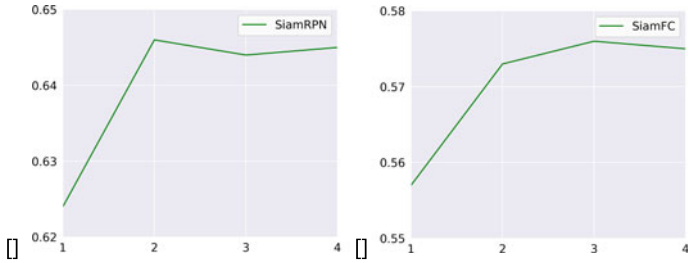


Fig. 2.39 Performance of **a** DSTrpn and **b** DSTfc on OTB-100 [207] with different student numbers in terms of AUC

the teacher, ground-truth and "intelligent" student, the "dim" SiamRPN student obtains the best performance.

Experiments on More Students. Our TSsKD model can be extended to more students. Given n students s1, s2, ..., sn, the objective function for si is as follows:

$$L_{\text{si}}^{\text{KD}} = L_{\text{si}}^{\text{KT}} + \frac{1}{n}\sum_{j=1}^{n} \beta_{ij}\sigma(\text{s1})L^{\text{KS}}(\text{si}||\text{sj}). \tag{2.72}$$

Here β_{ij} is the discount factor between si and sj considering their different reliabilities. For example, in our case of two students, $\beta_{12} = 1$ and $\beta_{21} = 0.5$. We conduct an experiment using different numbers of student and report the results in Fig. 2.39. Students are generated by reducing the number of convolutional channels to a scale (0.4, 0.45, 0.5, 0.55). In our case, since our "dim" students already achieve a performance similar to the teacher's with only one "intelligent" student, more students do not bring significant improvements.

Sensibility Evaluation. To further evaluate the robustness of the proposed KD methods, we select several models generated during the DRL process to evaluate our method's sensibility to model selection. In this experiment, we select several models and Table 2.16 reports their tracking performance on OTB [207]. We can see that their performances are very comparable to the teachers', verifying our method's robustness to the architectures of the searched networks.

Grayscale. To evaluate the impact of grayscales on the training of student networks, we conduct an experiment to train them with different scales of gray images and test them on OTB [207] dataset. As shown in Table 2.17, 15% to 25% is a suitable setting for performance, and too many or too few gray images will cause a performance drop in the training.

CPU Speed. To provide a more practical speed comparison of different networks, we also test them on an Intel(R) Xeon(R) 2.20GHz CPU. In this computation-constrained running environment, our DSTrpn and DSTfc can run at a speed of 20 and 30 FPS, respectively, which is close to real-time speed. SiamRPN and SiamFC only achieve 8 and 12 FPS. These results clearly demonstrate our merits.

Table 2.16 AUC score of searched students in different DRL iterations, tested on the OTB [207]

DRL Iteration times	35	40	45
SiamRPN	0.645	0.646 (**DSTrpn**)	0.642
SiamFC	0.573 (**DSTfc**)	0.580	0.576

Table 2.17 AUC score of students trained with different scales of gray images, tested on the OTB [207]

Grayscale	0%	5%	15%	25%	35%
DSTrpn	0.621	0.631	0.643	0.646	0.640
DSTfc	0.565	0.570	0.574	0.573	0.569

Extended Experiments on SiamRPN++

The backbone network of SiamRPN++ is pre-trained on ImageNet for image labeling. Two sibling convolutional layers are attached to the stride-reduced ResNet-50 to perform proposal classification and bounding box regression with five anchors. Three randomly initialized 1x1 convolutional layers are attached to conv3, conv4, conv5 for reducing the feature dimension to 256. The whole network is trained with stochastic gradient descent (SGD) on the four datasets as with SiamRPN. A warmup learning rate of 0.001 is used for the first five epochs to train the RPN branches. For the last 15 epochs, the whole network is end-to-end trained with the learning rate exponentially decayed from 0.005 to 0.0005. A weight decay of 0.0005 and momentum of 0.9 are used. The training loss is the sum of the classification loss and the standard smooth L1 loss for regression.

The backbone of SiamRPN++ is deep and it is difficult to obtain a reliable performance without pre-training it on ImageNet, so it is not suitable to use the same reinforcement learning method as SiamRPN. On the other side, there are many smaller classic networks with guaranteed performance (such as ResNet18, ResNet34 [72]), which can be selected as our student models. In this part, we first train the original SiamRPN++ tracker (with ResNet50 as the backbone). Then, two SiamRPN++ models with a pre-trained ResNet34 and ResNet18 backbone are trained simultaneously as the students in our TSsKD. Results in Table 2.18 show that our TSsKD can further improve the SOTA SiamRPN++ trackers with small backbones (ResNet34 or ResNet18). All training settings are kept the same as in [109]. We can see that, with the proposed knowledge distillation method, both trackers are improved significantly on all benchmarks.

Table 2.18 Results of different trackers trained with(w)/without(w/o) TSsKD. "r34" and "r18" denote trackers using ResNet34 and ResNet18 as their backbone, respectively

	VOT 2019			LaSOT		TrackingNet		
	EAO	A	R	AUC	P_{norm}	AUC	P	FPS
SiamRPN++	0.287	0.596	0.472	0.496	0.568	0.733	0.694	35
SiamRPN++r34 (w/o)	0.270	0.585	0.515	0.464	0.548	0.690	0.651	50
SiamRPN++r18 (w/o)	0.255	0.586	0.552	0.443	0.520	0.672	0.617	75
SiamRPN++r34 (w)	0.288	0.604	0.484	0.472	0.562	0.699	0.657	50
SiamRPN++r18 (w)	0.271	0.588	0.517	0.465	0.544	0.676	0.623	75

Conclusion

This paper proposed a new Distilled Siamese Tracker (DST) framework to learn small, fast and accurate trackers from larger Siamese trackers. This framework is built upon a teacher-students knowledge distillation model that includes two types of knowledge transfer: (1) knowledge transfer from teacher to students by a tracking-specific distillation strategy; (2) mutual learning between students in a knowledge sharing manner. The theoretical analysis and extensive empirical evaluations on two Siamese trackers have clearly demonstrated the generality and effectiveness of the proposed DST. Specifically, for the SOTA SiamRPN, the distilled tracker achieved a high compression rate, ran at an extremely high speed, and obtained a similar performance as the teacher. Thus, we believe such a distillation method could be used for improving many SOTA deep trackers for practical tracking tasks.

2.2 Multi-object Tracking

2.2.1 Introduction

In this section, we introduce one of the most essential setting of the video content understanding tasks, Multi-object tracking (MOT), which aims at extracting the spatial and temporal trajectories of multiple moving objects from the video. Specifically, MOT obtains the spatial locations of the interested targets in each single frame technically through object detection, segmentation and tracking. Based on the temporal correspondence of individual targets among consecutive frames, MOT establishes moving information connections and achieves accurate trajectories tracking along spatial and temporal dimension. An illustration of MOT is shown in Fig. 2.40.

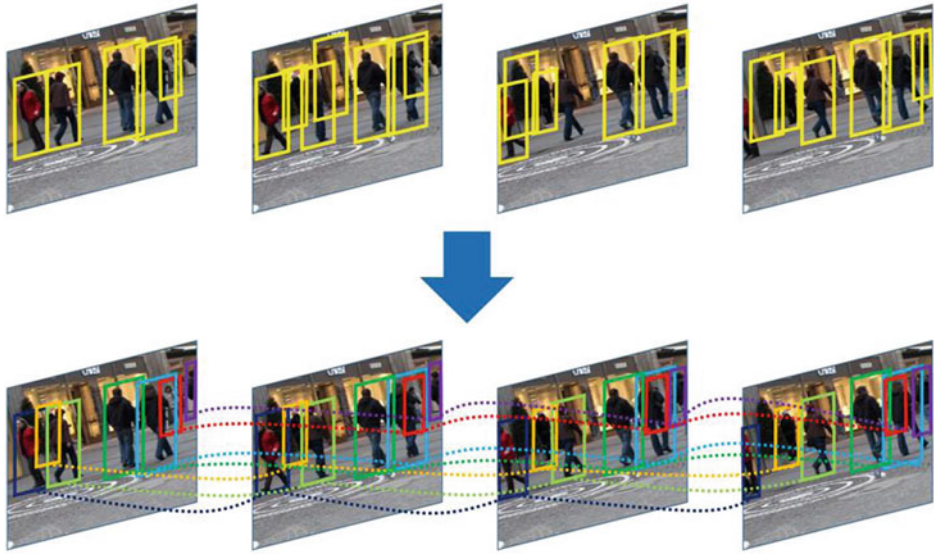


Fig. 2.40 The upper row represents the detected bounding box of different object in each frame, the lower row represents the trajectories generated by merging the relationship of detected objects among consecutive frames. Different colors denotes different objects

MOT is a mid-level task and the object detection is the foundation for MOT. Moving objects such as pedestrians, vehicles and so on are first detected and followed by identities maintaining and trajectories yielding. Besides, MOT underpins high-level tasks such as pose estimation, action recognition, and behavior analysis. Due to the sufficient modelling of the locations and movement of the scene targets, MOT plays an important role in numerous practical applications, for example, intelligent transportation, security surveillance and human computer interaction.

In the rest of this section, the common challenges of this task will be introduced. We then show the popular benchmarks and evaluation metrics as well as overview of common and effective methods. Finally, we present two novel MOT methods in detail which introduce *Tracklet-Plane Matching* and *Spatio-Temporal Point Process* respectively.

2.2.2 Challenges

MOT can be viewed as an extension to Single Object Tracking (SOT). SOT enables primary focuses on overcoming serious challenges such as scale changes, out-of-plane rotations and illumination variations brought by sophisticated appearance and motion in the scene, since there is only one object of interest and it is always in the considered image scene. However, in the MOT context, multiple objects with similar appearances and geometries in the searching

area may confuse the single object tracker. Hence, MOT has two additional requirements: determining the number of objects, which typically varies over time, and maintaining their identities.

Apart from the common challenges in both SOT and MOT, we further discuss a few challenging problems in MOT:

- *Frequent occlusions* is one of the the most critical problems for MOT. It is a primary cause for ID switches or fragmentation of trajectories.
- *Initialization and termination of tracks* is a common problem since some targets may disappear or reappear in the view which may have negative impact on MOT methods.
- *Similar appearance* often happen in large public scene such as school, gym and so on, where pedestrian objects wear similar outfits. This will pose a challenge to MOT methods which heavily rely on associating the individual objects among consecutive frames into trajectories.
- *Interactions among multiple objects* can be effectively solved by interaction model. In a crowd scenario, an object would experience some influence by other agents and objects, which can serve as cues for MOT method.

2.2.3 Datasets and Metrics

This section will talk about popular public dataset as well as common evaluation metrics.

Evaluation Metrics

The two primary metrics that are commonly considered while evaluating MOT algorithm are Multiple Object Tracking Precision (MOTA) and Multiple Object Tracking Precision (MOTP) [13].

MOTA. Given the number of miss detection FN_t where object is present in ground truth but not detected by the detection algorithm, of false positives FP_t where object is not present in the ground truth, but the detection algorithm detects it as objects, and of mismatch error IDS_t where object in ground truth is falsely related to some other object due to false tracking, for time t. The MOTA can be derived as:

$$\mathrm{MOTA} = 1 - \frac{\Sigma_t(\mathrm{FN}_t + \mathrm{FP}_t + \mathrm{IDS}_t)}{\Sigma_t \mathrm{GT}_t} \tag{2.73}$$

where GT_t is the number of ground truths for time t. As can be seen, MOTA combines the false negative rate, false positive rate and mismatch rate, measuring the overall tracking performance of an approach. MOTA is by far the most widely accepted evaluation metric for MOT.

MOTP.Given the distance between the localization of objects in the ground truth and the detection output $d_{i,t}$ and total matches made between ground truth and the detection output c_t, for object i and time t. The MOTP can be calculated with the following formula:

$$\text{MOTP} = \frac{\Sigma_{i,t}(d_{i,t})}{\Sigma_t c_t} \tag{2.74}$$

where $d_{i,t}$ can be measured by overlaps or distance. Compared with MOTA that deals with both tracker output and detection output, MOTP deals with only detection output from the model and not doing anything with tracker output.

Public Dataset and Challenge

First, we list and make comparison for all the existing MOT datasets (Table 2.19).

Among them, MOT [107, 144] and KITTI [65] become the mainstream benchmark datasets in recent years.

MOT. There are two version for MOT, MOT15 [107] and MOT16 [144]. MOT15 consists of 22 high quality videos which are subdivided into a training-set (11 videos) and a test-set (11 videos) for evaluation. MOT15 gives the largest diversity in terms of resolution ratio and with the presence of large variations in camera motion and viewpoint and involves numerous interactions and occlusions among individuals. MOT15 uses ACF [42] as the object detector.

MOT16 contains 7 videos for training and 7 videos for testing. Although with less videos, MOT16 includes more frequent occlusions, spatial density, longer duration and temporal appearance changes compared with MOT15. It is the most challenging data set to date. MOT16 uses DPM [56] as the object detector.

KITTI. KITTI is a multi-object dataset in which 8 different classes have been labeled and provides a two-classes evaluation in the benchmark including 'Car' and 'Pedestrian'. It consists of 21 training videos and 29 test ones, with a total of about 19000 frames (32 minutes), spanning multiple occurrences of common multiple object tracking challenges such as frequent occlusions, similar appearance and multiple iteraction.

Table 2.19 Comparison among existing datasets. "Multi-view" and "GT" denotes whether multi-view data and ground truth provided. "Indoor" indicates that the the dataset provides indoor scenarios, otherwise, only the outdoor scenarios are provided

	PET [154]	UCF [2]	ParkingLOT [169]	ETH [48]	KITTI [65]	MOT [107, 144]		**HiEve** [124]
Videos	13	3	3	8	50	22	14	**32**
Frames	–	1.3k	2.7k	4k	–	11k	11k	**50k**
Multi-view	✓	✗	✗	✓	✓	✗	✗	✗
GT	✓	✓	✓	✓	✓	✓	✓	✓
Indoor	✗	✗	✗	✗	✗	✗	✓	✓

2.2.4 Overview of Methods

We admit that there are not only one criteria that can reasonably categorize various MOT methods. Based on the tracking paradigm, MOT can be categorized into tracking-by-detection method, joint-detection-and-tracking method and query-based method.

Tracking-by-Detection Method

Tracking-by-detection method is a framework that separates the processing of detection and tracking. These methods usually extract set of detection results from the video frames which are then used to guide the tracking process. They follow a four-step paradigm including object detection, object feature extraction, similarity calculation and similarity-based association.

Due to the low accuracy of early object detection algorithms, most of the MOT methods focused on designing complicated mathematical model and optimizing data association. Multiple Hypothesis Tracking (MHT) [18] assumes a tracking tree that contains all the potential tracking possibility for each candidate target. The most possible association can be obtained by calculating the probability from the tracking tree. MHT considers global and context information while resulted in large computational complexity. Joint probabilistic data association (JPDA) [58] builds on the assumption that the measurements originated from the target have Gaussian densities and the false measurements are uniformly distributed. The target state estimate is updated by the best association according to all the probability of feasible associations.

With the continuous development of object detection based on deep neural network, The MOT methods based on advanced object detector and feature descriptor have been widely studied. Simple Online and Realtime Tracking (SORT) [16] leverages Faster R-CNN [160] for the detection. Kalman filter and Hungarian method, are employed to handle the motion prediction and data association, respectively. SORT was ranked as the best-performing open source algorithm on the MOT15 dataset. The SORT algorithm was later refined by many other researchers. DeepSORT [204] is new version of SORT. DeepSORT adds an extra pedestrian ReID network for integrating deep appearance information but places much of the computational complexity. As YOLO iteratively improves, DeepSORT-YOLO was proposed to replace the Faster R-CNN [160] by YOLO for real-world scenarios. MOTDT [25] integrates the missing detection with new candidates and reaches competitive performance on MOT.

Some other researchers paid attention to modify the similarity calculation and similarity-based association by deep neural network. In Dual Matching Attention Networks (DMAN) [233], spatial and temporal attention mechanism is introduced to enable the the network to focus on the matching patterns and suppress the impact of mis-matching. LSST[57] make matching decision by utilizing the short term and long term cues from a SOT sub-net and a ReID sub-net.

Moreover, researches on target association based on graph neural network (GNN) begin flourishing recently. MPNN [21] models data association by using a graph, where each

detected ReID feature is represented as a node, and edges indicates possible link among them. As the GNN operate and feature update, each node of the graph integrates the features of its adjacency and obtain more accurate representation. In GNMOT [113], an appearance graph network and a motion graph network are separately used to update the appearance node features and motion node features, which are then fuse to compute the similarity. EDA_GNN [87] additionally update the edge feature which is input to a a multi-layer perceptron to directly output the final matching results. EDA_GNN enables the simultaneous learning of the process of feature updating and matching, achieving better tracking performance.

Joint-Detection-and-Tracking Method

As mentioned above, most existing methods separate the process of detection and tracking, which causes additional computational complexity and running speed reduction. Besides, the gradients of the two process cannot be shared and the information is not fully utilized. Thus, joint-detection-and-tracking methods propose to coupling the detection and tracking by sharing the vast majority of the weights between detection and tracking network. In doing this, MOT can take advantage of multi-task learning through joint optimization of detection and tracking feature extraction in the process back-propagation. JDE [202] adds an extra branch of appearance embedding learning which shares the same feature map with the detection network, such that the detection results and the corresponding embedding can be simultaneously output. Different from JDE that uses RPN [160] as detection network, FairMOT [226] is based on anchor-free detector CenterNet [230]. Besides, FairMOT trains the tracking embedding learning model through a classification task and outperforms the JDE on both running speed and accuracy. However, the tracking strategy of these methods is much simpler than those of the tracking-by-detection methods. Although joint-detection-and-tracking methods are more efficient, the improvement benefits more from the accurate detectors, which results in unsatisfactory association and a large number of ID switches. In light of CenterNet, CenterTrack [229] discards object feature extraction and adopts the object movement prediction among frames. However, due to the lack of use of appearance features, it is prone to tracking failure for long-term occlusion. GSDT [200] consider the fact that object detection and data association are dependent on each other, and use GNN to operate feature fusion on historic trajectory and candidate point of current feature map.

Query-Based Method

Query-based method is mainly inspired by SOT methods. The main idea is to take each historical target detected in the beginning frames as a query and obtain the new position in the current frame, which simplifies the MOT pipeline by getting rid of the process of target association. Tracktor [11] takes an object in the last frame as proposal and exploits the second-stage bounding box regression of Faster R-CNN to predict the new position in the next frame. TrackFormer [143] is on the basis of DETR [22]. It regards the object in the last frame and the image feature in the current frame as the query and key of the next frame, respectively. The position corresponding to the query in the new frame can be

obtained through the encoder-decoder Transformer [188]. TrackFormer can deal with new-coming tracks by setting extra random initialized query. TransTrack [177] takes new object and historical object as a query of the current frame to enable detecting in current frame and then applies IOU-matching between detection and historical object query. TransCenter [209] modifies CenterTrack into a query-based method where the detection query enables inferring the center and scale of object's bounding box globally and robustly and, in parallel, the tracking query is used for obtaining the offset of the center.

2.2.5 Tracklet-Plane Matching

Motivation

Most of the existing MOT approaches focused on developing a proper object association strategy such that objects in different frames are optimally associated under some cost functions. However, since different objects in a video may be confused due to their similarity in appearance or motion, the association results are often interfered with by these confusing detections. Some researchers aim to reduce this interference by developing more differentiable feature representations or similarity metrics. These methods still have limitations when handling highly confusing objects. At the same time, since noisy detections are inevitable in the association process, their results are also interfered with by these noisy detections. Some recent approaches aim to reduce the confusion from noisy detections by introducing more accurate object detectors, or developing more reliable object-wise similarity metrics. However, they do not discriminate good detections from noisy ones very well, which in turn, impacts their performance when handling visually similar or easily confusable noisy detections (Fig. 2.41).

Tracklet-Plane Matching Process

The proposed tracklet-plane matching process aims to associate detections of high similarity into short tracklets, group these highly-related short tracklets into tracklet planes, and further associate these in-plane tracklets into long trajectories. In short, it contains three main steps: tracklet construction, tracklet-plane construction and in-plane tracklet matching. An overview of our approach is shown in Fig. 2.46.

Tracklet construction

The tracklet construction process merges highly-related objects into tracklets, which will be used as the basic units in the association process later. In this paper, we first use a min-max normalization process to evaluate the confidence of each detected object; those of low confidence are excluded. Then, we apply Kuhn-Munkres (KM) algorithm [105] to dynamically associate temporally related objects into short tracklets [191]. During object

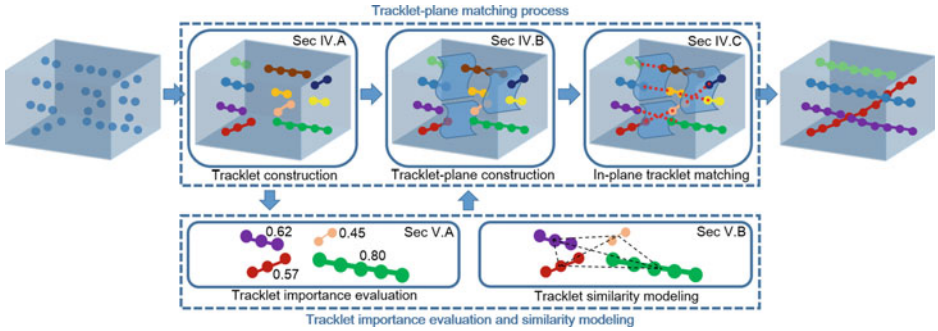


Fig. 2.41 An **overview** of TPM. we first associate initial detections (blue dots) into tracklets in the *tracklet construction* module. Then the importance of tracklets and the similarity among tracklets are measured in the *tracklet importance evaluation* and *tracklet similarity modeling* modules. Based on the evaluated tracklet importance and similarity information, the *tracklet-plane construction* module builds a set of tracklet-planes, in which each tracklet-plane is connected to several tracklets. Finally, the *in-plane tracklet matching* module merges or associates tracklets within each tracklet-plane, and obtains the final long trajectories

association, we model the similarity between an object D in frame t and a tracklet T constructed in the previous frame $t-1$ as:

$$S_{to}(T, D) = A(T, D) + \lambda_s M(T, D), \tag{2.75}$$

where $S_{to}(T, D)$ represents the similarity between tracklet T and object D. $A(T, D)$ is the appearance similarity and $M(T, D)$ is the motion similarity. $\lambda_s = 0.5$ is the weight balancing the importance between the appearance similarity and the motion similarity. We compute the cosine similarity between *pool5* features of ResNet-50 [72] as the appearance similarity, while the velocity and position information of the tracklet and the object is computed to obtain the motion similarity [216].

Tracklet-plane construction

After obtaining short tracklets, we need to further associate them to form long trajectories. However, due to the interferences of confusing or noisy detections, tracklets belonging to the same trajectory may become temporally disconnected, temporally overlapping, or created by noisy detections. Directly applying association methods [10, 64, 180, 214] may lead to low performances.

To address the aforementioned interferences, we develop a tracklet-plane matching method to organize related tracklets into planes and apply association methods in each tracklet plane. This resolves the association confusions caused by noisy or missing

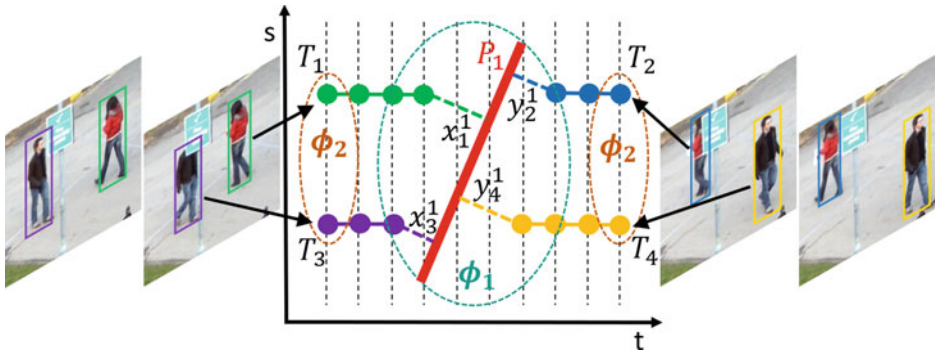


Fig. 2.42 Schematic diagram with symbols. t axis represents time and s axis represents space

detections. In our method, the optimization function that constructs the tracklet-planes is given by:

$$\begin{aligned}(X^*, Y^*, n_p^*) &= \arg\min_{X,Y,n_p} \Phi_1(X, Y, n_p) + \Phi_2(X, Y, n_p) + \lambda_p n_p, \\ s.t. \quad & x_i^m, y_j^m \in \{0, 1\}; \sum_{m=1}^{n_p} x_i^m \leq 1; \sum_{m=1}^{n_p} y_j^m \leq 1\end{aligned} \tag{2.76}$$

where $X = \{x_i^m\}, i = 1, ..., n_t, m = 1, ..., n_p$ and $Y = \{y_j^m\}, j = 1, ..., n_t, m = 1, ..., n_p$ are the sets representing the tracklet-plane construction status of all tracklets in a video, $x_i^m = 1$ indicates the end of tracklet t_i that is connected to tracklet-plane P_m and $y_j^m = 1$ indicates the start of tracklet t_j that is connected to tracklet-plane P_m (See Fig. 2.42). n_t is the total number of tracklets and n_p is the total number of tracklet-planes. $\lambda_p = -0.1$ is the balancing weight. The constraints guarantee that the start and the end of every tracklet are connected to at most one tracklet-plane. $\Phi_1(X, Y, n_p)$ and $\Phi_2(X, Y, n_p)$ are the optimization terms for evaluating tracklet-plane construction qualities, which are defined in Eq. 2.77 and Eq. 2.78, respectively:

$$\Phi_1(X, Y, n_p) = -\sum_{m=1}^{n_p} \sum_{i=1}^{n_t} \sum_{j=1}^{n_t} 2 x_i^m y_j^m W_i W_j S_{tt}(T_i, T_j), \tag{2.77}$$

$$\Phi_2(X, Y, n_p) = \sum_{m=1}^{n_p} \sum_{i=1}^{n_t} \sum_{j=1}^{n_t} (x_i^m x_j^m + y_i^m y_j^m) W_i W_j S_{tt}(T_i, T_j), \tag{2.78}$$

where W_i represents the importance of tracklet T_i, which is evaluated by the tracklet importance evaluation scheme. $S_{tt}(T_i, T_j)$ denotes the similarity between tracklet T_i and tracklet T_j, which is calculated by the representative-based similarity modeling scheme.

Inference. Solving Eq. 2.76 is not trivial as the optimization terms and constraints are discrete and complicated. In this paper, we use Local Gradient Descent algorithm [5] to obtain an approximate solution that iteratively derives candidate tracklet-plane solutions by sequentially connecting tracklets to neighboring planes to find the best one.

We set S as the matrix combined with tracklet similarity and tracklet importance:

$$S_{ij} = W_i W_j S_{tt}(T_i, T_j), \forall i, j \in [1, n_t], \tag{2.79}$$

then $\Phi_1(X, Y, n_p)$ and $\Phi_2(X, Y, n_p)$ can be expressed as:

$$\Phi_1(X, Y, n_p) = -2 \sum_{m=1}^{n_p} X_m^T S Y_m, \tag{2.80}$$

$$\Phi_2(X, Y, n_p) = \sum_{m=1}^{n_p} (X_m^T S X_m + Y_m^T S Y_m), \tag{2.81}$$

where X_m is the mth column of matrix X and Y_m is the mth column of matrix Y. Due to that S is a symmetric matrix, the optimization function $\Phi(X, Y, n_p)$ can be expressed as:

$$\Phi(X, Y, n_p) = \sum_{m=1}^{n_p} (X_m - Y_m)^T S (X_m - Y_m) + \lambda_p n_p, \tag{2.82}$$

let $A = X - Y$, then:

$$\Phi(A, n_p) = \sum_{m=1}^{n_p} A_m^T S A_m + \lambda_p n_p, \tag{2.83}$$

the function in Eq. 2.83 can be made convex by the normalized Laplacian matrix of S:

$$\widehat{S} = I - G^{\frac{1}{2}} S G^{\frac{1}{2}}, \tag{2.84}$$

where I is the identity matrix and G is the diagonal matrix by the row sums of S. Therefore, $\Phi(X, Y, n_p)$ is convex for both X and Y and we can apply the Local Gradient Descent algorithm to solve Eq. 2.76.

In-plane tracklet matching

After connecting related tracklets onto tracklet-planes, we are able to perform tracklet-wise association (i.e, matching) within each tracklet-plane to obtain final trajectories. Particularly, the in-plane tracklet matching process aims to find the best one-to-one matching among tracklets which are connected to different sides of a tracklet-plane. Therefore, the process can be modeled as:

$$Z^* = \arg\max_{Z} \sum_{m=1}^{n_p} \sum_{i=1}^{n_t} \sum_{j=1}^{n_t} x_i^m y_j^m z_{ij} W_i W_j S_{tt}(T_i, T_j),$$
$$s.t. \quad z_{ij} \in \{0, 1\}; \sum_{j=1}^{n_t} z_{ij} \leq 1; \sum_{i=1}^{n_t} z_{ij} \leq 1 \tag{2.85}$$

where $Z = \{z_{ij}\}, i = 1...n_t, j = 1...n_t$ is the set representing the tracklet association status. $z_{ij} = 1$ means the end of tracklet T_i is connected to the start of tracklet T_j (i.e, T_i and T_j are associated). x_i^m and y_j^m are obtained by Eq. 2.76, which guarantees that only the tracklets from both sides of the same tracklet-plane can be associated. Finally, KM algorithm [105] is applied to solve Eq. 2.85.

Tracklet Importance Evaluation and Similarity Modeling

To accurately discriminate and exclude noisy detections/tracklets in the tracklet-plane matching process, it is important to find effective ways to evaluate tracklet reliability and model tracklet-wise similarity. To this end, we propose a tracklet-importance evaluation scheme and a representative-based similarity modeling scheme.

Tracklet importance evaluation

The importance of tracklet T_i is calculated by:

$$W_i = \frac{\sum_{n=1}^{L_i} C_i^n}{L_i} \frac{\sum_{n=1}^{L_i-1} S_{oo}(D_i^n, D_i^{n+1})}{L_i - 1} (1 - e^{-\sqrt{L_i}}), \tag{2.86}$$

where D_i^n denotes the n-th object of tracklet T_i, C_i^n represents the confidence of detection D_i^n similar to those by [42, 56, 160, 213], $S_{oo}(D_i^n, D_i^{n+1})$ represents the appearance similarity between two adjacent objects D_i^n and D_i^{n+1}, and L_i represents the length of tracklet T_i.

Representative-based similarity modeling

Tracklet similarity $S_{tt}(T_i, T_j)$ is another key factor affecting the performance of the tracklet matching process. Intuitively, tracklet-wise similarity can be modeled by the similarity between the features of tracklets' terminal objects [76, 180, 214, 216] (Fig. 2.43a) or between the global features of tracklets [115, 140] (Fig. 2.43b). However, since tracklets may include noisy detections, directly using terminal object features or global features may improperly introduce noisy information and reduce the reliability of the similarity measure.

Therefore, we propose a representative-based similarity modeling scheme, which introduces a neural network to select the most representative objects from each tracklet and use them to model tracklet similarity. Our proposed representative-based similarity modeling scheme is shown in Fig. 2.43c. When calculating the similarity $S_{tt}(T_i, T_j)$ between two tracklets T_i and T_j, we first input them into a representative-selection network and select

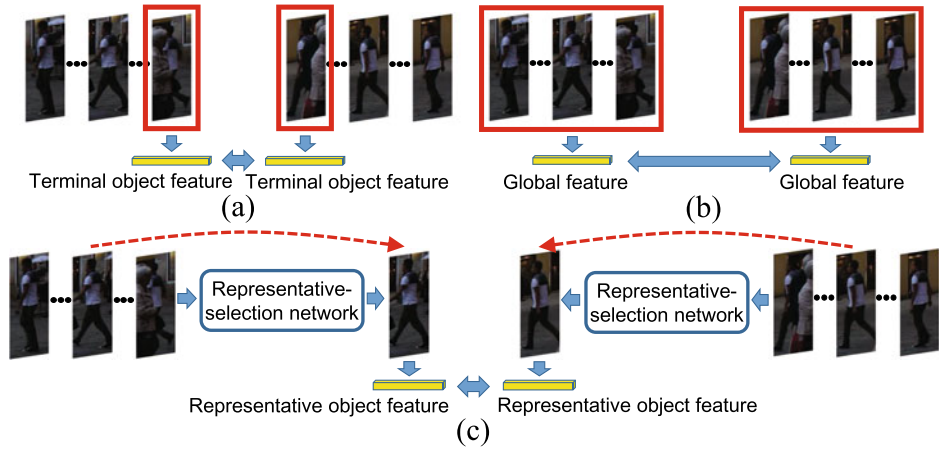


Fig. 2.43 Three tracklet similarity measurement strategies: **a** By the features of terminal objects. **b** By the global features. **c** By the features of representative objects (Ours)

the most reliable representative object from each tracklet. Then, the tracklet-wise similarity between T_i and T_j is calculated by the feature similarity between the selected representative objects.

Representative-selection network. In order to capture the temporal variation in a tracklet, we adopt convolutional LSTM [167] as the major structure of the representative-selection network. Moreover, we select two representative objects from each tracklet: one is used to measure the similarity with tracklets *before* it, while the other is used to measure the similarity with tracklets *after* it. Therefore, we develop a bi-directional convolutional LSTM as the representative-selection network, as depicted in Fig. 2.44.

Learning and loss function. In order to make the representative-selection network select proper objects, we need to define proper ground-truth representative scores to guide the learning process. In this paper, we define the ground-truth by

$$V_i^t = \frac{1}{\gamma}\sum_{\tau \epsilon G}(S_{oo}(D_i^t, D_i^{t+\tau}) - \max_{j \neq i}(S_{oo}(D_i^t, D_j^{t+\tau}))), \tag{2.87}$$

where D_i^t is obtained from the training dataset and it denotes the ground-truth detection box of the ith tracklet in frame t. V_i^t is the ground-truth representative score for D_i^t. S_{oo} denotes the appearance similarity between two detections, which is defined in the same way as in Eq. 2.86. The set G represents the frame clip of the tracklet. For the forward stream, the set $G = \{1, 2, \ldots \gamma\}$, while for the backward stream, $G = \{-1, -2, \ldots -\gamma\}$.

With the ground-truth representative scores determined by Eq. 2.87, we define the loss function of the representative-selection network as:

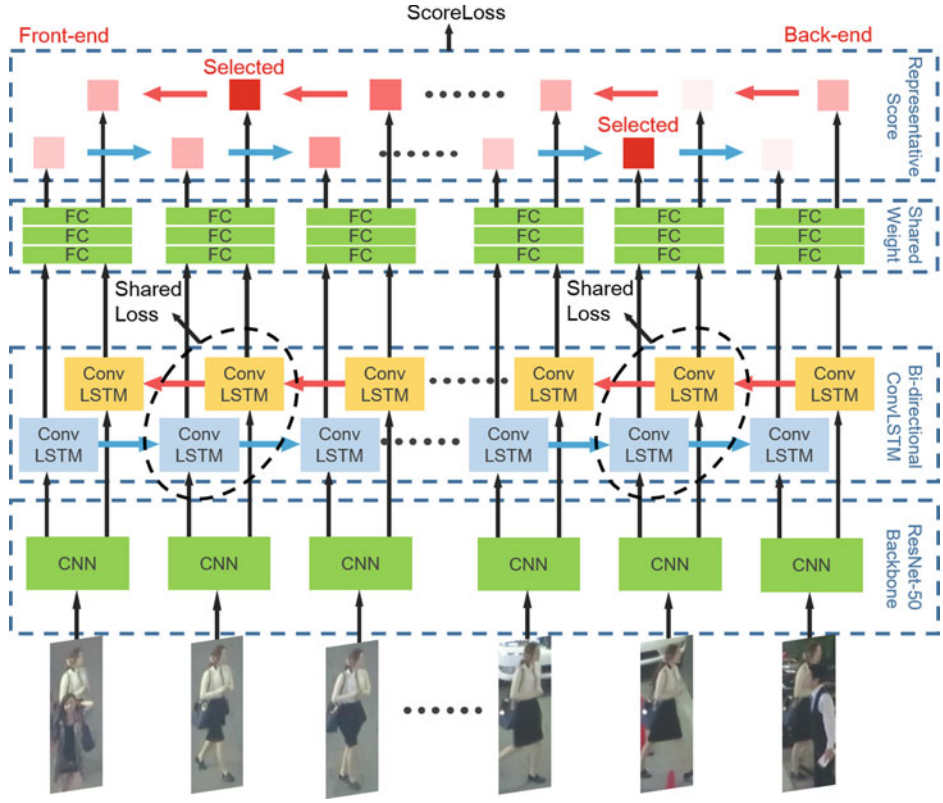


Fig. 2.44 The representative-selection network. Bi-directional convolutional LSTM units are merged into the representative-selection network. Two score values are calculated from two streams for each object based on the network. The object with the largest score (the brightest red square) is selected for tracklet similarity calculation

$$\mathcal{L} = \mathcal{L}_F + \mathcal{L}_B + \mathcal{L}_P, \tag{2.88}$$

$$\mathcal{L}_F = \frac{1}{\gamma} \sum_{\tau=0}^{\gamma} \left\| y_i^{t_B-\gamma+\tau} - V_i^{t_B-\gamma+\tau} \right\|^2, \tag{2.89}$$

$$\mathcal{L}_B = \frac{1}{\gamma} \sum_{\tau=0}^{\gamma} \left\| y_i^{t_F+\gamma-\tau} - V_i^{t_F+\gamma-\tau} \right\|^2, \tag{2.90}$$

where $\mathcal{L}_F$ and $\mathcal{L}_B$ are the losses for the forward and backward streams, respectively. $\mathcal{L}_P$ denotes the aforementioned similarity supervision between two feature maps extracted from bi-directional convolutional LSTM units. $y_i^t = f(D_i^t)$ is the prediction score from this network and the function $f(\cdot)$ represents the representative-selection network. $V_i^{t_B-\gamma+\tau}$ and

$V_i^{t_F+\gamma-\tau}$ are the ground-truth representative scores for the forward and backward streams, respectively. t_B denotes the back-end frame of tracklet T_i, and t_F denotes the front-end frame of the tracklet.

Experiments

We perform experiments on two benchmark datasets: MOT16 [144] and MOT17. Based on the MOTChallenge Benchmark, tracking performance is measured by Multiple object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP), the total number of False Negatives (FN), the total number of False Positives (FP), the total number of Identity Switches (IDs), the percentage of Mostly Tracked Trajectories (MT) and the percentage of Mostly Lost Trajectories (ML). Specifically, MOTA measures the overall tracking performance of an approach, together with FN, FP, and IDs (Table 2.20).

Ablation study

To evaluate the effectiveness of different components in our approach, we compare the following six methods:

- *Baseline*. Directly applying KM algorithm to associate the detected objects into trajectories without constructing tracklets.
- *Softassign*. Using KM algorithm to generate reliable tracklets and applying Softassign algorithm [191] to associate the tracklets into trajectories. The terminal detection in each tracklet is used to calculate tracklet-wise similarity, as shown in Fig. 2.43a.
- *TP*. Using our tracklet-plane matching process to perform tracking, but excluding both the tracklet-importance evaluation scheme and the representative-based similarity modeling scheme(i.e, setting all importance weight W in Eqs. 2.77–2.85 to 1 and simply using the terminal detection in each tracklet to calculate tracklet-wise similarity, as shown in Fig. 2.43a).

Table 2.20 Ablation study on MOT16 validation dataset

Method	MOTA↑	MOTP↑	MT↑ (%)	ML↓ (%)
Baseline	39.6	75.4	14.8	49.1
Softassign	40.5	75.4	15.3	48.2
TP	42.1	75.4	16.3	46.7
TP+Imp	43.0	75.4	16.7	45.8
TP+Imp+Glob	43.8	75.5	18.0	43.4
TP+Imp+Rep (TPM)	**44.7**	**75.5**	**19.1**	**40.8**

- *TP+Imp*. Using our tracklet-plane matching process to perform tracking, which includes the tracklet-importance evaluation scheme but excludes the representative-based similarity modeling scheme.
- *TP+Imp+Glob*. Using our tracklet-plane matching process to perform tracking, including the tracklet-importance evaluation scheme. Global features of tracklets [140] are used to measure tracklet-wise similarity.
- *TP+Imp+Rep (TPM)*. Using the full version of our proposed approach by including both the tracklet-importance evaluation scheme and the representative-based similarity modeling scheme.

Table 2.22 compares the tracking results on the MOT16 validation dataset and Fig. 2.45 shows several tracking results of different methods on a sample test sequence (MOT16-06). From Table 2.22 and Fig. 2.45, we can observe that:

(1) *TP* performs significantly better than *Baseline* and *Softassign*. This indicates that directly performing the association of detected objects or simply using a general tracklet association algorithm can be easily interfered with by noisy detections and similar objects, leading to unsatisfactory results.

(2) *TP+Imp* performs better than *TP*. This demonstrates that our tracklet-importance evaluation scheme can effectively evaluate the reliability of tracklets and provide track-

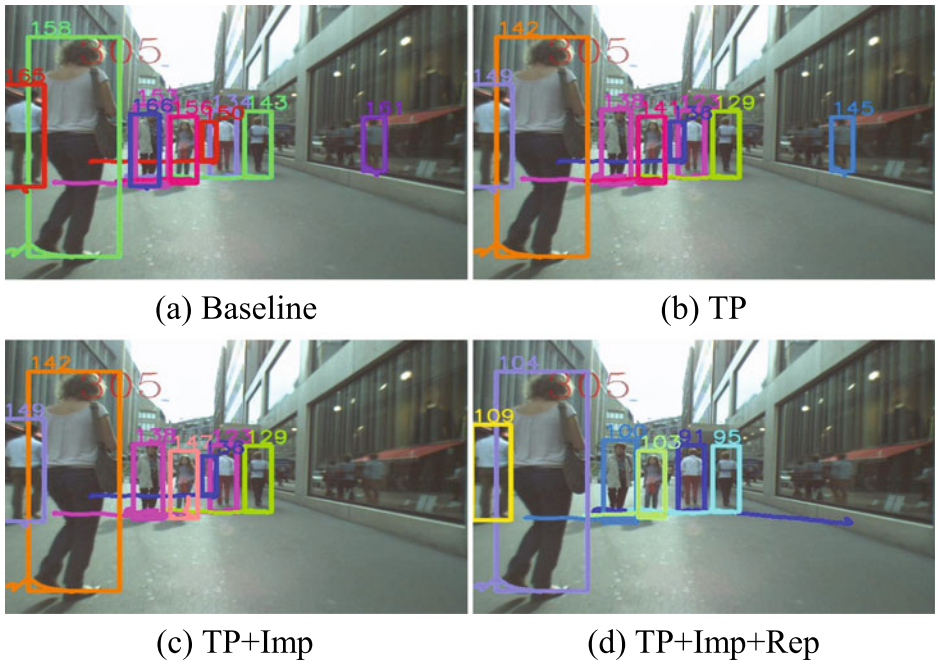


Fig. 2.45 Qualitative comparisons of different methods on a test sequence (MOT16-06)

Table 2.21 Tracking performance of TPM and state-of-the-art methods on MOT16 and MOT17 test datasets

Dataset	MOT16						
Method	MOTA↑	MOTP↑	MT↑ (%)	ML↓ (%)	FP↓	FN↓	IDS↓
oICF [96]	43.2	74.3	11.3	48.5	6651	96515	381
CDA-DDAL [8]	43.9	74.7	10.7	44.4	6450	95175	676
Quad-CNN [171]	44.1	76.4	14.6	44.9	6388	94775	745
STAM [31]	46.0	74.9	14.6	43.6	6895	91117	473
JMC [181]	46.3	75.7	15.5	39.7	6373	90914	657
NOMT [30]	46.4	76.6	18.3	41.4	9753	87565	**359**
NLLMPa [108]	47.6	78.5	17.0	40.4	5844	89093	629
LMP [182]	48.8	**79.0**	18.2	40.1	6654	86245	481
TPM (Ours)	**50.9**	74.9	**19.4**	**39.4**	**4866**	**84022**	619
Dataset	MOT17						
MHT-bLSTM [98]	47.5	**77.5**	18.2	41.7	25981	268042	2069
PHD-GSDL [59]	48.0	77.2	17.1	35.6	23199	265954	3998
DMAN [233]	48.2	75.9	19.3	38.3	26218	263608	2194
EDMT [24]	50.0	77.3	21.6	36.3	32279	247297	2264
MHT-DAM [97]	50.7	**77.5**	20.8	36.9	22875	252889	2314
MOTDT [128]	50.9	76.6	17.5	35.7	24069	250768	2474
JCC [92]	51.2	75.9	20.9	37.0	25937	247822	**1802**
FWT [76]	51.3	77.0	21.4	**35.2**	24101	247921	2648
TPM (Ours)	**52.4**	76.6	**22.4**	40.0	**19922**	**246183**	2215

ing performance improvement by discriminating and excluding the interference of noisy detections and tracklets.

(3) *TP+Imp+Rep* outperforms *TP+Imp* and *TP+Imp+Glob*, which shows that our proposed representative-based similarity modeling scheme calculates the tracklet-wise similarity more accurately by selecting the most reliable detection in a tracklet (Table 2.21).

Comparison with State-of-the-art Methods

Finally, Table 2.23 compares the proposed TPM with the state-of-the-art MOT methods on the test sequences of MOT16 and MOT17 datasets. For a fair comparison, all the methods are performed based on the same public detection results.

From Table 2.23, we make the following observations:

(1) TPM significantly outperforms existing MOT methods in terms of MOTA (the primary metric) on both MOT16 and MOT17, which demonstrates the effectiveness of our approach.

(2) Our TPM approach produces the highest MT and the lowest FN on both MOT16 and MOT17. This shows that our TPM algorithm can associate the tracklets accurately and cater for the missing detections correctly. On the other hand, the MOTP of our approach is slightly lower than some methods because the interpolated detections tend to be ineffective when there are significant camera motions.

(3) On both MOT16 and MOT17, our approach produces the lowest FP among all methods. This demonstrates TPM's advantage of effectively handling the visually-similar or easily confusable objects by discriminating and excluding the less important tracklets and selecting more representative detections for tracklet similarity modeling.

Conclusion

This paper introduces a new approach which can alleviate the problem of noisy object detections in MOT. Our approach consists of two key ingredients: (1) a tracklet-plane matching process which organizes related tracklets into planes and perform in-plane tracklet matching to associate tracklets into long trajectories; (2) a tracklet-importance evaluation scheme and a representative-based similarity modeling scheme which can properly evaluate and differentiate the reliability of detection results and pick up reliable ones during association. Experiments on the MOT16 and MOT17 benchmarks demonstrate the effectiveness of our approach.

2.2.6 Spatio-Temporal Point Process

Motivation

One major issue of MOT is that some "bad" detection results always hinder the performance of MOT. In general, such "bad" results can be divided into two different types: (1) noisy detection results, i.e. false positives in object detection, and (2) confusing detection results, i.e. two highly overlapping objects with similar appearances. Both the noisy and confusing objects had misguided the matching process and most of the failure examples in MOT are caused by them, directly or indirectly. To tackle this issue in MOT, First, we note that the "bad" detections can be formulated as a sequence of events that happen in different frames and locations. Thus, we need to infer when and where these events are likely to happen, given the feature of current frame and the historical behaviour of the detector as prior information. More specifically, we model these events based on the pixels that are inside the bounding boxes of "bad" detections. Such events are distributed across the spatio-temporal tube of a video, and are generated based on complicated latent mechanisms, which is hard to capture

through simple modeling. Because these events happen in the motion of objects, so there exists relationship between these events which actually reflects the motion information of the objects.

Problem Formulation

The input data is a sequence $C = \{I_1, I_2, ...\}$ of consecutive frames in a video, combined with the object detection results $D_j = \{d_1^j, d_2^j, ...\}$ in each frame I_j. Note that the detection results D_j are generated by some baseline detection algorithms and are not guaranteed to be entirely correct. For example, the false positive detections that always occur in crowded scenes are regarded as noisy detections. Besides, other erroneous detection results are caused by confusion from detected objects of similar appearances which overlapped with each other during motion. The MOT task aims to model the relationship among these detected objects and associate them into trajectories. Hence these noisy or confusing detection results will hinder the performance of MOT. In this paper, we adopt the spatio-temporal point process to address this issue.

Let $B_j = \{b_1^j, b_2^j, ...\}$ denote the set of "bad" detections in each frame I_j (note that $B_j \subseteq D_j$). In our spatio-temporal point process, we define the events e to be the pixels $e = (t, x, y)$ that are inside some "bad" detection results b_i^t in frame I_t. We use $\mathcal{E} = \{e_1, e_2, ..., e_n\}$ to denote the set of events from all frames. In general, the dynamics of spatio-temporal point process is captured by its conditional intensity function $\lambda^*(t, x, y)$ (where $*$ emphasizes that this function is conditional on the history), which can be defined as:

$$\lambda^*(t', x', y') = \frac{\partial^3 E[N(t, x, y)|\mathcal{H}_t]}{\partial t \partial x \partial y}|_{t=t', x=x', y=y'} \tag{2.91}$$

where $\mathcal{H}_t$ denotes the history of events before time t and $N(t, x, y)$ is the counting function which counts the number of events with spatial coordinates less than (x, y) and temporal coordinates less than t, while $E[N]$ is the expectation of the counting function. Since we assume that the point process satisfies the regularity condition:

$$P\{N([t, t+\epsilon_1], [x, x+\epsilon_2], [y, y+\epsilon_3]) > 1\} = o(\epsilon_1\epsilon_2\epsilon_3) \tag{2.92}$$

where P measures the probability of event (its argument), $[t, t+\epsilon_1]$, $[x, x+\epsilon_2]$, $[y, y+\epsilon_3]$ indicates a small area around the point (t, x, y) while $o(\epsilon_1\epsilon_2\epsilon_3)$ refers to a term of higher-order infinitesimal *w.r.t* $\epsilon_1\epsilon_2\epsilon_3$. Equation 2.92 shows that in point process there will be at most one event within a short time and in a small area. The intensity function $\lambda^*(t, x, y)$ can also be interpreted as the conditional probability density that an event occurs at (t, x, y). Thus the probability density for an event sequence $\{e_1, e_2, ...\}$ can be written as:

$$f(e_1, e_2, ...) = \prod_j f^*(t_j, x_j, y_j|\mathcal{H}_{t_j}) \tag{2.93}$$

where

$$f^*(t, x, y|\mathcal{H}_{t_j}) = \frac{\lambda^*(t, x, y)}{\exp(\int_{\widetilde{t_j}}^{t} \int_{u,v} \lambda^*(\tau, u, v) d\tau du dv)} \tag{2.94}$$

Here, $\widetilde{t_j}$ denotes the maximal time-stamp t_k that satisfies $t_k < t_j$, the denominator presents the probability that no new event occurs up to time t_j since $\widetilde{t_j}$. Point processes are always learned by maximizing the likelihood function; whereby its implementation is not specified. Thus, one problem is how to define and implement the intensity function $\lambda^*(t, x, y)$ in the context of MOT.

Method

In this section, we describe in detail the method which characterizes the proposed neural network model for spatio-temporal point process. We have two RNNs in our model. The first one is for modeling the time series and the second one is for modeling the event sequence. The feature h_i^e which encodes the event sequence is aligned in time and synchronized with the feature h_t^s from time series. Then, we merge the features from two RNNs and calculate the intensity function based on it. After obtaining the intensity function, we can infer the event through sampling on the probability density function (c.f. Eq. 2.94). In the testing phase, for each bounding box b, we calculate the ratio r_b between the number of events in it and its area. Then, a detection b is treated as a "bad" detection if r_b is larger than some threshold. An overview of our approach is shown in Fig. 2.46.

Synchronous RNN

The synchronous RNN takes in time series as input data. We have two kinds of time series: one is the raw images; one is the binary maps for detection results, where the pixels that are inside a detected bounding box are masked to 1, otherwise 0. As shown in [93], though the

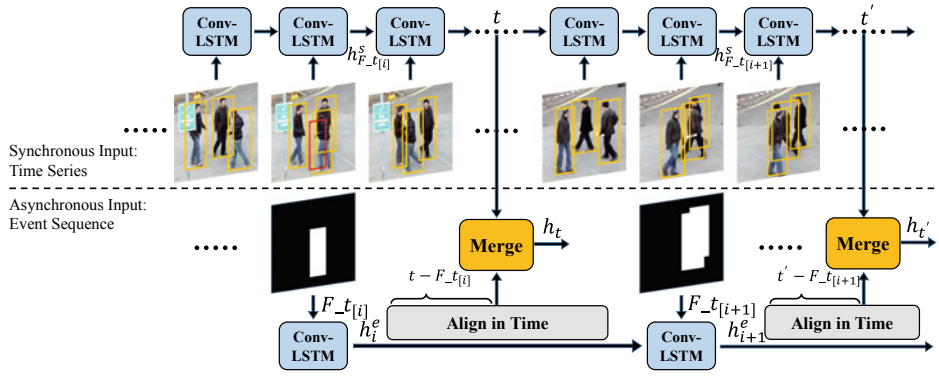


Fig. 2.46 An **overview** of proposed point process model. $[i]$ denotes the i-th equivalence class of events that have same time-stamp, so that $t_{[i]}$ means the i-th frame that contains at least one event

mask of bounding box is a weaker label than the segmentation mask of the object, it still provides sufficient information to train the segmentation task. However, we intend for the proposed model to focus on learning potential areas of "bad" detections instead of actual segmentation of objects. Thus, the binary maps of bounding boxes are used as auxiliary input data instead of training label.

At each step t, we first send the raw image I_t and the binary map M_t^D of detection results to a CNN (*e.g.* VGG, ResNet) to extract feature maps. Then the feature maps are merged and fed to the recurrent unit (conv-LSTM) to extract temporal features. Therefore, the hidden states h_t^s for the time series can be obtained by:

$$(h_t^s, c_t^s) = ConvLSTM(h_{t-1}^s, c_{t-1}^s, \psi_1(I_t, M_t^D)) \tag{2.95}$$

where ψ_1 denotes the feature extractor CNN, c_t^s denotes the cell status and both h_t^s, c_t^s are feature maps. Note that such input data is sampled uniformly over time, which can reflect the exogenous intensity in point process.

Asynchronous RNN

The asynchronous RNN aims to capture the relation between events. It takes in an event sequence as input data. In this spatio-temporal point process for MOT, each event provides both spatial and temporal features. For spatial feature, our strategy is similar to that in previous section. We use binary map to represent the spatial information of events that occur in a frame, i.e., those pixels that are inside the "bad" detections are masked to 1, otherwise 0. For the temporal feature, we use the inter-event duration $F\_t_{[i]} - F\_t_{[i-1]}$ for each event occurring at time $F\_t_{[i]}$. Here, $[i]$ denotes the i-th equivalent class of events, so $F\_t_{[i]}$ denotes the i-th frame that contains at least one event. We reformulate the scalar $F\_t_{[i]} - F\_t_{[i-1]}$ to a map that has same size as the raw image and all pixels are represented by this scalar. Then, these two maps are concatenated and passed through a CNN feature extractor and conv-LSTM to obtain the hidden state h_i^e:

$$h_i^e = ConvLSTM(h_{i-1}^e, c_{i-1}^e, \psi_2(M_i^B, F\_t_{[i]} - F\_t_{[i-1]})) \tag{2.96}$$

where ψ_2 denotes the CNN feature extractor, M_i^B denotes the binary map of "bad" detection results, c_i^e denotes the cell status and both h_i^e, c_i^e are feature maps. Note that the events are always sparsely located and unevenly distributed in the time domain, which shows that the sequences are asynchronous in nature and the temporal intervals can be very long. Thus, one advantage of such asynchronous RNN is that it can better capture long-term dependencies of events, which can reflect the endogenous intensity in point process.

Align and Merge

We aim to construct the intensity function based on the features from both time series and event sequence. To this end, the asynchronous features should be first aligned and synchronized with the time series. Traditionally, one may consider multiplying a decaying

function $\gamma(t - F\_t_{[i]})$ with the feature h_i^e to simulate the influence from historical events to current time, where $\gamma(t)$ can be specified by $\exp(-t)$. However, the latent dynamics of point process in MOT are still unknown. Directly specifying the evolving function $\gamma(t)$ in such a manner may lead to the model misspecification problem [47]. Therefore, we use a three-layer MLP whose input is the concatenation of $t - F\_t_{[i]}$ and h_i^e to learn the evolving function $\psi_3(\cdot, t)$, where the model capacity can be sufficiently large to cover any dynamic patterns. Assuming current time is t and the latest hidden state of the event e is h_i^e, we have:

$$\hat{h}_t^e = \psi_3(h_i^e, t - F\_t_{[i]}) \tag{2.97}$$

where $\hat{h}_t^e$ denotes the aligned event feature at time t. After that, we merge the outputs from both time series and event sequence to obtain $h_t = [h_t^s, \hat{h}_t^e]$. Finally, the intensity function $\lambda(t)$ can be formulated as

$$\lambda^*(t) = \sigma(w^s h_t^s + w^e \hat{h}_t^e) \tag{2.98}$$

where $\lambda^*(t)$ in this equation is an intensity map, the $w^s h_t^s$ term reflects the exogenous intensity, the $w^e \hat{h}_t^e$ term reflects the endogenous intensity, and σ denotes the activation function. A reasonable choice of the activation function would be one that ensures that the intensity function is non-negative and almost linear when the input is much greater than 0. Several activation functions fit this purpose: the *softplus* function, defined as $log(exp(x) + 1)$ or exponential linear unit *(elu)*, defined as $elu(x) + 1$ where $elu(x) = x$ when $x \geq 0$ and $elu(x) = \exp(x) - 1$ when $x < 0$.

Learning Method

Given the sequence of events $E = \{e_1, e_2, \ldots, e_n\}$, we obtain the intensity function $\lambda^*(t, x, y)$ by Eq. 2.98. Then, the log-likelihood function can be formulated based on Eqs. 2.93 and 2.94 as follows:

$$\log f = \sum_j \log \lambda^*(t_j, x_j, y_j) - \int_{t_0}^{t_n} \int_{u,v} \lambda^*(\tau, u, v) d\tau du dv \tag{2.99}$$

The first part is the accumulative summation of log-intensity function for those events occurring at $\{(t_1, x_1, y_1), (t_2, x_2, y_2), \ldots, (t_n, x_n, y_n)\}$. The second part is the integral of intensity over space and time where no event happens. In our implementation, we set the spatio-temporal resolution to be 1 and adopt a discrete approximation of Equation 2.99:

$$\log f = \sum_j \log \lambda^*(t_j, x_j, y_j) - \sum_{(\tau,u,v) \neq (t_j, x_j, y_j)} \lambda^*(\tau, u, v) \tag{2.100}$$

This objective function is differentiable and can be efficiently optimized by stochastic gradient descent.

Experiments

Our experiments are performed on two benchmark datasets: MOT16 [144] and MOT17. n MOT Challenge Benchmark [107, 144], the relationship between ground truth and tracker output is established using bounding boxes. The intersection over union (IOU) is used as the similarity criterion, where the threshold is set to 0.5. Then, the tracking performance is measured by Multiple Object Tracking Accuracy (MOTA, the primary metric), Multiple Object Tracking Precision (MOTP), the total number of False Negatives (FN), the total number of False Positives (FP), the total number of Identity Switches (IDs), the percentage of Mostly Tracked Trajectories (MT) and the percentage of Mostly Lost Trajectories (ML).

Ablation study

We compare the baseline tracking algorithm to the proposed point process enhanced tracking algorithm. The experiments are performed on MOT2016 with different components of our approach. The results are shown in Table 2.22. To outline the superiority and generalization ability of the proposed method, we evaluated the point process model on other tracking methods [11, 204]. Figure 2.47 provides visuals of some tracking results by the baseline tracking method and the point process enhanced tracking method.

In this comparison, the following tracking algorithms are evaluated upon to demonstrate the generalization ability of our approach:

- **Baseline**. We apply k-means clustering algorithm to generate reliable tracklets followed by softassign [191] algorithm to associate the tracklets into trajectories. For simplicity,

Table 2.22 Ablation experiments on different components. All models are tested on MOT2016 dataset

Components	MOTA↑	MOTP↑	MT↑ (%)	ML↓ (%)
Baseline	38.2	77.7	16.1	49.3
+ Time Independent	39.6	77.6	16.6	49.7
+ Sync RNN	41.6	77.1	19.0	39.3
+ Sync/Async RNN	**44.1**	**82.7**	**23.8**	**36.0**
Deep-SORT [204]	28.4	**78.8**	4.8	63.1
+ Time Independent	29.1	78.5	5.8	59.4
+ Sync RNN	29.9	78.4	6.6	57.4
+ Sync/Async RNN	**30.4**	78.2	**7.2**	**55.1**
Tracktor [11]	42.4	78.2	17.3	40.4
+ Time Independent	42.9	78.0	17.7	39.9
+ Sync RNN	43.9	78.3	18.5	39.1
+ Sync/Async RNN	**44.5**	**78.5**	**19.2**	**38.6**

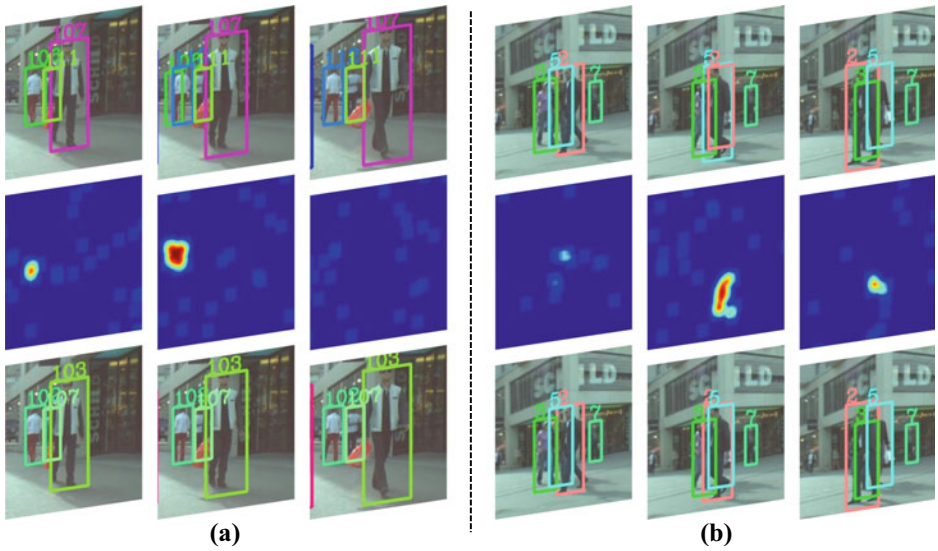


Fig. 2.47 Two tracking examples. First row: the results by baseline tracking algorithm. Second row: the intensity maps generated from the proposed point process model. Third row: the results by our proposed method. **a** An example of noisy detection. The noisy detection occurs in the second frame of the top row (green box), which causes two different trajectories for one object. **b** An example of confusing detection. The confusing detections occurs in the second frame of the top row (blue and red boxes), which causes a mismatch error (identity switch)

we use the terminal detection in each tracklet to calculate tracklet-wise similarity. The baseline algorithm includes a simple threshold-based strategy to filter out noisy detections with low confidences.

- **DeepSORT** [204]. The DeepSORT variant integrates appearance information to improve the performance of SORT. It learns a deep association metric on large-scale person re-identification dataset in the offline pre-training stage. Then, during online application, it establishes measurement-to-track associations using nearest neighbor queries in the visual appearance space.
- **Tracktor** [11]. Tracktor is a tracker without any training or optimization on tracking data. It exploits the bounding box regression of an object detector to predict the position of an object in the next frame. It has good extensibility and performs well on three multi-object tracking benchmarks by extending it with a straightforward re-identification and camera motion compensation.

The following components are evaluated to demonstrate the effectiveness of our approach:

- *Time Independent*. In this method, we only use the weight-shared CNN to handle the time series input. In other words, at each step, the current frame is input into a fully convolutional neural network (FCN) [129] to generate the intensity map for event prediction.
- *Synchronous RNN*. This method, which is the partial version of our proposed method, takes only the time series as input data in order to reflect the state of environment where events happen. Correspondingly, the synchronous RNN models the exogenous intensity in our point process.
- *Synchronous + Asynchronous RNN*. This is the full version of our proposed method. It includes both synchronous and asynchronous RNN, where the time series and associated event sequence are taken as input data. We adopt a neural network based time evolving mechanism to align and merge these two features, for the generation of the intensity maps.

From Table 2.22 and Fig. 2.47, we make the following observations:

(1) The time independent prediction method outperforms our baseline tracking algorithm. Tracking examples in Fig. 2.47 show that directly associating the detected objects or simply using general tracklet association algorithms is ineffective and easily interfered by noisy or confusing detections. Comparatively, by applying the time independent model to predict and discard the "bad" detections before association process, we are able to mitigate these noisy and confusing detections to a reasonable measure of success.

(2) The Synchronous RNN model has significantly better results compared to the time independent model. The performance gains come from the fact that a learnable model of larger capacity is employed to capture the historical information to cater for complex temporal dynamics. In addition, we observe that both the time independent model and synchronous RNN model have lower MOTP values than the baseline algorithm; MOTP measures the precision of size and location of bounding boxes in trajectories. This is likely because these two methods mask out the "bad" detections effectively, so that they can improve tracking accuracy such as MOTA. This occurs at the expense of losing some "good" detections to mis-classification and masking. These errors occur sparsely in time, and can be recovered in the tracklet clustering process using linear interpolation (a common scheme also adopted by our baseline tracking algorithm). The only caution being that, bounding boxes generated by linear interpolation are less precise than the detector results especially when the objects exhibit large or highly dramatic motions. Nevertheless, we observe that the overlaps between interpolated boxes and ground-truth boxes are still larger than 50%, which may lower the MOTP score, but will not affect the MOTA score.

(3) The full version of the proposed method (synchronous + asynchronous RNN) performs much better than all other compared methods. This demonstrates the importance of adopting event sequence in addition to time series as inputs. Intuitively, this helps the model to better capture long-term dependencies of events. From another aspect, the asynchronous RNN

Table 2.23 Tracking performances of our approach and state-of-the-art methods on MOT2016

Method	MOTA↑	MOTP↑	MT↑ (%)	ML↓ (%)	FP↓	FN↓	IDS↓
OICF [96]	43.2	74.3	11.3	48.5	6651	96515	381
CBDA [8]	43.9	74.7	10.7	44.4	6450	95175	676
QCNN [171]	44.1	76.4	14.6	44.9	6388	94775	745
STAM [31]	46.0	74.9	14.6	43.6	6895	91117	473
MDM [181]	46.3	75.7	15.5	39.7	6449	90713	663
NOMT [30]	46.4	76.6	18.3	41.4	9753	87565	**359**
JGD-NL [108]	47.6	78.5	17.0	40.4	**5844**	89093	629
TSN-CC [155]	48.2	75.0	**19.9**	**38.9**	8447	85315	665
LM-PR [182]	48.8	**79.0**	18.2	40.1	6654	86245	481
Ours	**50.5**	74.9	19.6	39.4	5939	**83694**	638

Table 2.24 Tracking performances of our approach and state-of-the-art methods on MOT2017

Method	MOTA↑	MOTP↑	MT↑ (%)	ML↓ (%)	FP↓	FN↓	IDS↓
EEBMM [138]	44.2	–	–	–	–	–	**1529**
NG-bL [98]	47.5	**77.5**	18.2	41.7	25981	268042	2069
OGSDL [59]	48.0	77.2	17.1	35.6	23199	265954	3998
DMAN [233]	48.2	75.9	19.3	38.3	26218	263608	2194
EDM [24]	50.0	77.3	21.6	36.3	32279	247297	2264
MHT [97]	50.7	**77.5**	20.8	36.9	22875	252889	2314
DLCS [128]	50.9	76.6	17.5	35.7	24069	250768	2474
CCC [92]	51.2	75.9	20.9	37.0	25937	247822	1802
FHFD [76]	51.3	77.0	21.4	**35.2**	24101	247921	2648
Ours	**52.4**	76.6	**22.4**	40.0	**20176**	**246158**	2224

also reflects the endogenous intensity, as a complementary part to the exogenous intensity already modeled by synchronous RNN.

(4) Our proposed method consistently improve all the three methods, including the baseline algorithm, the alternative Deep-SORT [204] tracking algorithm and the Tracktor [11] algorithm in all metrics, which demonstrates its effectiveness and strong generalization ability.

Comparison with State-of-the-art Methods

With the best settings of the proposed method affirmed in the ablation study, we conduct further comparisons against state-of-the-art MOT tracker on the widely used benchmark dataset: MOT2016 and MOT2017. For fair comparison, all the methods are evaluated and reported based on the same evaluation protocol and metrics.

Tables 2.23 and 2.24 list the benchmark results for MOT2016 and MOT2017 respectively, comparing the proposed method against recent state-of-the-art MOT trackers such as OICF [96], CBDA [8], QCNN [171], STAM [31], MDM [181], NOMT [30], JGD-NL [108], TSN-CC [155], LM-PR [182], EEBMM [138], NG-bL [98], OGSDL [59], DMAN [233], EDM [24], MHT [97], DLCS [128], CCC [92], and FHFD [76].

From the tracking performance comparisons in Tables 2.23 and 2.24, it can be seen that our tracker surpasses all other methods in terms of the primary evaluation metric MOTA. In MOT16, compared to the closest competitor LM-PR [182], our method achieves a 1.7% improvement (50.5% *vs.* 48.8%) in MOTA, a 1.4% improvement (19.6% *vs.* 18.2%) in MT and a 0.7% improvement (39.4% *vs.* 40.1%) in ML. In MOT17, compared to the closest competitor FHFD [76], our method achieves a 1.1% improvement (52.4% *vs.* 51.3%) in MOTA and a 1.0% improvement (22.4% *vs.* 21.4%) in MT. Note that both LM-PR [182] and FHFD [76] have more sophisticated and computationally heavy tracking pipelines than our baseline algorithm. This implies that we can make significant strides to improve the state-of-the-art by formulating it as a point process method. On the other hand, the MOTP of our approach is slightly lower than some methods because the interpolated detections tend to be less precise when the objects have some large or highly dramatic motions.

Our approach also produces the lowest FN on both MOT16/MOT17 and highest MT in MOT17 (second highest on MOT16), which shows that the proposed method can accurately associate the tracklets and the interpolation process is effectively in generating missing detections. Our approach produces the lowest FP on MOT17 (second lowest on MOT16), which demonstrates the strengths of our approach in handling noisy and confusing detections by formulating these "bad" detections using our spatio-temporal point process model.

Although some methods (e.g., CCC [92]) have better performances on the ML and IDS metrics, they obtain this at the expense of sacrificing the performance on other metrics (i.e., MT and FP), which leads to a low overall result in MOTA. Comparatively, our approach can comprehensively balance different aspects in tracking and obtain a better overall result in MOTA. Besides, our approach has relatively high values on some metrics (i.e., IDS) because we use simple features and association schemes to perform tracking. In fact, since our proposed point-process model is generic, we can easily incorporate into our model more sophisticated features and association schemes (e.g., aggregated local flow descriptor in NOMT [30]) to potentially obtain better performances on these metrics.

Conclusion

In this paper, we address the issue of mis-detections in Multiple Object Tracking (MOT) task by proposing a novel framework that effectively predicts and masks out noisy and confusing detection results prior to associating objects into trajectories. We formulate the "bad" detection results as a sequence of events, adopting the spatio-temporal point process to model such event sequences. A convolutional recurrent neural network (Conv-RNN) is introduced to instantiate the point process, where the temporal and spatial evolutions are well captured. Experimental results demonstrate notable performance improvement with the proposed method over state-of-the-art MOT algorithms across different metrics and benchmark datasets.

2.3 Multi-object Tracking and Segmentation

2.3.1 Introduction

In this section, we briefly introduce a new tracking sub-task, Multi-object tracking and segmentation (MOTs) in 2019, which requires detecting, tracking, and segmenting objects belonging to a set of given classes from the video. Voigtlaender et al. [190] firstly extended the popular task of multi-object tracking to multi-object tracking and segmentation (MOTS). They also created two MOTS datasets: KITTI MOTS and MOTS Challenge.

Specifically, MOTs first computes the spatial locations of the interested targets in each single frame technically (such as mask-RCNN [71]). Based on the temporal correspondence of individual targets among consecutive frames, MOTs establishes moving information connections by re-identification and achieves accurate trajectories tracking along spatial and temporal dimension.

MOTs is highly similarly to VIS, the core of MOTs lies in the object detection per-frame and the instance association across different frames.

In the rest of this section, we will first introduce popular benchmarks and evaluation metrics as overview of common and effective methods. Then we presents the representative MOTs methods in detail.

2.3.2 Dataset and Metric

This section will talk about popular public dataset as well as common evaluation metrics.

Considering the evaluation datasets origin from current MOT benchmarks, the main moving objects is the videos are pedestrians, vehicles while VIS datasets contain more categories and long-term videos. The detail information about each dataset is listed in Table 2.25.

Correspondingly, there are three evaluation metrics for the new task: MOTSA, MOTSP and sMOTSA.

Table 2.25 Comparison among existing datasets

Dataset	Number	Frames	Pedestrian	Car
KITTI MOTS (Train+val)	12+9	8,008	8,073+ 3,347	18,831+ 8,068
MOTSChallenge	4	2862	2,6894	–

MOTSA (multi-object tracking and segmentation accuracy) is computed as:

$$MOTSA = 1 - \frac{\|FN\| + \|FP\| + \|IDS\|}{\|M\|}, \tag{2.101}$$

where $\|FN\|$ denotes the false negative, $\|FP\|$ denotes the false positive while IDS denotes the set of id mis-tracked. M means the ground truth masks.

MOTSP (mask-based multi-object tracking and segmentation precision) is defined as:

$$MOTSP = \frac{\widetilde{TP}}{\|TP\|}. \tag{2.102}$$

where $\widetilde{TP} = \sum_{h \in TP} IOU(h, c(h))$, $c(h)$ is a mapping function.

sMOTSA (soft multi-object tracking and segmentation accuracy) is computed as:

$$sMOTSA = \frac{\widetilde{TP} - \|FP\| - \|IDS\|}{\|M\|}. \tag{2.103}$$

2.3.3 Overview of Methods

In this section, we introduce several representative MOTS methods based deep learning solution. The first one (i.e., the first MOTS dataset) is TrackR-CNN [190]. In this work, the authors extend Mask R-CNN by 3D convolutions to incorporate temporal context and an extra association head that produces association vectors for each detection. The Euclidean distances between association vectors are used to associate detected bounding boxes over time into tracks.

Then, Song et al. [174] proposed an enhanced TrackR-CNN that is based on the Gaussian mixture probability hypothesis density (GMPHD) filter, a hierarchical data association (HDA), and a mask-based affinity fusion (MAF) model to achieve high-performance online tracking. Subsequently, inspired by hard sample mining, Xu et al. [210] presented a novel data augmentation strategy named continuous copy-paste (CCP) by exploiting the pixel-wise annotations provided by MOTS to actively increase the number of instances as well as unique instance IDs in training.

Furthermore, to fully leverage rich spatio-temporal information among the video, Ke et al. [91] introduce a Prototypical Cross-Attention Network (PCAN) first distills a space-time memory into a set of prototypes and then employs cross-attention to retrieve rich information from the past frames.

UniTrack [201] build the first self-supervised unified method to address five different tasks (SOT, VOS, MOT, MOTS and PoseTrack) within the same framework. The framework can be divided in three stages. Stage-1: a trainable appearance model. In stage-2: the fundamental primitives of propagation and association. Stage-3: task-specific heads are provided. Subsequently, Yan et al. [211] present a supervised unified method, termed Unicorn, that can simultaneously solve four tracking problems (SOT, MOT, VOS, MOTS).

Finally, in [20], the authors formulate MOT and MOTS as a Message Passing procedure on the graph. In this way, a fully differentiable framework based on message passing networks can exploit the underlying graph structure of the tracking problem. The proposed method achieves state-of-the-art results in both multi-object tracking and segmentation tasks on several benchmarks.

Overall, for MOTS, fusing tracking and segmentation cues with a unified model is an under-explored research direction.

References

1. B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. 34(11):2189–2202, 2012.
2. Saad Ali and Mubarak Shah. Floor fields for tracking in high density crowd scenes. In *European conference on computer vision*, pages 1–14. Springer, 2008.
3. M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.
4. Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M Kitani. N2n learning: Network to network compression via policy gradient reinforcement learning. arXiv:1709.06030, 2017.
5. Shayan Modiri Assari, Haroon Idrees, and Mubarak Shah. Human re-identification in crowd videos using personal, social and environmental constraints. In *ECCV*, 2016.
6. Shai Avidan. Support vector tracking. *IEEE transactions on pattern analysis and machine intelligence*, 26(8):1064–1072, 2004.
7. Boris Babenko, Ming-Hsuan Yang, and Serge J. Belongie. Robust object tracking with online multiple instance learning. 33(8), 2011.
8. Seung-Hwan Bae and Kuk-Jin Yoon. Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):595–610, 2017.
9. Chenglong Bao, Yi Wu, Haibin Ling, and Hui Ji. Real time robust l1 tracker using accelerated proximal gradient approach. In *2012 ieee conference on computer vision and pattern recognition*, pages 1830–1837. IEEE, 2012.
10. Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE TPAMI*, 2011.

11. Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 941–951, 2019.
12. Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: a tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4413–4421, 2018.
13. Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.
14. Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip H. S. Torr. Staple: Complementary learners for real-time tracking. pages 1401–1409, 2016.
15. Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016.
16. Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.
17. Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. pages 493–509, 2018.
18. Samuel S Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, 2004.
19. David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. pages 2544–2550, 2010.
20. Guillem Braso, Orcun Cetintas, and Laura Leal-Taixe. Multi-object tracking and segmentation via neural message passing. arXiv:2207.07454, 2022.
21. Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6247–6257, 2020.
22. Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.
23. Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. *Advances in neural information processing systems*, 30, 2017.
24. Jiahui Chen, Hao Sheng, Yang Zhang, and Zhang Xiong. Enhancing detection model for multiple hypothesis tracking. In *CVPRW*, 2017.
25. Long Chen, Haizhou Ai, Zijie Zhuang, and Chong Shang. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In *2018 IEEE international conference on multimedia and expo (ICME)*, pages 1–6. IEEE, 2018.
26. Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8126–8135, 2021.
27. Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6668–6677, 2020.
28. Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. Deep meta learning for real-time target-aware visual tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 911–920, 2019.
29. Jongwon Choi, Hyung Jin Chang, Tobias Fischer, Sangdoo Yun, Kyuewang Lee, Jiyeoup Jeong, Yiannis Demiris, and Jin Young Choi. Context-aware deep feature compression for high-speed

visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 479–488, 2018.
30. Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *ICCV*, 2015.
31. Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In *ICCV*, 2017.
32. Robert T Collins, Yanxi Liu, and Marius Leordeanu. Online selection of discriminative tracking features. *IEEE transactions on pattern analysis and machine intelligence*, 27(10):1631–1643, 2005.
33. J. B. Copas. Regression, prediction and shrinkage. *Journal of the Royal Statistical Society*, 45, 1983.
34. Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6638–6646, 2017.
35. Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. Bmva Press, 2014.
36. Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Convolutional features for correlation filter based visual tracking. In *ICCV Workshops*, 2015.
37. Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Discriminative scale space tracking. *IEEE transactions on pattern analysis and machine intelligence*, 39(8):1561–1575, 2016.
38. Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Convolutional features for correlation filter based visual tracking. In *ICCV Workshops*, pages 58–66, 2015.
39. Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 4310–4318, 2015.
40. Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost van de Weijer. Adaptive color attributes for real-time visual tracking. pages 1090–1097, 2014.
41. Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. pages 472–488, 2016.
42. Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1532–1545, 2014.
43. Xingping Dong and Jianbing Shen. Triplet loss in siamese network for object tracking. In *European Conference on Computer Vision*, pages 459–474, 2018.
44. Xingping Dong, Jianbing Shen, Wenguan Wang, Yu Liu, Ling Shao, and Fatih Porikli. Hyperparameter optimization for tracking with continuous deep q-learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 518–527, 2018.
45. Xingping Dong, Jianbing Shen, Wenguan Wang, Ling Shao, Haibin Ling, and Fatih Porikli. Dynamical hyperparameter optimization via deep reinforcement learning in tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
46. Kai Du, Yongfeng Ju, Yinli Jin, Gang Li, Yanyan Li, and Shenglong Qian. Object tracking based on improved meanshift and sift. In *2012 2nd International conference on consumer electronics, communications and networks (CECNet)*, pages 2716–2719. IEEE, 2012.
47. Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In

Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1555–1564. ACM, 2016.

48. A. Ess, B. Leibe, K. Schindler, , and L. van Gool. A mobile vision system for robust multi-person tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. IEEE Press, June 2008.
49. Matej Kristan et al. The visual object tracking VOT2015 challenge results. In *ICCV Workshops*, pages 564–586, 2015.
50. David Exner, Erich Bruns, Daniel Kurz, Anselm Grundhöfer, and Oliver Bimber. Fast and robust camshift tracking. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 9–16. IEEE, 2010.
51. Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5374–5383, 2019.
52. Heng Fan and Haibin Ling. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. pages 5486–5494, 2017.
53. Heng Fan and Haibin Ling. Sanet: Structure-aware network for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 42–49, 2017.
54. Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7952–7961, 2019.
55. Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. 2019.
56. Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.
57. Weitao Feng, Zhihao Hu, Wei Wu, Junjie Yan, and Wanli Ouyang. Multi-object tracking with multiple cues and switcher-aware classification. arXiv:1901.06129 2019.
58. Thomas Fortmann, Yaakov Bar-Shalom, and Molly Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE journal of Oceanic Engineering*, 8(3):173–184, 1983.
59. Zeyu Fu, Pengming Feng, Federico Angelini, Jonathon Chambers, and Syed Mohsen Naqvi. Particle phd filter based multiple human tracking using online group-structured dictionary learning. *IEEE Access*, 2018.
60. Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. arXiv:1703.05884, 2017.
61. Jin Gao, Haibin Ling, Weiming Hu, and Junliang Xing. Transfer learning based visual tracking with gaussian processes regression. pages 188–203. 2014.
62. Junyu Gao, Tianzhu Zhang, and Changsheng Xu. Graph convolutional tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4649–4659, 2019.
63. Junyu Gao, Tianzhu Zhang, Xiaoshan Yang, and Changsheng Xu. P2t: Part-to-target tracking via deep regression learning. *IEEE Transactions on Image Process.*, 27(6):3074–3086, 2018.
64. Shan Gao, Xiaogang Chen, Qixiang Ye, Junliang Xing, Arjan Kuijper, and Xiangyang Ji. Beyond group: Multiple person tracking via minimal topology-energy-variation. *IEEE TIP*, 2017.
65. Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

66. Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *European conference on computer vision*, pages 234–247. Springer, 2008.
67. Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. Learning dynamic siamese network for visual object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1763–1771, 2017.
68. Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2096–2109, 2015.
69. Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. Towards a better match in siamese network based visual object tracker. pages 132–147, September 2018.
70. Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4834–4843, 2018.
71. Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.
72. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
73. David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. pages 749–765, 2016.
74. João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596, 2014.
75. João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge P. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. pages 702–715, 2012.
76. Roberto Henschel, Laura Leal-Taixe, Daniel Cremers, and Bodo Rosenhahn. Fusion of head and full-body detectors for multi-object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1428–1437, 2018.
77. Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. arXiv:1503.02531, 2(7), 2015.
78. Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. Online tracking by learning discriminative saliency map with convolutional neural network. pages 597–606, 2015.
79. Zhibin Hong, Zhe Chen, Chaohui Wang, and Xue Mei. Multi-store tracker (muster): a cognitive psychology inspired approach to object tracking. pages 749–758, 2015.
80. Zhibin Hong, Zhe Chen, Chaohui Wang, Xue Mei, Danil V. Prokhorov, and Dacheng Tao. Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. pages 749–758, 2015.
81. Yang Hua, Karteek Alahari, and Cordelia Schmid. Online object tracking with proposal selection. In *Proceedings of the IEEE international conference on computer vision*, pages 3092–3100, 2015.
82. Dafei Huang, Lei Luo, Mei Wen, Zhaoyun Chen, and Chunyuan Zhang. Enable scale and aspect ratio adaptability in visual tracking with detection proposals. pages 185.1–185.12, 2015.
83. Lianghua Huang, Xin Zhao, and Kaiqi Huang. Bridging the gap between detection and tracking: A unified approach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3999–4009, 2019.
84. Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1562–1577, 2021.
85. Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017.

86. Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. 2014.
87. Xiaolong Jiang, Peizhao Li, Yanjing Li, and Xiantong Zhen. Graph neural based end-to-end data association framework for online multiple-object tracking. arXiv:1907.05315, 2019.
88. Ilchae Jung, Jeany Son, Mooyeol Baek, and Bohyung Han. Real-time mdnet. In *European Conference on Computer Vision*, pages 83–98, 2018.
89. Briechle Kai and Uwe D. Hanebeck. Template matching using fast normalized cross correlation. In *Aerospace/Defense Sensing, Simulation, and Controls*, pages 95–102, 2001.
90. Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2011.
91. Lei Ke, Xia Li, Martin Danelljan, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Prototypical cross-attention networks for multiple object tracking and segmentation. *Advances in Neural Information Processing Systems*, 34:1192–1203, 2021.
92. Margret Keuper, Siyu Tang, Bjorn Andres, Thomas Brox, and Bernt Schiele. Motion segmentation & multiple object tracking by correlation co-clustering. *IEEE TPAMI*, 2018.
93. Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *CVPR*, 2017.
94. Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey. Learning background-aware correlation filters for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 1135–1143, 2017.
95. Hamed Kiani Galoogahi, Terence Sim, and Simon Lucey. Correlation filters with limited boundaries. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4630–4638, 2015.
96. Hilke Kieritz, Stefan Becker, Wolfgang Hübner, and Michael Arens. Online multi-person tracking using integral channel features. In *AVSS*, 2016.
97. Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *ICCV*, 2015.
98. Chanho Kim, Fuxin Li, and James M Rehg. Multi-object tracking with neural gating using bilinear lstm. In *ECCV*, 2018.
99. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. pages 749–765, 2015.
100. Matej Kristan, Ales Leonardis, Jiri Matas, and et al. The visual object tracking VOT2016 challenge results. In *ECCV Workshops*, 2016.
101. Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, and Pflugfelder. *The Sixth Visual Object Tracking VOT2018 Challenge Results*, pages 3–53. 01 2019.
102. Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kamarainen, Luka Čehovin Zajc, Ondrej Drbohlav, Alan Lukezic, Amanda Berg, et al. The seventh visual object tracking vot2019 challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.
103. Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, 2016.
104. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.
105. Harold W Kuhn. The hungarian method for the assignment problem. *NRL*, 1955.
106. Matjaz Kukar and Igor Kononenko. Cost-sensitive learning with neural networks. In *ECAI*, pages 445–449, 1998.

107. Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. arXiv:1504.01942, 2015.
108. Evgeny Levinkov, Jonas Uhrig, Siyu Tang, Mohamed Omran, Eldar Insafutdinov, Alexander Kirillov, Carsten Rother, Thomas Brox, Bernt Schiele, and Bjoern Andres. Joint graph decomposition & node labeling: Problem, algorithms, applications. In *CVPR*, 2017.
109. Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019.
110. Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. pages 8971–8980, 2018.
111. Feng Li, Cheng Tian, Wangmeng Zuo, Lei Zhang, and Ming Hsuan Yang. Learning spatial-temporal regularized correlation filters for visual tracking. pages 4904–4913, 2018.
112. Hanxi Li, Yi Li, and Fatih Porikli. Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In *British Machine Vision Conference*, pages 1420–1429, 2014.
113. Jiahe Li, Xu Gao, and Tingting Jiang. Graph networks for multiple object tracking. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 719–728, 2020.
114. Peixia Li, Boyu Chen, Wanli Ouyang, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Gradnet: Gradient-guided network for visual object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6162–6171, 2019.
115. Shuang Li, Slawomir Bak, Peter Carr, and Xiaogang Wang. Diversity regularized spatiotemporal attention for video-based person re-identification. In *CVPR*, 2018.
116. Siyi Li and Dit-Yan Yeung. Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models. In *Thirty-first AAAI conference on artificial intelligence*, 2017.
117. Xi Li, Anthony Dick, Hanzi Wang, Chunhua Shen, and Anton van den Hengel. Graph mode-based contextual kernels for robust svm tracking. In *2011 international conference on computer vision*, pages 1156–1163, 2011.
118. Xin Li, Chao Ma, Baoyuan Wu, Zhenyu He, and Ming-Hsuan Yang. Target-aware deep tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1369–1378, 2019.
119. Yang Li and Jianke Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *European conference on computer vision*, pages 254–265. Springer, 2014.
120. Pengpeng Liang, Erik Blasch, and Haibin Ling. Encoding color information for visual tracking: Algorithms and benchmark. 24(12), 2015.
121. Wang Lijun, Ouyang Wanli, Wang Xiaogang, and Lu Huchuan. STCT: sequentially training convolutional networks for visual tracking. pages 1373–1381, 2016.
122. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Protr Dollr. Focal loss for dense object detection. pages 2999–3007, 2017.
123. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.
124. Weiyao Lin, Huabin Liu, Shizhan Liu, Yuxi Li, Rui Qian, Tao Wang, Ning Xu, Hongkai Xiong, Guo-Jun Qi, and Nicu Sebe. Human in events: A large-scale benchmark for human-centric video analysis in complex events. arXiv:2005.04490, 2020.
125. Meng Liu, Chengdong Wu, and Yunzhou Zhang. Motion vehicle tracking based on multi-resolution optical flow and multi-scale harris corner detection. In *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2032–2036. IEEE, 2007.

126. Qiao Liu, Xin Li, Zhenyu He, Chenglong Li, Jun Li, Zikun Zhou, Di Yuan, Jing Li, Kai Yang, Nana Fan, et al. Lsotb-tir: A large-scale high-diversity thermal infrared object tracking benchmark. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 3847–3856, 2020.
127. Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *Proc. ACM Int. Conf. Mach. Learn.*, volume 2, page 7, 2016.
128. C Long, A Haizhou, Z Zijie, and S Chong. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In *ICME*, 2018.
129. Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
130. Xiankai Lu, Chao Ma, Bingbing Ni, Xiaokang Yang, Ian Reid, and Ming-Hsuan Yang. Deep regression tracking with shrinkage loss. pages 369–386, 2018.
131. Xiankai Lu, Wenguan Wang, Chao Ma, Jianbing Shen, Ling Shao, and Fatih Porikli. See more, know more: Unsupervised video object segmentation with co-attention siamese networks. In *CVPR*, 2019.
132. Alan Lukezic, Tomás Vojír, Luka Cehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter tracker with channel and spatial reliability. 126(7):671–688, 2018.
133. Alan Lukežič, Luka Čehovin Zajc, and Matej Kristan. Deformable parts correlation filters for robust visual tracking. *IEEE transactions on cybernetics*, 48(6):1849–1861, 2018.
134. Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 3074–3082, 2015.
135. Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Adaptive correlation filters with long-term and short-term memory for object tracking. *IJCV*, pages 1–26, 2018.
136. Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Robust visual tracking via hierarchical convolutional features. 2018.
137. Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. pages 5388–5396, 2015.
138. Andrii Maksai and Pascal Fua. Eliminating exposure bias and metric mismatch in multiple object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4639–4648, 2019.
139. Christoph Mayer, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. Learning target candidate association to keep track of what not to track. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13444–13454, 2021.
140. Niall McLaughlin, Jesus Martinez del Rincon, and Paul Miller. Recurrent convolutional network for video-based person re-identification. In *CVPR*, 2016.
141. Xue Mei and Haibin Ling. Robust visual tracking and vehicle classification via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2259–2272, 2011.
142. Xue Mei, Haibin Ling, Yi Wu, Erik Blasch, and Li Bai. Minimum error bounded efficient ℓ_1 tracker with occlusion detection. In *CVPR 2011*, pages 1257–1264. IEEE, 2011.
143. Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8844–8854, 2022.
144. Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. arXiv:1603.00831, 2016.
145. Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. pages 445–461. Springer, 2016.
146. Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for UAV tracking. pages 445–461, 2016.

147. Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European conference on computer vision (ECCV)*, pages 300–317, 2018.
148. Hyeonseob Nam, Mooyeol Baek, and Bohyung Han. Modeling and propagating cnns in a tree structure for visual tracking. arXiv:1608.07242, 2016.
149. Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4293–4302, 2016.
150. Hyeonseob Nam, Seunghoon Hong, and Bohyung Han. Online graph-based tracking. pages 112–126, 2014.
151. Georg Nebehay and Roman P. Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *IEEE Winter Conference on Applications of Computer Vision*, pages 862–869, 2014.
152. Jifeng Ning, Jimei Yang, Shaojie Jiang, Lei Zhang, and Ming-Hsuan Yang. Object tracking via dual linear structured SVM and explicit feature map. pages 4266–4274, 2016.
153. Hitesh A Patel and Darshak G Thakore. Moving object tracking using kalman filter. *International Journal of Computer Science and Mobile Computing*, 2(4):326–332, 2013.
154. Luis Patino, Tom Cane, Alain Vallee, and James Ferryman. Pets 2016: Dataset and challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2016.
155. Jinlong Peng, Fan Qiu, John See, Qi Guo, Shaoshuai Huang, Ling-Yu Duan, and Weiyao Lin. Tracklet siamese network with constrained clustering for multiple object tracking. In *2018 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2018.
156. Horst Possegger, Thomas Mauthner, and Horst Bischof. In defense of color-based model-free tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2113–2120, 2015.
157. Yuankai Qi, Shengping Zhang, Feng Jiang, Huiyu Zhou, Dacheng Tao, and Xuelong Li. Siamese local and global networks for robust face tracking. *IEEE Transactions on Image Processing*, 29:9152–9164, 2020.
158. Yuankai Qi, Shengping Zhang, Lei Qin, Hongxun Yao, Qingming Huang, Jongwoo Lim, and Ming-Hsuan Yang. Hedged deep tracking. pages 4303–4311, 2016.
159. Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5296–5305, 2017.
160. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. Advances Neural Inf. Process. Syst.*, pages 91–99, 2015.
161. Gonzalo R Rodríguez-Canosa, Stephen Thomas, Jaime Del Cerro, Antonio Barrientos, and Bruce MacDonald. A real-time method to detect and track moving objects (datmo) from unmanned aerial vehicles (uavs) using a single camera. *Remote Sensing*, 4(4):1090–1111, 2012.
162. Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. arXiv:1412.6550, 2014.
163. David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. 77(1-3):125–141, 2008.
164. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

165. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. 115(3):211–252, 2015.
166. Samuele Salti, Andrea Cavallaro, and Luigi di Stefano. Adaptive appearance modeling for video tracking: Survey and evaluation. *IEEE Trans. Image Processing*, 21(10):4334–4348.
167. Xingjian Shi, Zhourong Chen, Hao Wang, Dit Yan Yeung, Wai Kin Wong, and Wang Chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, 2015.
168. Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016.
169. Guang Shu, Afshin Dehghan, Omar Oreifej, Emily Hand, and Mubarak Shah. Part-based multiple-person tracking with partial occlusion handling. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1815–1821. IEEE, 2012.
170. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. pages 744–752, 2015.
171. Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. Multi-object tracking with quadruplet convolutional neural networks. In *CVPR*, 2017.
172. Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson WH Lau, and Ming-Hsuan Yang. Crest: Convolutional residual learning for visual tracking. pages 2574–2583, 2017.
173. Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Lau Rynson, and Ming-Hsuan Yang. Vital: Visual tracking via adversarial learning. pages 8990–8999, 2018.
174. Young-min Song and Moongu Jeon. Online multi-object tracking and segmentation with gmphd filter and simple affinity fusion. arXiv:2009.00100, 3, 2020.
175. Zikai Song, Junqing Yu, Yi-Ping Phoebe Chen, and Wei Yang. Transformer tracking with cyclic shifting window attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8791–8800, June 2022.
176. Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Learning spatial-aware regressions for visual tracking. pages 8962–8970, June 2018.
177. Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple object tracking with transformer. arXiv:2012.15460, 2020.
178. James S Supancic and Deva Ramanan. Self-paced learning for long-term tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2379–2386, 2013.
179. Feng Tang and Qiang Ling. Ranking-based siamese visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8741–8750, June 2022.
180. Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Subgraph decomposition for multi-target tracking. In *CVPR*, 2015.
181. Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Multi-person tracking by multicut and deep matching. In *ECCVW*, 2016.
182. Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *CVPR*, 2017.
183. Ran Tao, Efstratios Gavves, and Arnold W. M. Smeulders. Siamese instance search for tracking. pages 1420–1429, 2016.
184. Min Tian, Weiwei Zhang, and Fuqiang Liu. On-line ensemble svm for robust object tracking. In *Asian conference on computer vision*, pages 355–364. Springer, 2007.

185. Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.
186. Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
187. Jack Valmadre, Luca Bertinetto, João Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. pages 2805–2813, 2017.
188. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
189. Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for MATLAB. In *Proceedings of the Annual Conference on Multimedia Conference,*, pages 689–692, 2015.
190. Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-object tracking and segmentation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 7942–7951, 2019.
191. Bing Wang, Li Wang, Bing Shuai, Zhen Zuo, Ting Liu, Kap Luk Chan, and Gang Wang. Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association. In *CVPRW*, 2016.
192. Guangting Wang, Chong Luo, Xiaoyan Sun, Zhiwei Xiong, and Wenjun Zeng. Tracking by instance detection: A meta-learning approach. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6288–6297, 2020.
193. John Y. A. Wang and Edward H. Adelson. Layered representation for motion analysis. In *Conference on Computer Vision and Pattern Recognition, CVPR 1993, 15-17 June, 1993, New York, NY, USA*, pages 361–366. IEEE, 1993.
194. Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 3119–3127, 2015.
195. Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. pages 20–36, 2016.
196. Mengmeng Wang, Yong Liu, and Zeyi Huang. Large margin object tracking with circulant feature maps. pages 4800–4808, 2017.
197. Naiyan Wang, Siyi Li, Abhinav Gupta, and Dit-Yan Yeung. Transferring rich feature hierarchies for robust visual tracking. arXiv:1501.04587, 2015.
198. Naiyan Wang, Jianping Shi, Dit-Yan Yeung, and Jiaya Jia. Understanding and diagnosing visual tracking systems. In *Proceedings of the IEEE international conference on computer vision*, pages 3101–3109, 2015.
199. Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1571–1580, 2021.
200. Yongxin Wang, Kris Kitani, and Xinshuo Weng. Joint object detection and multi-object tracking with graph neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13708–13715. IEEE, 2021.
201. Zhongdao Wang, Hengshuang Zhao, Ya-Li Li, Shengjin Wang, Philip Torr, and Luca Bertinetto. Do different tracking tasks require different appearance models? *Advances in Neural Information Processing Systems*, 34:726–738, 2021.
202. Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In *European Conference on Computer Vision*, pages 107–122. Springer, 2020.

203. Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.
204. Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.
205. John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):210–227, 2008.
206. Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.
207. Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 37(09):1834–1848, 2015.
208. Tianyang Xu, Zhen-Hua Feng, Xiao-Jun Wu, and Josef Kittler. Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual object tracking. *IEEE Transactions on Image Processing*, 28(11):5596–5609, 2019.
209. Yihong Xu, Yutong Ban, Guillaume Delorme, Chuang Gan, Daniela Rus, and Xavier Alameda-Pineda. Transcenter: Transformers with dense queries for multiple-object tracking. arXiv:2103.15145, 2021.
210. Zhenbo Xu, Ajin Meng, Zhenbo Shi, Wei Yang, Zhi Chen, and Liusheng Huang. Continuous copy-paste for one-stage multi-object tracking and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15323–15332, 2021.
211. Bin Yan, Yi Jiang, Peize Sun, Dong Wang, Zehuan Yuan, Ping Luo, and Huchuan Lu. Towards grand unification of object tracking. arXiv:2207.07078, 2022.
212. Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10448–10457, 2021.
213. Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *CVPR*, 2016.
214. Min Yang, Yuwei Wu, and Yunde Jia. A hybrid data association framework for robust online multi-object tracking. *IEEE TIP*, 2017.
215. Tianyu Yang, Pengfei Xu, Runbo Hu, Hua Chai, and Antoni B Chan. Roam: Recurrently optimizing tracking model. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6718–6727, 2020.
216. Fengwei Yu, Wenbo Li, Quanquan Li, Yu Liu, Xiaohua Shi, and Junjie Yan. Poi: multiple object tracking with high performance detection and appearance feature. In *ECCV*, 2016.
217. Jianming Zhang, Shugao Ma, and Stan Sclaroff. Meem: robust tracking via multiple experts using entropy minimization. In *European conference on computer vision*, pages 188–203. Springer, 2014.
218. Kaihua Zhang, Lei Zhang, Qingshan Liu, David Zhang, and Ming-Hsuan Yang. Fast visual tracking via dense spatio-temporal context learning. In *European conference on computer vision*, pages 127–141. Springer, 2014.
219. Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time compressive tracking. pages 864–877, 2012.
220. Lichao Zhang, Abel Gonzalez-Garcia, Joost van de Weijer, Martin Danelljan, and Fahad Shahbaz Khan. Learning the model update for siamese trackers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4010–4019, 2019.
221. Pengyu Zhang, Jie Zhao, Dong Wang, Huchuan Lu, and Xiang Ruan. Visible-thermal uav tracking: A large-scale benchmark and new baseline. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8886–8895, June 2022.

222. Shun Zhang, Jia-Bin Huang, Jongwoo Lim, Yihong Gong, Jinjun Wang, Narendra Ahuja, and Ming-Hsuan Yang. Tracking persons-of-interest via unsupervised representation adaptation. *International Journal of Computer Vision*, 128(1):96–120, 2020.
223. Tianzhu Zhang, Adel Bibi, and Bernard Ghanem. In defense of sparse tracking: Circulant sparse tracker. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3880–3888, 2016.
224. Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja. Robust visual tracking via structured multi-task sparse learning. *International journal of computer vision*, 101(2):367–383, 2013.
225. Tianzhu Zhang, Si Liu, Narendra Ahuja, Ming-Hsuan Yang, and Bernard Ghanem. Robust visual tracking via consistent low-rank sparse learning. *International Journal of Computer Vision*, 111(2):171–190, 2015.
226. Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129(11):3069–3087, 2021.
227. Yunhua Zhang, Lijun Wang, Jinqing Qi, Dong Wang, Mengyang Feng, and Huchuan Lu. Structured siamese network for real-time visual tracking. In *European Conference on Computer Vision*, pages 351–366, 2018.
228. Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. In *European Conference on Computer Vision*, pages 771–787. Springer, 2020.
229. Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020.
230. Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. arXiv:1904.07850, 2019.
231. Zikun Zhou, Jianqiu Chen, Wenjie Pei, Kaige Mao, Hongpeng Wang, and Zhenyu He. Global tracking via ensemble of local trackers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8761–8770, June 2022.
232. Gao Zhu, Fatih Porikli, and Hongdong Li. Beyond local search: Tracking objects everywhere with instance-specific proposals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 943–951, 2016.
233. Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 366–382, 2018.
234. Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. pages 101–117, 2018.
235. C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. pages 391–405. 2014.

HiEve Challenge on VOT

3

3.1 Introduction

Along with the development of modern smart cities, human-centric video analysis has been encountering the challenge of analyzing diverse and complex events in real scenes. A complex event relates to dense crowds, anomalous individuals, or collective behaviors. However, limited by the scale and coverage of existing video datasets, few human analysis approaches have reported their performances on such complex events. To advance the development of cutting-edge techniques in human-centric analysis and the understanding of complex events, we present a new large-scale dataset with comprehensive annotations, named HiEve (Human-centric video analysis in complex Events), for the understanding of human motions, poses, and actions in a variety of realistic events, especially in crowd and complex events.

Based on the proposed dataset, we successfully held challenge and corresponding workshop in conjunction with ACM Multimedia 2020. And our grand challenge won the best Grand Challenge organization award in ACM Multimedia 2020. Moreover, our dataset and challenge system has been re-opened to the public as a long-term challenge after ACM Multimedia challenge to welcome researchers to use our dataset to evaluate their research works.

Although our proposed dataset HiEve is a multi-task dataset for multi-object tracking (MOT), multi-person pose estimation, pose tracking and action recognition. In this book, we will mainly focuses on MOT task. Please visit homepage if interested in other tasks of our dataset and challenge.

In the rest of this section, we will first introduce our proposed dataset HiEve Sect. 3.2, then we will present the MOT track of our held challenge as well as the winners' methods in detail Sect. 3.2.

N. Xu et al., *Video Object Tracking*, Synthesis Lectures on Computer Vision,
https://doi.org/10.1007/978-3-031-44660-3_3

3.2 HiEve Dataset and Benchmark

3.2.1 Motivation

Before our work, there have been corpus of datasets that provide multi-object bounding-box and track annotations in video sequences, which have fostered the development of MOT task. PETS [3] is an early proposed multi-sensor video dataset, it includes annotation of crowd person count and tracking of an individual within a crowd. Its sequences are all shot in the same scene, which leads to relatively simple samples. KITTI [4] tracking dataset features videos from a vehicle-mounted camera and focuses on street scenarios, it owns 2D and 3D bounding-boxes and tracklets annotations. Meanwhile, is has a limited variety of video angles. The most popular benchmark to data for the evaluation of tracking is the MOT-Challenge [2], which shows pedestrians from a variety of different viewpoints. However, with the rapid development of MOT algorithms, the limited scenarios and human actions of MOT-Challenge dataset make it less suitable for a comprehensive and general benchmark on tracking performance of each method on complex scenarios of the real world. To overcome these limitation, we collect a new large-scale video dataset–HiEve [5] under various realistic complex events.

3.2.2 Video and Annotation

Video

Our HiEve dataset is composed of 32 real-world videos. They are collected from 9 different scenes: *airport, dining hall, indoor, jail, mall, square, school, station and street* , and contain one or more complex events. For each scene, we keep several videos captured at different sites and with different types of events happening to ensure the diversity of scenarios. Moreover, data redundancy is avoided through manual checking. Finally, these video sequences are split into training and testing sets of 19 and 13 videos so that both sets cover all the scenes but with different camera angles or sites.

Annotation

We manually give the human identities, bounding-boxes and other annotations for each video. Our annotations are double-checked to ensure their quality. Specifically, there are two groups of humans to conduct data annotation. All videos are first sent to one group for the 1st round labelling following the standard annotation procedure. After the 1st round annotation, the labelled data are then sent to another group for double-checking. During the double-check, annotations of each sample will be evaluated with a confidence score (value from 0 to 10), which indicates the confidence of labelling. Then, data with less than 9 confidence scores will be sent back to the forehead group for the 2nd round annotation. We repeat the above process until all annotations satisfy the rule of confidence score. Some video clips with annotation are shown in Fig. 3.2.

3.2.3 Statistic

Figure 3.1 shows the frame number of different scenes in HiEve. We also compare HiEve to some previous MOT datasets in Table 3.1. Compare to MOT series, our dataset is competitive with them in scale. Moreover, our dataset contains more complex events and actions, which is not included by MOT.

3.2.4 Evaluation and Metrics

Following previous work, some traditional metrics–MOTA, MOTP [6], ID F1 Score, ID Sw [8], and ID Sw-DT are selected to perform evaluation.

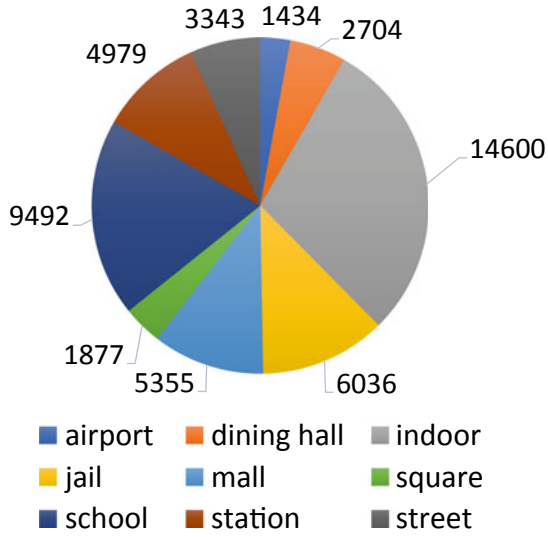


Fig. 3.1 The frame number distribution of different scenes in HiEve dataset

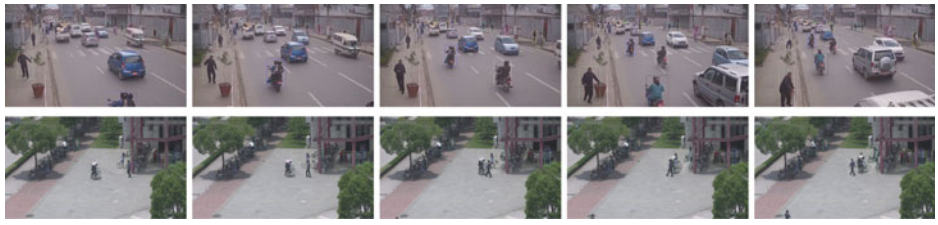


Fig. 3.2 The ground truth bounding-boxes of sample video clips in HiEve. The bounding-boxes with the same color among frames belongs to the same person

Table 3.1 Comparison between HiEve and existing datasets. "∼" denotes approximated value. "traj." means trajectory and "avg" indicates average trajectory length

Dataset	Year	Box	Traj. (avg)	Total length	Complex events
PoseTrack [1]	2018	∼26,000	5,245(49)	2,750s	×
MOT16 [6]	2016	292,733	1,276(229)	463s	×
MOT17	2017	901,119	3,993(226)	1,389s	×
MOT20 [2]	2020	1,652,040	3457(478)	535s	×
HiEve	2021	1,302,481	2,687(485)	1,839s	√

Besides, in order to evaluate how algorithms perform on tracks with disconnected parts, we design a weighted MOTA (w-MOTA) metric. This metric is computed in a similar manner as MOTA except that we assign a higher weight γ to the ID switch cases happening in disconnected tracks, consequently the metric can be formulated as

$$\textbf{w-MOTA} = 1 - (N_{fp} + N_{fn} + N_{sw} + (\gamma - 1)N_{sw-dt})/N_{gt}$$

where N_{fp} and N_{fn} are the number of false positive and false negative, N_{sw} is the total times of ID switch, N_{sw-dt} is the ID switch times happening in disconnected tracks and N_{gt} is the number of bounding boxes in annotations.

3.3 MOT Track of HiEve Challenge

In this section, we mainly present the result of MOT track of our held challenge–Large-scale Human-centric Video Analysis in Complex Events (HiEve). Note that we further divide MOT track into two sub-track:

- Private Detection: Participants in this sub-track is able to generate their own detection bounding-boxes through any detectors.
- Public Detection: In this sub-track, all competitors can only use the public object detection results provided by us, which is obtained via Faster-RCNN. This sub-track only focus on the tracking performance of methods.

The quantitative results of winner team for two sub-tracks are shown in Table 3.2. We omit some anonymous teams. We then give a brief introduction to winner teams and their methods.

Table 3.2 Quantitative results of top-performing methods on the MOT track of the HiEve challenge

Method	MOTA	w-MOTA	HOTA	MOTP	IDF1	MT	ML	FP	FN	IDSw	IDSw-DT
Private detection											
Adaptive FairMOT [10]	60.2282	54.6567	60.0026	41.2382	20.2518	4530	21292	1889	97	1808	74.9261
NewTracker [9]	46.4815	41.4269	43.2299	26.3379	30.8499	4667	30489	2133	88	1981	76.8703
Public detection											
LinkBox [7]	51.3828	46.5580	47.1540	29.2760	29.0661	2804	29345	1725	84	1645	76.4136
Selective JDE [10]	50.5863	45.4169	56.8138	25.0787	30.3253	2860	29850	1719	90	2993	76.7882
FCS-Track [9]	47.8048	42.5205	49.8188	25.2886	30.2204	3847	30862	1658	92	2625	76.8539

3.3.1 Private Detection

Adaptive FairMOT is a team from Sun Yat-sen University and ACCUVISION Technology consisting of Ancong Wu, Chengzhi Lin, Bogao Chen, Weihao Huang, Zeyu Huang and Wei-Shi Zheng. Their proposed method 'Transductive Multi-Object Tracking in Complex Events by Interactive Self-Training' won the 1st title on *Private Detection* sub-track. In this work, they mainly concerned the domain gap in unseen testing scenes and severe occlusion problem cause disconnected tracklets. To alleviate the affect of domain gap, they studied the problem in a transductive learning setting, which assumes that unlabeled testing data is available for learning offline tracking. They proposed a transductive interactive self-training method to adapt the tracking model to unseen crowded scenes with unlabeled testing data by teacher-student interactive learning. To reduce prediction variance in the unseen testing domain, they trained two different models and teach one model with pseudo labels of unlabeled testing data predicted by the other model interactively. To increase the robustness against occlusions during self-training, they exploited disconnected track interpolation (DTI) to refine the predicted pseudo labels. One weakness of their method is that the self-training during testing phase need lots of time.

NewTracker is a team from Amazon consisting of Bing Shuai, Andrew Berneshawi, Manchen Wang, Chunhui Liu, Davide Modolo, Xinyu Li and Joseph Tighe. In their work, they proposes a two stage detect-and-track framework–Siamese Track-RCNN, which aims at unifying different MOT modules (i.e., detector, tracker, re-identification feature extractor and a solver that combine the output from these separate components to make the final prediction.) into a single tracking system. Their proposed Siamese Track-RCNN is composed of three functional branches: (1) the detection branch to detect object instances; (2) the Siamese-

based track branch to estimate the motion and (3) the object re-identification branch to re-activate the previously terminated tracks when they re-emerge. Their method won the 2nd place on *Private Detection* sub-track.

3.3.2 Public Detection

LinkBox is a team from Tencent Youtu Lab consisting of Jinlong Peng, Yueyang Gu, Yabiao Wang, Chengjie Wang, Jilin Li and Feiyue Huang. They won the 1st title on *Public Detection* sub-track. Their method mainly focus on the challenging of MOT task due to the occlusion problem, especially in dense scenes. To overcome this, they first designed the Layer-wise Aggregation Discriminative Model (LADM) to filter the noisy detections. Then, to associate remaining detections correctly, they proposed the Global Attention Feature Model (GAFM) to extract appearance feature and use it to calculate the appearance similarity between history tracklets and new detections. Finally, they propose the Box-Plane Matching (BPM) method to realize association according to the motion and appearance similarity between history tracklets and new detections.

Selective JDE is the same team as *Adaptive FairMOT* of *Private Detection* sub-track. The method is also the same. They also won the 2nd place on *Private Detection* sub-track.

FCS-Track is the same team as *NewTracker* of *Private Detection* sub-track. The method is also the same. They also won the 3rd place on *Private Detection* sub-track.

References

1. Mykhaylo Andriluka, Umar Iqbal, Eldar Insafutdinov, Leonid Pishchulin, Anton Milan, Juergen Gall, and Bernt Schiele. Posetrack: A benchmark for human pose estimation and tracking. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 5167–5176, 2018.
2. Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. arXiv:2003.09003, 2020.
3. James Ferryman and Ali Shahrokni. Pets2009: Dataset and challenge. In *2009 Twelfth IEEE Intl. workshop on performance evaluation of tracking and surveillance*, pages 1–6. IEEE, 2009.
4. Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.
5. Weiyao Lin, Huabin Liu, Shizhan Liu, Yuxi Li, Rui Qian, Tao Wang, Ning Xu, Hongkai Xiong, Guo-Jun Qi, and Nicu Sebe. Human in events: A large-scale benchmark for human-centric video analysis in complex events. *arXiv preprint*arXiv:2005.04490, 2020.
6. Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. arXiv:1603.00831, 2016.
7. Jinlong Peng, Yueyang Gu, Yabiao Wang, Chengjie Wang, Jilin Li, and Feiyue Huang. Dense scene multiple object tracking with box-plane matching. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4615–4619, 2020.

8. Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conf. on Computer Vision*, pages 17–35, 2016.
9. Bing Shuai, Andrew Berneshawi, Manchen Wang, Chunhui Liu, Davide Modolo, Xinyu Li, and Joseph Tighe. Application of multi-object tracking with siamese track-rcnn to the human in events dataset. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4625–4629, 2020.
10. Ancong Wu, Chengzhi Lin, Bogao Chen, Weihao Huang, Zeyu Huang, and Wei-Shi Zheng. Transductive multi-object tracking in complex events by interactive self-training. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4620–4624, 2020.