Huchuan Lu · Dong Wang

Online Visual Tracking

Springer

Online Visual Tracking

Huchuan Lu · Dong Wang

Online Visual Tracking

Springer

Huchuan Lu
Dalian University of Technology
Dalian, China

Dong Wang
Dalian University of Technology
Dalian, China

ISBN 978-981-13-0468-2 ISBN 978-981-13-0469-9 (eBook)
https://doi.org/10.1007/978-981-13-0469-9

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

This book introduces some representative trackers through practical algorithm analysis and experimental evaluations. This book is intended for professionals and researchers who are interested in visual tracking, also students can take it as a reference book. Readers will get comprehensive knowledge of tracking and can learn state-of-the-art methods through this content. In general, the book is organized as follows:

Chapter 1 provides a brief introduction of visual tracking. First, we introduce basic components of tracking including representation scheme, search mechanism, and model update. Then, challenges in visual tracking are displayed. Finally, we show the datasets and evaluation metrics to evaluate trackers in the following chapters.

In Chaps. 2–7, tracking methods based on sparse representation, local model, model fusion, foreground-background segmentation, correlation filters, and advanced deep learning techniques are introduced to understand tracking in a comprehensive view. In each chapter, we first give a brief introduction of the topic and the existing methods. Then, a few representative trackers and their experimental results are discussed in detail. Finally, each chapter ends with a summary.

The last chapter summarizes the book and points out some potential directions of future research for visual tracking.

Dalian, China
February 2019

Huchuan Lu

Acknowledgements

First of all, we would like to express our gratitude to our collaborators, Prof. Ming-Hsuan Yang, Mr. Xu Jia, Mr. Wei Zhong, Mr. Dong Wang, Mr. Chong Sun, Ms. Peixia Li, Mr. Boyu Chen, Mr. Lijun Wang, Mr. Pingping Zhang, Mr. Shu Wang, Mr. Fan Yang, Ms. Yunhua Zhang, and Ms. Huilan Jiang. We would also like to thank the organizers of visual tracker benchmark, Dr. Yi Wu, Dr. Jongwoo Lim, and Prof. Ming-Hsuan Yang. Their efforts on the large-scale benchmark make us conduct sufficient experiments for different types of online trackers. We are thankful to the Springer editors, Celine Chang and Jane Li, for their enthusiasm, support, patience, and understanding in the course of our preparation of this work. Part of the content in this book is based on the work supported by the National Natural Science Foundation of China under Nos. 61725202, 61751212, 61829102, 61528101, 61502070, 61472060, and 61272372.

Contents

Chapter 1
Introduction to Visual Tracking

Visual tracking is a rapidly evolving field of computer vision that has been attracting increasing attention in the vision community. One reason is that visual tracking offers many challenges as a scientific problem. Moreover, it is a part of many high-level problems of computer vision, such as motion analysis, event detection, and activity understanding. In this chapter, we give a detailed introduction to visual tracking which includes basic components of tracking algorithms, difficulties in tracking, datasets used to evaluate trackers, and evaluation metrics.

1.1 Basic Components of Visual Tracking Algorithms

Considerable progress in the field of object tracking has been made in the past few decades. Online tracking algorithms typically include three fundamental components, namely, representation scheme, search mechanism, and model update.

Representation Scheme: Object representation is one of the major components of any visual tracking algorithm. Since the early work of Lucas and Kanade (LK) [27], holistic templates (based on raw intensity values) have been widely used for tracking [1, 28]. However, when the visual properties of a target object change significantly, the LK approaches do not perform well. Matthews et al. [28] developed a template update method by exploiting the information of the first frame to correct drifts. Subspace-based tracking approaches have been proposed to effectively account for appearance changes. In [13], Hager and Belhumeur proposed an efficient LK algorithm and used low-dimensional representations for tracking under varying illumination conditions. To enhance tracking robustness, Black and Jepson [6] proposed an algorithm using a pre-trained view-based eigenbasis representation and adopted a robust error norm.

Recently, many tracking methods based on sparse representations have been proposed. For instance, Mei and Ling [29, 30] used a dictionary of holistic intensity templates composed of target and trivial templates. Local sparse representations

H. Lu and D. Wang, *Online Visual Tracking*,
https://doi.org/10.1007/978-981-13-0469-9_1

and collaborative representations for object tracking have also been introduced to handle occlusion. To enhance tracking robustness, a local sparse appearance model was proposed in [26] with the mean shift (MS) algorithm to locate objects. By assuming the representation of particles as jointly sparse, Zhang et al. [45] formulated object tracking as a multi-task sparse learning problem. Zhong et al. [46] proposed a collaborative tracking algorithm that combines a sparsity-based discriminative classifier and a sparsity-based generative model. In [18], sparse codes of local image patches with spatial layout in an object were used to model the object appearance for tracking. To deal with outliers in object tracking, Wang et al. [41] proposed a least soft-threshold squares algorithm by modeling image noise with the Gaussian–Laplacian distribution other than the trivial templates used in [29].

A number of tracking methods based on color histograms have been developed. Comaniciu et al. [9] applied the mean shift algorithm to object tracking on the basis of color histograms. Collins [7] extended the mean shift tracking algorithm to deal with the scale variation of target objects. Birchfield and Rangarajan [5] proposed the spatiogram to capture the statistical properties of pixels and their spatial relationships instead of relying on pixel-wise statistics. A locality sensitive histogram [15] was developed by considering the contribution of local regions at each pixel to clearly describe the visual appearance for object tracking. Histograms of oriented gradients have been adopted for tracking [38] to exploit local directional edge information. Representations based on covariance region descriptors [39] were introduced to object tracking to fuse different types of features. In covariance descriptors, the spatial and statistical properties as well as their correlations are characterized within the same representation. In addition, local binary patterns (LBP) [31] and Haar-like features [40] have been utilized to model object appearance for tracking [4, 11].

Recently, discriminative models have been developed in the field of visual tracking. In these models, a binary classifier is learned online to separate the target from the background. Numerous classifiers have been adopted to visual tracking, and they include support vector machine (SVM), structured output SVM, ranking SVM, boosting, semi-boosting, and online multi-instance boosting. To handle appearance changes, Avidan [2] integrated a trained SVM classifier in an optical flow framework for tracking. In [8], the most discriminative feature combination was learned online to build a confidence map in each frame and thereby separate a target object from the background. In [3], an ensemble of online learned weak classifiers was used to determine whether a pixel belongs to the target region or background. Grabner et al. [11] proposed an online boosting method to select discriminative features for the separation of a foreground object and the background. To balance tracking adaptivity and drifting, Stalder et al. [36] combined multiple supervised and semi-supervised classifiers for tracking. Multiple instance learning (MIL) has also been applied to tracking [4]; that is, all ambiguous positive and negative samples are put into bags to learn a discriminative model. Hare et al. [14] designed a tracking algorithm based on a kernelized structured SVM, which exploits the constraints of predicted outputs.

Several approaches based on multiple representation schemes have also been developed to effectively handle appearance variations. Stenger et al. [37] fused multiple observation models online in a parallel or cascaded manner. Recently,

Kwon and Lee [23] developed an object tracking decomposition algorithm that uses multiple observation and motion models to account for relatively large appearance variations caused by drastic lighting changes and fast motion. This approach has been further extended to search for appropriate trackers by Markov chain Monte Carlo sampling [24].

Search Mechanism: Deterministic and stochastic search methods have been developed to estimate object states. When a tracking problem is posed within an optimization framework with an objective function differentiable with respect to motion parameters, gradient descent methods can be used to locate targets efficiently. In [3], the first-order Taylor expansion was used to linearize nonlinear cost functions, and motion parameters were estimated iteratively. Furthermore, mean shift estimation was used to search targets locally by utilizing the Bhattacharyya coefficient as the similarity metric for kernel-regularized color histograms [8]. In [16], Sevilla Lara and Learned Miller proposed a tracking algorithm based on distribution fields; this algorithm allows smoothing objective functions without blurring images and locates targets by searching for the local minimum on the basis of a coarse-to-fine strategy.

However, objective functions for object tracking are usually nonlinear with many local minima. Dense sampling methods have been adopted [4, 11, 14] to alleviate this problem at the expense of a high computational load. Meanwhile, stochastic search algorithms such as particle filters [32] have been widely used because they are relatively insensitive to the local minimum and are computationally efficient. Recent methods based on particle filters have been developed using effective observation models [18, 29, 33] with demonstrated success.

Model Update: The online updating of target representations to account for appearance variations has been known to play an important role in robust object tracking. Matthews et al. [28] addressed the template update problem of the LK algorithm by updating templates using the combination of a fixed reference template extracted from the first frame and the result from the most recent frame. Effective update algorithms have also been proposed in the form of online mixture models [17], online boosting [11], and incremental subspace updating [33].

Considerable attention has been paid to the drawing of samples that are effective in training online classifiers in discriminative models. Grabner et al. [12] formulated the update problem as a semi-supervised task in which the classifier is updated with labeled and unlabeled data; such task is in contrast to supervised discriminative object tracking. To handle ambiguously labeled positive and negative samples obtained online, Babenko et al. [4] focused on the tracking problem within the multiple instance learning framework and developed an online algorithm. To exploit the underlying structure of unlabeled data, Kalal et al. [19] developed a tracking algorithm within the semi-supervised learning framework to select positive and negative samples for model updating. In [14], a tracking algorithm that directly predicts target location changes between frames on the basis of structured learning was proposed. Yu et al. [44] presented a tracking method that is based on co-training to combine

generative and discriminative models. Although considerable progress has clearly been made, developing an adaptive appearance model without drifts remains to be difficult.

1.2 Challenges in Visual Tracking

Many difficulties plague visual tracking, and they include occlusion, scale variation, deformation, fast motion, motion blur, background clutter, in-plane rotation, out-of-plane rotation, illumination variation, out-of-view, and low resolution. The descriptions of these challenges are shown in Table 1.1, and some visual examples are illustrated in Fig. 1.1.

1.3 Tracking Datasets

In this section, we briefly introduce the commonly used datasets for evaluating the performance of different online trackers.

OTB: OTB-2013 [42] contains 50 target objects and OTB-2015 [43] expands the sequences in OTB-2013 to include 100 target objects in the tracking benchmark TB-100 dataset. As some of the target objects are similar or less challenging, 50 difficult and representative ones in the TB-50 dataset were also selected for an in-depth analysis. Note that as humans are the most important target objects in practice, the

Table 1.1 Description of challenging factors in visual tracking

Challenge	Description
Occlusion	The target is partially or fully occluded
Scale variation	The ratio of the bounding boxes of the first frame and the current frame is out of range
Deformation	The object has nonrigid deformation
Fast motion	The motion of the ground truth is too large
Motion blur	The target region is blurred due to the motion of the target or the camera
Background clutter	The background near the target has a similar color or texture as the target
In-plane rotation	The target rotates in the image plane
Out-of-plane rotation	The target rotates out of the image plane
Illumination variation	The illumination in the target region is significantly changed
Out-of-view	Some portion of the target leaves the view
Low resolution	The number of pixels inside the ground truth bounding box is too small

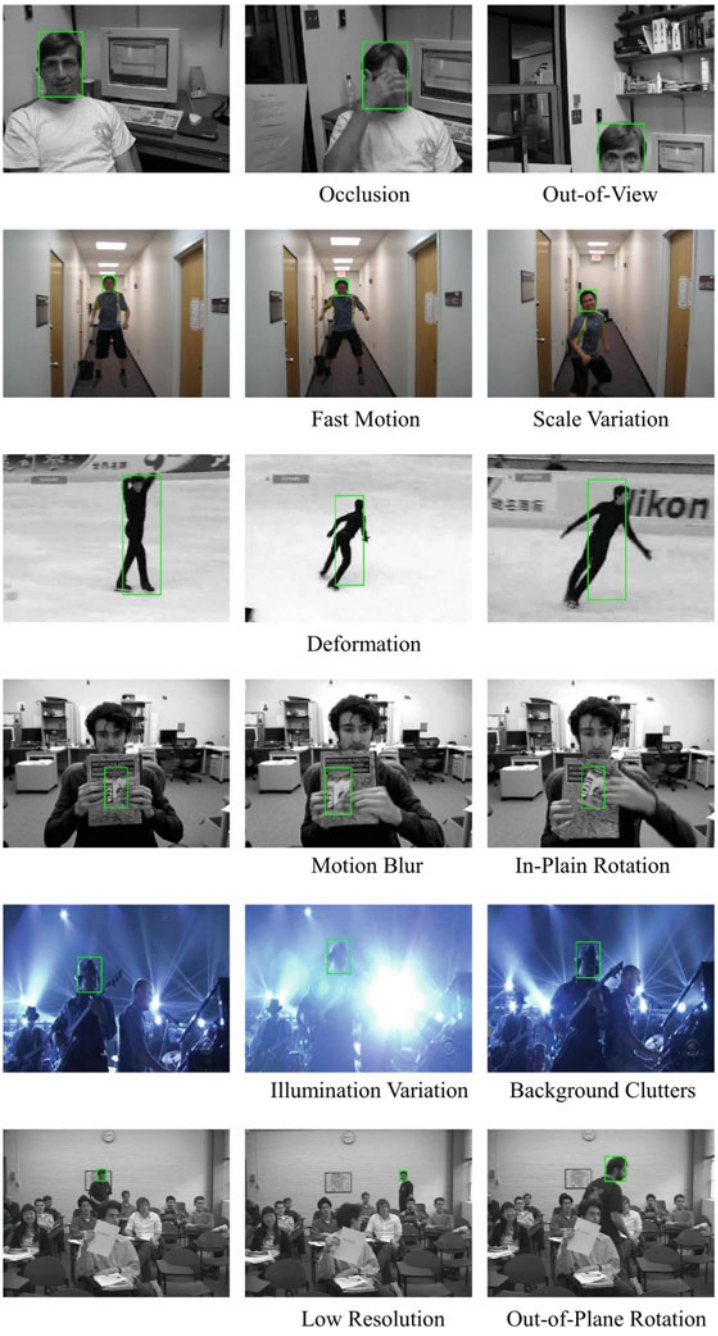


Fig. 1.1 Some visual examples of different challenges in visual tracking

TB-100 dataset contains more sequences of this category (36 body and 26 face/head videos) than of others.

For a satisfactory analysis of the strengths and weaknesses of tracking algorithms, TB-100 categorizes sequences according to 11 attributes, namely, illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutter, and low resolution. Each attribute represents a specific challenging factor in object tracking. One sequence may be annotated with many attributes, and some attributes occur more frequently than others do. The characteristics of tracking algorithms can be clearly analyzed from the sequences with the same attributes. For example, to evaluate how well the tracker handles occlusion, one may use 49 sequences (29 in TB-50) annotated with the OCC attribute.

VOT: The VOT challenges provide the visual tracking community with a precisely defined and repeatable way of comparing short-term trackers as well as a common platform for discussing the evaluation and advancements that are made in the field of visual tracking.

The VOT2015 dataset consists of 60 short sequences annotated with 6 different attributes, namely, occlusion, illumination change, motion change, size change, camera motion, and unassigned. The major difference between VOT2015 and OTB-2015 is that the VOT2015 challenge provides a re-initialization protocol (i.e., trackers are reset with ground truth in the middle of the evaluation if tracking failures are observed).

In [20], the VOT committee analyzed the properties of an average overlap with and without resets in terms of tracking accuracy estimator. The analysis showed that measures with resets can drastically reduce bias. More important, the no-reset measure becomes reliable only on extremely large datasets. Hence, large variances of no-reset estimators combined with small numbers of sequences can distort performance measurements. As datasets typically do not contain sequences of equal lengths, variances are even increased. VOT2013 [22] introduced a ranking-based methodology that accounts for the statistical significance of results, and this methodology was extended in VOT2014 with tests of practical differences [21]. VOT2015 follows the VOT2014 challenge and considers the same class of trackers.

TColor-128 [25]: TColor-128 is a large dataset with 128 color sequences and is devoted to color visual tracking. Sequences in TColor-128 mainly come from new collections and previous studies, such as OTB-2013 and VOT2013. The new collections of TColor-128 contain 78 color sequences newly collected from the Internet. The 78 sequences largely increase the diversity and difficulty of the previous 50 sequences as they involve various circumstances, such as highways, airport terminals, railway stations, concerts, and so on. Similar to that in the OTB dataset [42, 43], each sequence in TColor-128 is also annotated by its challenge factors with 11 attributes.

PTB [35]: PTB has 100 video clips with RGB and depth data. These videos are captured by a standard Microsoft Kinect 1.0, which uses a paired infrared projector

and camera to calculate depth value. However, Kinect's performance is severely impaired in outdoor environments under direct sunlight. Furthermore, Kinect requires minimum and maximum distances between objects and cameras to obtain accurate depth values. Given the above constraints, videos in PTB are captured indoors, with object depth values ranging from 0.5 m to 10m.

ALOV++ [34]: The ALOV++ dataset contains 315 video sequences, 65 sequences have been reported earlier in the PETS workshop, and 250 are new. In composing the ALOV++ dataset, preference was given to many assorted short videos over a few longer ones to maximize diversity. The average length of the short videos is 9.2 s with a maximum of 35 s. One additional category contains 10 long videos with a duration of 1–2 min. The main source of the data is a set of real-life videos from YouTube with 64 different types of targets, including a human face, a person, a ball, an octopus, microscopic cells, a plastic bag, and a can. The collection is categorized by 13 levels of difficulty, from "hard" to "very hard" videos, including a dancer, a rock singer in a concert, complete transparent glass, octopus, a flock of birds, a soldier in camouflage, a completely occluded object, and videos with extreme zooming introducing abrupt motions of targets. The data in ALOV++ are annotated by a rectangular bounding box with a flexible size along the main axes every fifth frame. In rare cases, when motion is rapid, the annotation becomes frequent. The ground truth is acquired for intermediate frames by linear interpolation. The ground truth bounding box in the first frame is specified for the trackers.

1.4 Evaluation Metrics for Visual Tracking

For quantitative evaluation, the precision plot and success plot are usually exploited to compare different online trackers with the OTB-100 dataset.

Precision Plot: One widely used evaluation metric for object tracking is the center location error, which computes the average Euclidean distance between the center locations of the tracked targets and the manually labeled ground truth positions of all frames. When a tracking algorithm loses track of a target object, the output location can be random, and thus, the average error value does not measure the tracking performance correctly. A preferable metric for measuring tracking performance is the percentage of frames in which the estimated locations are within a given threshold distance of the ground truth positions. However, the center location error only measures pixel differences and does not reflect the size and scale of a target object. As a solution, the representative precision plots of trackers averaged over all sequences using the threshold of 20 pixels [4] was introduced.

Success Plot: Another commonly used evaluation metric is the overlap score [10]. The average overlap score (AOS) can be used as a performance measure. In addition, overlap scores can be utilized to determine whether an algorithm successfully tracks a target object in one frame, by testing whether S is larger than a certain threshold.

As the threshold varies between 0 and 1, the success rate changes and the resultant curve is presented. The area under curve (AUC) score of each success plot is usually used to rank different trackers.

1.5 Outline of This Book

This book presents the state-of-the-art methods in online visual tracking, including the motivations, practical algorithms, and experimental evaluations. The outline of this book is as follows: In this chapter, we provide a brief review of fundamental concepts in visual tracking. Chapters 2–7 introduce readers to tracking methods based on sparse representation, local model, model fusion, foreground-background segmentation, correlation filters, and advanced deep learning techniques. Finally, Chap. 8 summarizes the book and points out potential future research directions for visual tracking.

References

1. Alt, N., Hinterstoisser, S., Navab, N.: Rapid selection of reliable templates for visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1355–1362 (2010)
2. Avidan, S.: Support vector tracking. IEEE Trans. Pattern Anal. Mach. Intell. **26**(8), 1064–1072 (2004)
3. Avidan, S.: Ensemble tracking. IEEE Trans. Pattern Anal. Mach. Intell. **29**(2), 261–271 (2007)
4. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. IEEE Trans. Pattern Anal. Mach. Intell. **33**(8), 1619–1632 (2011)
5. Birchfield, S.T., Rangarajan, S.: Spatiograms versus histograms for region-based tracking. IEEE Conf. Comput. Vis. Pattern Recognit. **2**, 1158–1163 (2005)
6. Black, M.J., Jepson, A.D.: Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. Int. J. Comput. Vis. **26**(1), 63–84 (1998)
7. Collins, R.T.: Mean-shift blob tracking through scale space. In: IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. II–234 (2003)
8. Collins, R.T., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1631–1643 (2005)
9. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Trans. Pattern Anal. Mach. Intell. **25**(5), 564–577 (2003)
10. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
11. Grabner, H., Grabner, M., Bischof, H.: Real-time tracking via on-line boosting. In: The British Machine Vision Conference, vol. 1, p. 6 (2006)
12. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised on-line boosting for robust tracking. In: European Conference on Computer Vision, pp. 234–247 (2008)
13. Hager, G.D., Belhumeur, P.N.: Efficient region tracking with parametric models of geometry and illumination. IEEE Trans. Pattern Anal. Mach. Intell. **20**(10), 1025–1039 (1998)
14. Hare, S., Saffari, A., Torr, P.H.: Struck: structured output tracking with kernels. In: IEEE International Conference on Computer Vision, pp. 263–270 (2011)
15. He, S., Yang, Q., Lau, R.W., Wang, J., Yang, M.H.: Visual tracking via locality sensitive histograms. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2427–2434 (2013)

16. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Trans. Pattern Anal. Mach. Intell. **37**(3), 583–596 (2015)
17. Jepson, A.D., Fleet, D.J., El-Maraghi, T.F.: Robust online appearance models for visual tracking. IEEE Trans. Pattern Anal. Mach. Intell. **25**(10), 1296–1311 (2003)
18. Jia, X., Lu, H., Yang, M.H.: Visual tracking via adaptive structural local sparse appearance model. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1822–1829 (2012)
19. Kalal, Z., Matas, J., Mikolajczyk, K.: Pn learning: bootstrapping binary classifiers by structural constraints. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 49–56 (2010)
20. Kristan, M., Matas, J., Leonardis, A., Vojíř, T., Pflugfelder, R., Fernandez, G., Nebehay, G., Porikli, F., Čehovin, L.: A novel performance evaluation methodology for single-target trackers. IEEE Trans. Pattern Anal. Mach. Intell. **38**(11), 2137–2155 (2016)
21. Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., Cehovin, L., Nebehay, G., Vojir, T., Fernandez, G., Lukezic, A., Dimitriev, A.: The visual object tracking vot2014 challenge results. In: European Conference on Computer Vision Workshops, vol. 8926, pp. 191–217 (2015)
22. Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., Porikli, F., Cehovin, L., Nebehay, G., Fernandez, G., Vojir, T., Gatt, A., Khajenezhad, A., Salahledin, A., Soltani-Farani, A., Zarezade, A., Petrosino, A., Milton, A., Bozorgtabar, B., Li, B., Chan, C.S., Heng, C., Ward, D., Kearney, D., Monekosso, D., Karaimer, H.C., Rabiee, H.R., Zhu, J., Gao, J., Xiao, J., Zhang, J., Xing, J., Huang, K., Lebeda, K., Cao, L., Maresca, M.E., Lim, M.K., Helw, M.E., Felsberg, M., Remagnino, P., Bowden, R., Goecke, R., Stolkin, R., Lim, S.Y., Maher, S., Poullot, S., Wong, S., Satoh, S., Chen, W., Hu, W., Zhang, X., Li, Y., Niu, Z.: The visual object tracking vot2013 challenge results. In: IEEE International Conference on Computer Vision Workshops, pp. 98–111 (2013)
23. Kwon, J., Lee, K.M.: Visual tracking decomposition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1269–1276 (2010)
24. Kwon, J., Lee, K.M.: Tracking by sampling trackers. In: IEEE International Conference on Computer Vision, pp. 1195–1202 (2011)
25. Liang, P., Blasch, E., Ling, H.: Encoding color information for visual tracking: algorithms and benchmark. IEEE Trans. Image Process. **24**(12), 5630–5644 (2015)
26. Liu, B., Huang, J., Yang, L., Kulikowsk, C.: Robust tracking using local sparse appearance model and k-selection. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1313–1320 (2011)
27. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence, vol. 2, pp. 674–679 (1981)
28. Matthews, L., Ishikawa, T., Baker, S.: The template update problem. IEEE Trans. Pattern Anal. Mach. Intell. **26**(6), 810–815 (2004)
29. Mei, X., Ling, H.: Robust visual tracking using ℓ_1 minimization. In: IEEE International Conference on Computer Vision, pp. 1436–1443 (2009)
30. Mei, X., Ling, H.: Robust visual tracking and vehicle classification via sparse representation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(11), 2259–2272 (2011)
31. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 971–987 (2002)
32. Poppe, R.: Condensation-conditional density propagation for visual tracking. Comput. Vis. Image Underst. **108**, 4–18 (2007)
33. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. Int. J. Comput. Vis. **77**(1–3), 125–141 (2008)
34. Smeulders, A.W.M., Chu, D.M., Cucchiara, R., Calderara, S., Dehghan, A., Shah, M.: Visual tracking: an experimental survey. IEEE Trans. Pattern Anal. Mach. Intell. **36**(7), 1442–1468 (2013)

35. Song, S., Xiao, J.: Tracking revisited using RGBD camera: unified benchmark and baselines. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 233–240 (2013)
36. Stalder, S., Grabner, H., Van Gool, L.: Beyond semi-supervised tracking: tracking should be as simple as detection, but not simpler than recognition. In: IEEE International Conference on Computer Vision Workshops, vol. 3, p. 6 (2009)
37. Stenger, B., Woodley, T., Cipolla, R.: Learning to track with multiple observers. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2647–2654 (2009)
38. Tang, F., Brennan, S., Zhao, Q., Tao, H.: Co-tracking using semi-supervised support vector machines. In: IEEE International Conference on Computer Vision, pp. 992–999 (2007)
39. Tuzel, O., Porikli, F., Meer, P.: Region covariance: a fast descriptor for detection and classification. In: European Conference on Computer Vision, pp. 589–600 (2006)
40. Viola, P., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vis. **57**(2), 137–154 (2004)
41. Wang, D., Lu, H., Yang, M.H.: Least soft-threshold squares tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2371–2378 (2013)
42. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: a benchmark. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2411–2418 (2013)
43. Wu, Y., Lim, J., Yang, M.H.: Object tracking benchmark. IEEE Trans. Pattern Anal. Mach. Intell. **37**(9), 1834–1848 (2015)
44. Yu, Q., Dinh, T.B., Medioni, G.: Online tracking and reacquisition using co-trained generative and discriminative trackers. In: European Conference on Computer Vision, pp. 678–691 (2008)
45. Zhang, T., Ghanem, B., Liu, S., Ahuja, N.: Robust visual tracking via multi-task sparse learning. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2042–2049 (2012)
46. Zhong, W., Lu, H., Yang, M.H.: Robust object tracking via sparse collaborative appearance model. IEEE Trans. Image Process. **23**(5), 2356–2368 (2014)

Chapter 2
Visual Tracking Based on Sparse Representation

Appearance change caused by intrinsic and extrinsic factors is a challenging problem in online object tracking. Sparse representation can learn effective information of tracking objects which is robust to appearance changes. In this chapter, we introduce two main applications of sparse representation in object tracking. The first exploits the strength of both subspace learning and sparse representation for modeling object appearance to handle the occlusion in object tracking. The second one proposes a sparse collaborative model which effectively exploits both holistic templates and local representations to consider drastic appearance changes.[1, 2]

2.1 Introduction

Sparse representation has been successfully applied in numerous vision applications [4, 5, 7, 10]. With sparsity constraints, one signal can be represented in the form of linear combination of only a few basis vectors. Since the first time sparse representation is introduced into visual tracking by Mei and Ling [5], it has been employed to build various efficient trackers (we refer them as sparse trackers in the following paper) with favorable experimental performance against other state-of-the-art trackers.

In [2], the feature vector of a candidate state is reconstructed by both the target templates and the trivial templates (accounting for noisy pixels) with the sparsity and non-negativity constraints on the reconstruction coefficients. In [4], Liu et al. select a sparse set of features for representing target objects and extend the sparsity constraint to the dynamic group sparsity constraint considering the contiguous distribution of noised pixels.

All the aforementioned sparsity-based methods yield impressive tracking performance, however, most of them focus on measuring how a candidate is resembling the

[1] ©2014 IEEE, Reprinted, with permission, from Ref. [12].

[2] ©2012 IEEE, Reprinted, with permission, from Ref. [2].

H. Lu and D. Wang, *Online Visual Tracking*,
https://doi.org/10.1007/978-981-13-0469-9_2

foreground object while ignoring the background information, which makes them subject to drifts when objects are similar to the target appearance or when the target appearance bears some similarity with the background objects due to partial occlusion.

To deal with these problems, [5], dynamic group sparsity which includes both spatial and temporal adjacency is introduced into the sparse representation to enhance the robustness of the tracker. In [4], a local sparse representation scheme is employed to model the target appearance and then represent the basis distribution of the target with the sparse coding histogram which performs well especially in handling the partial occlusion. Jia et al. [2] samples larger overlapped local image patches with fixed spatial layout where there are more spatial structural information in them.

To prevent accumulation of ambiguities caused by the sparse representation with trivial templates, Liu et al. [5] proposed a tracking method which selects a sparse and discriminative set of features to improve efficiency and robustness. As the number of discriminative features is fixed, this method is less effective for object tracking in dynamic and complex scenes. In [4], an algorithm based on histograms of local sparse representation for object tracking is proposed where the target object is located via mode seeking (using the mean shift algorithm) of voting maps constructed by reconstruction errors.

Zhong et al. [12] proposes and demonstrates an effective and robust tracking method based on the collaboration of generative and discriminative modules. The effective appearance models enable the tracker to better handle heavy occlusions.

In this chapter, we will introduce sparse representation from the following aspects, occlusion handling, appearance modeling with local sparse representation, experimental results, and discussion.

2.2 Sparse Representation in Occlusion Handling

Effective appearance model is one of the most essential factors for robust object tracking. Numerous methods have been proposed to design a robust appearance model that reduces drifts, but considerably few attempts have been made to directly address the occlusion problem, which remains as a subject of debate. Compared with part-based models, sparse representation is more efficient and less computationally expensive in dealing with occlusion problems. Then, we will introduce the motivation and development of sparse representation on occlusion handling.

Object Tracking with Incremental Subspace Learning and Sparse Representation: The incremental visual tracking (IVT) method [10] introduces an online update approach for efficiently learning and updating a low-dimensional principal component analysis (PCA) subspace representation of the target object. The PCA subspace representation with online update is effective in dealing with several appearance changes, such as illumination variation and pose change, but the PCA subspace-based representation scheme is sensitive to partial occlusion, which can be explained by Eq. (2.1) as follows:

$$\mathbf{y} = \mathbf{U}\mathbf{z} + \mathbf{e}, \tag{2.1}$$

where $\mathbf{y}$ denotes an observation vector, $\mathbf{z}$ indicates the corresponding coding or coefficient vector, $\mathbf{U}$ represents a matrix of column basis vectors, and $\mathbf{e}$ is the error term. The basic idea of the IVT method is shown in Fig. 2.1a.

In PCA, a hypothesis is that the error vector is small dense noise that is Gaussian distributed with small variances, and can be estimated by $\mathbf{z} = \mathbf{U}^\top \mathbf{y}$, and the reconstruction error can be approximated by $\left\|\mathbf{y} - \mathbf{U}\mathbf{U}^\top \mathbf{y}\right\|_2^2$. However, when partial occlusion occurs, the noise term cannot be modeled with small variance. Thus, the IVT method is sensitive to partial occlusion.

Inspired by sparse representation, Mei et al. [7] present an algorithm (ℓ_1 tracker) by casting the tracking problem as finding the most likely patch with sparse representation and handling partial occlusion with trivial templates by

$$\mathbf{y} = \mathbf{A}\mathbf{z} + \mathbf{e} = [\mathbf{A}\ \mathbf{I}] \begin{bmatrix} \mathbf{z} \\ \mathbf{e} \end{bmatrix} = \mathbf{B}\mathbf{c}, \tag{2.2}$$

where $\mathbf{y}$ denotes an observation vector, $\mathbf{A}$ represents a matrix of templates, $\mathbf{z}$ indicates the corresponding coefficients, and $\mathbf{e}$ is the error term that can be viewed as the coefficients of trivial templates.

By assuming that each candidate image patch is sparsely represented by a set of target and trivial templates, Eq. (2.2) can be solved via ℓ_1 minimization [7] as follows:

$$\min \|\mathbf{y} - \mathbf{B}\mathbf{c}\|_2^2 + \lambda \|\mathbf{c}\|_1, \tag{2.3}$$

where $\|.\|_1$ and $\|.\|_2$ denote the ℓ_1 as well as ℓ_2 norms, respectively.

The underlying assumption of this approach is that error $\mathbf{e}$ can be modeled by arbitrary but sparse noise, and therefore it can be used to handle partial occlusion. As illustrated in Fig. 2.1, the coefficients for both target and trivial templates should be sparse because the target templates are coherent and coefficients for trivial templates are used to model partial occlusion. However, the ℓ_1 tracker consumes a large amount of computations and does not exploit rich and redundant image properties, which can be captured compactly with subspace representations. Introducing ℓ_1 regularization into subspace representation with PCA can solve this problem. For object tracking, we model target appearance with PCA basis vectors, and account for occlusions with trivial templates by

$$\mathbf{y} = \mathbf{U}\mathbf{z} + \mathbf{e} = [\mathbf{U}\ \mathbf{I}] \begin{bmatrix} \mathbf{z} \\ \mathbf{e} \end{bmatrix}, \tag{2.4}$$

where $\mathbf{y}$ denotes an observation vector, $\mathbf{U}$ represents a matrix of column basis vectors, $\mathbf{z}$ indicates the coefficients of basis vectors, and $\mathbf{e}$ is the error term (which can be viewed as the coefficients of trivial templates). In this formulation, the prototypes consist of a small number of PCA basis vectors and a set of trivial templates

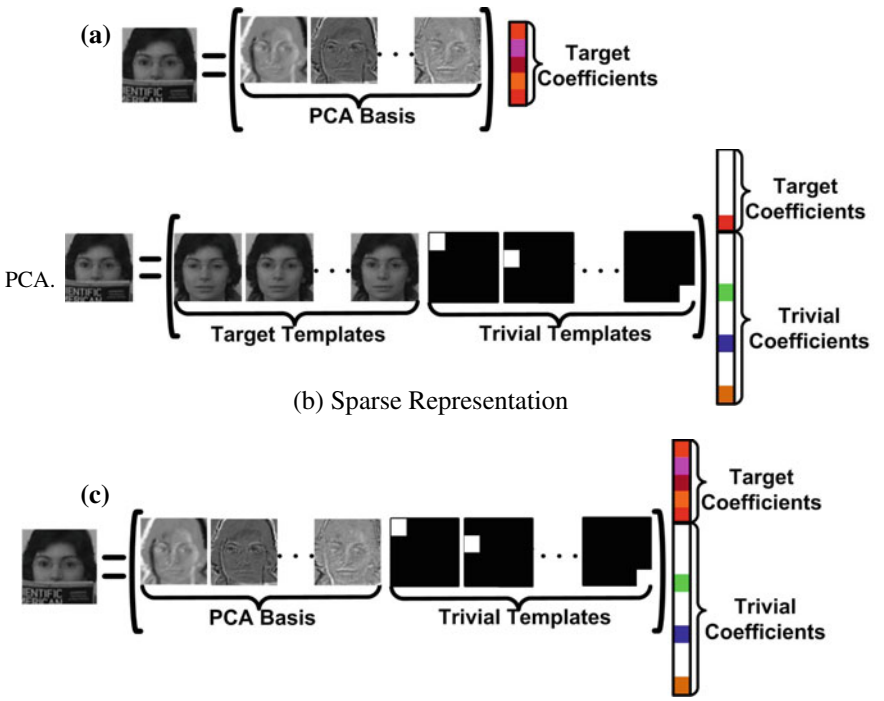


Fig. 2.1 Motivation of our work

(Fig. 2.1c). As the error $\mathbf{e}$ is assumed to be arbitrary but sparse noise, we solve Eq. (2.4) by

$$\min_{\mathbf{z},\mathbf{e}} \frac{1}{2} \|\mathbf{y} - \mathbf{U}\mathbf{z} - \mathbf{e}\|_2^2 + \lambda \|\mathbf{e}\|_1. \tag{2.5}$$

In this formula, coefficients for trivial templates should be sparse while the coefficients for the basis vectors are not sparse because PCA basis vectors are not coherent but orthogonal. As the number of trivial templates is much larger than the number of basis vectors, an observation can be sparsely represented by prototypes.

This different scheme has the following advantages. First, it models occlusion explicitly and therefore handles it effectively. Second, it is able to handle high-resolution image patches with less computational complexity by exploiting subspace representation. Then, we will illustrate how the sparse prototypes are used for object representation as shown in Eq. (2.5).

Object Representation via Orthogonal Basis Vectors and Regularization: Let the objective function be $L(\mathbf{z}, \mathbf{e}) = \frac{1}{2} \|\mathbf{y} - \mathbf{U}\mathbf{z} - \mathbf{e}\|_2^2 + \lambda \|\mathbf{e}\|_1$, the optimization problem is

Table 2.1 Algorithm for computing $\widehat{\mathbf{z}}$ and $\widehat{\mathbf{e}}$

Input: An observation vector $\mathbf{y}$, orthogonal basis vectors $\mathbf{U}$, and a small constant λ .
1: Initialize $\mathbf{e}_0 = \mathbf{0}$ and $i = 0$
2: Iterate
3: Obtain $\mathbf{z}_{i+1}$ via $\mathbf{z}_{i+1} = \mathbf{U}^\top (\mathbf{y} - \mathbf{e}_i)$
4: Obtain $\mathbf{e}_{i+1}$ via $\mathbf{e}_{i+1} = S_\lambda (\mathbf{y} - \mathbf{U}\mathbf{z}_{i+1})$
5: $i \leftarrow i + 1$
6: Until convergence or termination
Output: $\widehat{\mathbf{z}}$ and $\widehat{\mathbf{e}}$

$$\begin{aligned} &\min_{\mathbf{z},\mathbf{e}} L(\mathbf{z}, \mathbf{e}) \\ &\text{s.t. } \mathbf{U}^\top \mathbf{U} = \mathbf{I}, \end{aligned} \tag{2.6}$$

where $\mathbf{y} \in R^{d\times 1}$ denotes an observation vector, $\mathbf{U} \in R^{d\times k}$ represents a matrix of orthogonal basis vectors, $\mathbf{z} \in R^{k\times 1}$ indicates the coefficients of basis vectors, $\mathbf{e} \in R^{d\times 1}$ describes the error term, λ is a regularization parameter, and $\mathbf{I} \in R^{d\times d}$ indicates an identity matrix (where d is the dimension of the observation vector, and k represents the number of basis vectors). As no close-form solution is available for the optimization problem with Eq. (2.6), we propose an iterative algorithm to compute the optimal $\widehat{\mathbf{z}}$ and $\widehat{\mathbf{e}}$.

Lemma 1: Given the optimal sparse noise vector $\widehat{\mathbf{e}}$, the optimal coefficient vector $\widehat{\mathbf{z}}$ can be obtained from $\widehat{\mathbf{z}} = \mathbf{U}^\top (\mathbf{y} - \widehat{\mathbf{e}})$.

Lemma 2: Given the optimal coefficient vector $\widehat{\mathbf{z}}$, the optimal noise $\widehat{\mathbf{e}}$ can be obtained from $\widehat{\mathbf{e}} = S_\lambda (\mathbf{y} - \mathbf{U}\widehat{\mathbf{z}})$, where $S_\tau (x)$ is a shrinkage operation defined as

$$S_\tau (x) = \begin{cases} x - \tau & \text{if } x > \tau \\ \tau - x & \text{if } x < -\tau \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

The algorithm for computing $\widehat{\mathbf{z}}$ and $\widehat{\mathbf{e}}$ is shown in Table 2.1.

Object Tracking via Sparse Prototypes: In this section, we introduce object tracking via sparse prototypes from a dynamic model, an observation model, the update of the observation model. We apply an affine image warp to model the target motion between two consecutive frames. The six parameters of the affine transform are used to model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ of a tracked target. Let $\mathbf{x}_t = \{x_t, y_t, \theta_t, s_t, \alpha_t, \phi_t\}$, where $x_t, y_t, \theta_t, s_t, \alpha_t, \phi_t$ denote x, y translations, rotation angle, scale, aspect ratio, and skew, respectively. The state transition is formulated by random walk, i.e., $p(\mathbf{x}_t|\mathbf{x}_{t-1}) = N(\mathbf{x}_t; \mathbf{x}_{t-1}, \Psi)$, where Ψ is a diagonal covariance matrix.

If no occlusion occurs, then an image observation $\mathbf{y}_t$ can be assumed to be generated from a subspace of the target object spanned by $\mathbf{U}$ and centered at $\boldsymbol{\mu}$. However, we have to account for partial occlusion in an appearance model for robust

object tracking. We assume that a centered image observation $\overline{\mathbf{y}}_t$ ($\overline{\mathbf{y}}_t = \mathbf{y}_t - \boldsymbol{\mu}$) of the tracked object can be represented by a linear combination of the PCA basis vectors U and few elements of the identity matrix $\mathbf{I}$ (i.e., trivial templates) (Fig. 2.1c), i.e., $\overline{\mathbf{y}}_t = \mathbf{U}\mathbf{z}_t + \mathbf{e}_t$. We note that $\mathbf{U}$ consists of a few basis vectors and $\mathbf{z}_t$ is usually dense. On the other hand, $\mathbf{e}_t$ accounts for noise or occlusion. Some samples drawn by our dynamic model are shown in Fig. 2.2. If no occlusion occurs, then the most likely image patch can be effectively represented by the PCA basis vectors and coefficients corresponding to trivial templates (referred to as trivial coefficients) tend to be zeros (as illustrated by the sample $\mathbf{y}^1$ of Fig. 2.2b). On the other hand, a candidate patch that does not correspond to the true target location (e.g., misaligned sample) often leads to a dense representation (as illustrated in the samples $\mathbf{y}^2$ and $\mathbf{y}^3$ of Fig. 2.2b). If partial occlusion occurs, then the most likely image patch can be represented as a linear combination of PCA basis vectors and a small number of trivial templates (as illustrated by $\mathbf{y}^4$ in Fig. 2.2c). As shown in Fig. 2.2c, the trivial coefficients of the sample that best matches the target, $\mathbf{y}^4$, are much sparser than those that do not correspond to the true object location ($\mathbf{y}^5$ and $\mathbf{y}^6$). Based on these observations, the precise localization of the tracked target can be benefited by penalizing the sparsity of trivial coefficients.

For each observation corresponding to a predicted state, we solve the following equation efficiently using the proposed algorithm as summarized in Table 2.1:

$$L\left(\mathbf{z}^i, \mathbf{e}^i\right) = \min_{\mathbf{z}^i, \mathbf{e}^i} \frac{1}{2} \left\| \overline{\mathbf{y}}^i - \mathbf{U}\mathbf{z}^i - \mathbf{e}^i \right\| + \lambda \left\| \mathbf{e}^i \right\|_1, \tag{2.8}$$

and obtain $\mathbf{z}^i$ and $\mathbf{e}^i$, where i denotes the i-th sample of the state $\mathbf{x}$ (without loss of generality, the frame index t is dropped). The observation likelihood can be measured by the reconstruction error of each observed image patch,

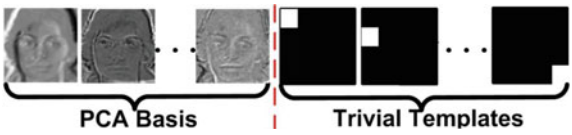
$$p(\overline{\mathbf{y}}^i | \mathbf{x}^i) = \exp(-\|\overline{\mathbf{y}}^i - \mathbf{U}\mathbf{z}^i\|_2^2). \tag{2.9}$$

However, Eq. (2.9) does not consider occlusion. Thus, we use a mask to factor out non-occluding and occluding parts,

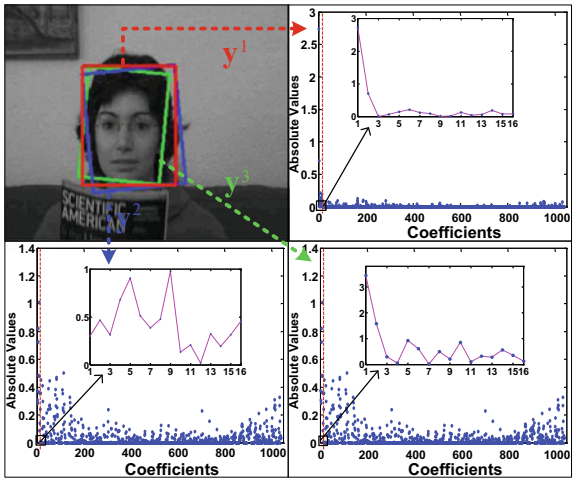
$$p\left(\overline{\mathbf{y}}^i | \mathbf{x}^i\right) = \exp\left[-\left(\left\|\mathbf{w}^i \odot \left(\overline{\mathbf{y}}^i - \mathbf{U}\mathbf{z}^i\right)\right\|_2^2 + \beta \sum \left(1 - \mathbf{w}^i\right)\right)\right], \tag{2.10}$$

where $\mathbf{w}^i = \left[w_1^i, w_2^i, ..., w_d^i\right]^\top$ is a vector that indicates the zero elements of $\mathbf{e}^i$, $\odot$ is the Hadamard product (element-wise product), and β is a penalty term (simply set to λ in this study). If the j-th element of $\mathbf{e}^i$ (obtained from Eq. (2.8)) is zero then $w_j^i = 1$; otherwise, $w_j^i = 0$. The first part of the exponent accounts for the reconstruction error of the unoccluded proportion of the tracked object, and the second term aims to penalize labeling any pixel as being occluded.

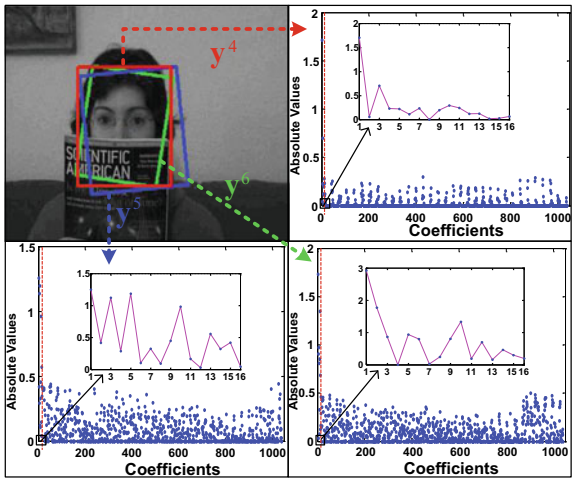
Updating the observation model is essential to address the appearance change of a target object for visual tracking. The model degrades if some imprecise samples are used for updates, thereby causing tracking drift and failure. Instead, we explore



(a) Prototypes



(b) No occlusion



(c) When a target object is occluded

Fig. 2.2 Coefficients and alignment as well as occlusions. This figure illustrates how the PCA and trivial coefficients indicate whether a sample is aligned on the target when it is unconcluded or occluded. **a** Prototypes consist of PCA basis vectors and trivial templates. **b** Good and bad candidates when no occlusion occurs. **c** Good and bad candidates when partial occlusion occurs. The red bounding box represents a good candidate while the blue and green bounding boxes denote two bad samples. The coefficients of PCA basis vectors are shown with circles on the right. The trivial coefficients are shown with solid lines in the subfigures pointed by arrows

the trivial coefficients for occlusion detection because the corresponding templates are used to account for noise. First, each trivial coefficient vector corresponds to a 2D map as a result of reverse raster scan of an image patch. The nonzero elements

of this map indicate that pixels are occluded (referred to as occlusion map). Second, we compute the ratio η of the number of nonzero pixels and the number of occlusion map pixels. We use two thresholds, namely, tr_1 and tr_2, to describe the degree of occlusion (e.g., $tr_1 = 0.1$ and $tr_2 = 0.6$). Third, based on the occlusion ratio η, full, partial, or no update is applied. If $\eta < tr_1$, then at most a small portion of the pixels of the target object are occluded or contaminated by noise, and we directly update the model with this sample. If $tr_1 \leq \eta \leq tr_2$, then the target is partially occluded. We then replace the occluded pixels by its corresponding parts of the average observation $\boldsymbol{\mu}$, and use this recovered sample for update. Otherwise, if $\eta > tr_2$, then a significant part of the target object is occluded, and we discard this sample without update. After we cumulate enough samples, we use an incremental PCA method [8] to update our observation model (i.e., PCA basis vectors $\mathbf{U}$ and the average vector $\boldsymbol{\mu}$).

Our tracker is robust because it can address the presence of potential outliers (e.g., occlusion and misalignment) with the proposed observation model and update scheme. For the accurate location of the tracked target, the proposed representation model (2.8) and observation likelihood (2.10) enable the tracker to handle partial occlusion explicitly and facilitate its selection of the well-aligned observation (see Fig. 2.2 for illustrated examples). With the update scheme of the proposed observation model, our tracker is able to alleviate the problem caused by inaccurate samples (i.e., update the model by outliers).

2.3 Sparse Collaborative Appearance Model

In Sect. 2.2, we introduce the tracking algorithm based on the sparse noise assumption, which could effectively handle partial occlusions. However, its adopted holistic representation limits the tracking performance on the large-scale dataset. To address this issue, in this section, we propose a robust object tracking algorithm based on a sparse collaborative model. This method effectively exploits both holistic templates and locals representations to consider drastic appearance changes. Within the proposed collaborative appearance model, we develop a sparse discriminative classifier (SDC) and sparse generative model (SGM) for object tracking. In the SDC module, we present a classifier that separates the foreground object from the background based on holistic templates. In the SGM module, we propose a histogram-based method that considers the spatial information of each local patch. The update scheme considers the most recent observations and original templates, thereby enabling the proposed algorithm to effectively deal with appearance changes and alleviate the tracking drift problem.

Sparsity-based Discriminative Classifier (SDC): Given the first frame with the tracked object, we sample n_p positive samples around the target location within a radius of few pixels, and n_n negative samples within an annular region a few pixels away from the target object. The positive template vector in the positive training set

is constituted with the stack of gray-scale values of all positive samples, so as to negative training set.

The constructed gray-scale feature space is rich yet redundant, and the sparse coding scheme facilities extracting determinative ones that distinguish foreground from background,

$$\min_{\mathbf{s}} \left\| \mathbf{A}^{\top}\mathbf{s} - \mathbf{p} \right\|_2^2 + \lambda_1 \|\mathbf{s}\|_1, \tag{2.11}$$

where $\mathbf{A} \in \mathbb{R}^{K \times (n_p + n_n)}$ is composed of n_p positive templates $\mathbf{A}_+$ and n_n negative templates $\mathbf{A}_-$, K is the dimension of the features, and λ_1 is a weight parameter. Each element of the vector $\mathbf{p} \in \mathbb{R}^{(n_p+n_n)\times 1}$ represents the property of each template in the training set $\mathbf{A}$, i.e., $+1$ for positive templates and -1 for negative templates. The solution of Eq. (2.11) is the sparse vector $\mathbf{s}$, whose nonzero elements correspond to discriminative features selected from the K-dimensional feature space.

According to the sparse project matrix, the original feature space can be projected to the selected feature space. Thus, the training template set and candidates in the projected space are $\mathbf{A}' = \mathbf{SA}$ and $\mathbf{y}' = \mathbf{Sy}$.

Under the assumption that the target can be better represented by the linear combination of positive templates while the background can be better represented by the span of negative templates, a candidate region $\mathbf{y}$ can be represented by the training template set with the coefficients $\mathbf{z}$ computed by

$$\min_{\mathbf{z}} \left\| \mathbf{y}' - \mathbf{A}'\mathbf{z} \right\|_2^2 + \lambda_2 \|\mathbf{z}\|_1, \tag{2.12}$$

where $\mathbf{y}'$ is the projected vector of $\mathbf{y}$ and λ_2 is a weight parameter.

A candidate region with smaller reconstruction error using the foreground template set indicates a greater likelihood of being a target object, and vice versa. Thus, we formulate the confidence value H_c of the candidate $\mathbf{y}$ by

$$H_c = \frac{1}{1 + \exp\left(-\left(\varepsilon_b - \varepsilon_f\right)/\sigma\right)}, \tag{2.13}$$

where $\varepsilon_f = \left\| \mathbf{y}' - \mathbf{A}'_+\mathbf{z}_+ \right\|_2^2$ is the reconstruction error of the candidate $\mathbf{y}$ with the foreground template set $\mathbf{A}'_+$, and $\mathbf{z}_+$ is the corresponding sparse coefficient vector. Similarly, $\varepsilon_b = \left\| \mathbf{y}' - \mathbf{A}'_-\mathbf{z}_- \right\|_2^2$ is the reconstruction error of the candidate $\mathbf{y}$ using the background template set $\mathbf{A}'_-$, and $\mathbf{z}_-$ is the corresponding sparse coefficient vector. The variable σ is fixed to be a small constant that balances the weight of the discriminative classifier and the generative model.

In [10], the reconstruction error is computed based on the target (positive) templates, which is less effective for tracking because both the negative and indistinguishable samples (e.g., patches covering some part of a target object) have large reconstruction errors when computed with the target (positive) template set. Thus, it introduces ambiguities in differentiating whether such patches are from the foreground or background. In contrast, our confidence measure exploits the distinct

properties of the foreground and the background in computing the reconstruction errors to effectively distinguish patches from the positive and negative classes.

Sparsity-based Generative Model (SGM): In addition to the aforementioned SDC model, an effective sparsity-based generative model to capture the appearance variations of the object during the tracking process. The basic components of our SGM are the following: We use overlapped sliding windows on the normalized images to obtain M patches and each patch is converted to a vector $\mathbf{y}_i \in \mathbb{R}^{G\times 1}$, where G denotes the size of the patch. The sparse coefficient vector of each patch is computed by

$$\min_{\mathbf{z}_i} \| \mathbf{y}_i - \mathbf{D}\mathbf{z}_i \|_2^2 + \lambda_3 \|\mathbf{z}_i\|_1, \tag{2.14}$$

where the dictionary $\mathbf{D} \in \mathbb{R}^{G\times J}$ is generated from J cluster centers using the k-means algorithm on the M patches from first frame (which consists of the most representative patterns of the target object), and λ_3 is a weight parameter.

In this study, the sparse coefficient vector $\mathbf{z_i} \in \mathbb{R}^{\mathbf{J}\times\mathbf{1}}$ of each patch is concatenated to form a histogram by

$$\rho = \left[\mathbf{z}_1^\top, \mathbf{z}_2^\top, \cdots, \mathbf{z}_M^\top\right]^\top, \tag{2.15}$$

where $\rho \in \mathbb{R}^{(J\times M)\times 1}$ is the proposed histogram for one candidate region.

To deal with occlusions, we modify the constructed histogram to exclude the occluded patches when describing the target object. A patch with large reconstruction error is regarded as occluding part and the corresponding sparse coefficient vector is set to zero. Thus, a weighted histogram is generated by

$$\varphi = \rho \odot \mathbf{o}, \tag{2.16}$$

where $\odot$ denotes the element-wise multiplication. Each element of $\mathbf{o}$ is an indicator of occlusion at the corresponding patch and is obtained by

$$o_i = \begin{cases} 1 & \varepsilon_i < \varepsilon_0 \\ 0 & \text{otherwise} \end{cases}, \tag{2.17}$$

where $\varepsilon_i = \|\mathbf{y_i} - \mathbf{D}\boldsymbol{\beta_i}\|_2^2$ is the reconstruction error of patch $\mathbf{y}_i$, and ε_0 is a predefined threshold that determines whether the patch is occluded or not. We thus have a sparse histogram $\boldsymbol{\varphi}$ for each candidate region. The proposed representation scheme considers spatial information of local patches and occlusion into account, thereby making it more effective and robust.

We use the histogram intersection function to compute the similarity of histograms between the candidate and the template due to its effectiveness [1] by

$$L_c = \sum\nolimits_{j=1}^{J\times M} \min\left(\boldsymbol{\varphi}_{\boldsymbol{c}}^{\boldsymbol{j}}, \boldsymbol{\psi}^{\boldsymbol{j}}\right), \tag{2.18}$$

where $\boldsymbol{\varphi_c}$ and $\boldsymbol{\psi}$ are the histograms for the c-th candidate and the template. The histogram of the template $\boldsymbol{\psi}$ is generated by Eqs. 2.14–2.16. The patches $\mathbf{D}$ in Eq. 2.14 are all from the first frame and the template histogram is computed only once for each image sequence.

The vector $\mathbf{o}$ in Eq. 2.17 reflects the occlusion condition of the corresponding candidate. The comparison between the candidate and the template should be carried out under the same occlusion condition, so the template and the c-th candidate share the same vector $\mathbf{o}_c$ in Eq. 2.16. For example, when the template is compared with the c-th candidate, the vector $\mathbf{o}$ of the template in Eq. 2.16 is set to $\mathbf{o}_c$.

Collaborative Model: We propose a collaborative model using SDC and SGM modules within the particle filter framework. In our tracking algorithm, both the confidence value based on the holistic templates and the similarity measure based on the local patches contribute to an effective and robust probabilistic appearance model. The likelihood function of the c-th candidate region is computed by

$$\begin{aligned} p\left(\mathrm{z}_t \left| x_t^c \right.\right) &= H_c L_c \\ &= \left(\sum_{j=1}^{J\times M} \min\left(\boldsymbol{\varphi}_{\boldsymbol{c}}^{\boldsymbol{j}}, \boldsymbol{\psi}^{\boldsymbol{j}}\right)\right) / \left(1+\exp\left(-\left(\varepsilon_b - \varepsilon_f\right)/\sigma\right)\right), \end{aligned} \tag{2.19}$$

and each tracking result is the candidate with the maximum a posterior estimation.

The multiplicative formula is more effective in our tracking scheme compared with the alternative additive operation. The confidence value H_c gives higher weights to the candidates considered as positive samples (i.e., ε_f smaller than ε_b) and penalizes the others. As a result, it can be considered as the weight of the local similarity measure L_c.

Update Scheme: Since the appearance of an object often changes significantly during the tracking process, the update scheme is important and necessary. We develop an update scheme in which the SDC and SGM modules are updated independently.

For the SDC module, we update the negative templates every several frames (5 in our experiments) from image regions away (e.g., more than 8 pixels) from the current tracking result. The positive templates remain the same in the tracking process. As the SDC module aims to distinguish the foreground from the background, it is important to ensure that the positive and negative templates are all correct and distinct.

For the SGM module, the dictionary $\mathbf{D}$ is fixed during the tracking process. Therefore, the dictionary is not incorrectly updated due to tracking failures or occlusions. In order to capture the appearance changes and recover the object from occlusions, the new template histogram $\boldsymbol{\psi_n}$ is computed by

$$\boldsymbol{\psi_n} = \mu \boldsymbol{\psi_f} + (1-\mu)\,\boldsymbol{\psi_l} \quad \text{if } O_n < O_0, \tag{2.20}$$

where $\boldsymbol{\psi_f}$ is the histogram at the first frame, $\boldsymbol{\psi_l}$ is the histogram last stored before update, and μ is the weight. The variable O_n denotes the occlusion measure of the tracking result in the new frame. It is computed by the corresponding occlusion indication vector $\mathbf{o}_n$ using

$$O_n = \frac{1}{J \times M} \sum_{i=1}^{J \times M} \left(1 - o_n^i\right). \tag{2.21}$$

The appearance model is updated as long as the occlusion measure O_n in this frame is smaller than a predefined constant O_0. This update scheme preserves the first template, which is usually correct [3, 6, 9] and takes the most recent tracking result into account.

In the collaboration model, holistic templates are incorporated to construct a discriminative classifier that can effectively deal with cluttered and complex background. Local representations are adopted to form a robust histogram that considers the spatial information among local patches with an occlusion handling module, which enables our tracker to better handle heavy occlusion.

2.4 Experimental Results and Discussions

We evaluate the performance of the proposed sparse representation trackers on the benchmark dataset proposed in [11]. Furthermore, we compare the top 10 tracking algorithms in the official website of OTB-2015 with the proposed methods (SCM [12] and ASLA [2]), and the precision score plots and success rate plots are shown in Fig. 2.3.

To analyze the performance comprehensively of the sparse representation algorithms, we also provide the experiment results on different attributes of OTB-2015.

Tables 2.2 and 2.3 indicates that SCM lists the performance of compared trackers in different attributes, and it indicates that SCM and ASLA perform well in most of them, such as occlusion, background clutter, illumination, and scale variation. In the following, we will analyze their advantages in dealing occlusion, background clutter and illumination changes.

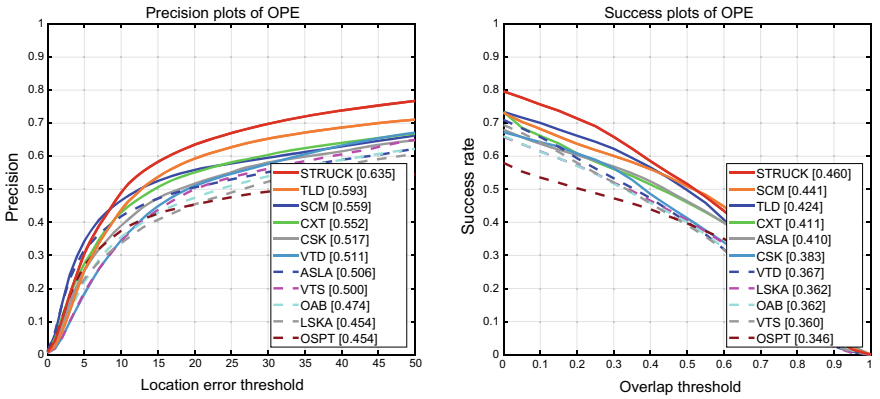


Fig. 2.3 Precision and success plots on OTB-2015 for the top-10 algorithms and proposed methods

Table 2.2 The success comparison of mentioned trackers

Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
STRUCK	*0.428*	***0.383***	***0.452***	***0.448***	*0.420*	*0.347*	***0.454***	**0.388**	***0.424***	***0.363***	*0.404*
TLD	0.354	0.341	**0.421**	*0.427*	0.414	***0.372***	**0.430**	0.365	0.390	**0.343**	0.389
OSPT	0.341	0.261	0.235	0.331	0.347	0.292	0.255	0.328	0.339	0.321	0.320
SCM	***0.457***	**0.381**	0.311	0.412	***0.486***	0.347	0.320	***0.420***	**0.421**	0.324	***0.432***
CXT	0.352	0.297	*0.403*	**0.442**	0.376	**0.366**	*0.403*	0.332	0.391	0.325	0.382
CSK	0.416	0.338	0.331	0.379	0.368	0.251	0.322	0.331	0.355	0.262	0.326
VTD	0.396	0.341	0.280	0.391	0.379	0.235	0.273	0.356	0.398	*0.336*	0.360
ASLA	**0.432**	*0.362*	0.291	0.393	**0.433**	0.341	0.282	*0.367*	*0.411*	0.321	**0.408**
VTS	0.383	0.339	0.277	0.376	0.374	0.203	0.258	0.340	0.388	0.319	0.351
OAB	0.313	0.314	0.364	0.354	0.320	0.266	0.375	0.334	0.340	0.302	0.335
LSK	0.353	0.318	0.312	0.363	0.364	0.300	0.274	0.353	0.369	0.327	0.351

Table 2.3 The precision comparison of mentioned trackers

Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
STRUCK	*0.550*	***0.527***	***0.603***	***0.628***	*0.549*	***0.628***	***0.583***	**0.527**	***0.593***	***0.468***	***0.599***
TLD	0.459	*0.484*	**0.550**	**0.606**	**0.559**	**0.552**	*0.541*	**0.527**	*0.570*	**0.463**	**0.579**
OSPT	0.410	0.336	0.282	0.451	0.431	0.472	0.298	0.428	0.459	*0.415*	0.437
SCM	***0.579***	**0.512**	0.320	0.533	***0.605***	0.484	0.322	***0.541***	0.554	0.409	*0.550*
CXT	0.444	0.394	*0.544*	*0.599*	0.500	***0.492***	**0.561**	0.433	0.524	0.388	0.525
CSK	**0.577**	0.453	0.400	0.512	0.482	0.367	0.379	0.428	0.484	0.283	0.455
VTD	0.536	0.461	0.332	0.559	0.504	0.378	0.304	0.482	**0.574**	0.398	0.526
ASLA	0.531	0.453	0.307	0.495	0.524	0.489	0.284	0.434	0.515	0.384	0.515
VTS	0.512	0.462	0.326	0.534	0.502	0.347	0.281	0.448	0.558	0.360	0.513
OAB	0.410	0.410	0.444	0.461	0.413	0.380	0.443	0.426	0.452	0.336	0.455
LSK	0.444	0.411	0.350	0.476	0.446	0.422	0.274	0.454	0.483	0.391	0.447

Heavy Occlusion: Occlusion is one of the most general yet crucial issues in object tracking. SGM model is designed to overcome heavy occlusion. It estimates the possible occluded patches and develops a robust histogram which only compares the patches that are not occluded. Thus, this scheme effectively alleviates the effects of occlusions. The structural local sparse appearance model in ASLA has both spatial and partial information of the target. Those information helps avoid much influence of occlusion and better estimate the target.

Background Clutter: SDC module is proposed to learn the discrimination between tracked object and background. In addition, the update scheme uses the newly arrived negative templates that facilitate separation of the foreground and the background. In OSPT, the best candidates can be well represented by the PCA basis which attributed to the strength of subspace representation and therefore the error terms are more sparse than those of the misaligned candidates.

Illumination Changes: Illumination changes may confuse foreground and background. But with SDC module, SCM can easily split the low contrast between the foreground and the background. The use of incremental subspace learning in ALSA is able to capture appearance change due to light change, which helps ASLA keep track of the target to the end and achieves low tracking error and high success rate.

2.5 Summary

In this chapter, we introduce two main application of sparse representation in object tracking. By exploiting the strength of both subspace learning and sparse representation for modeling object appearance, it's effective to handle the occlusion in object tracking. The combination of SDC and SGM helps to learn comprehensive appearance model of tracking objects. From the experiments on different challenges of the two algorithm, we can find that introduces ℓ_1 regularization into subspace representation with PCA is helpful in handling partial occlusion. Meanwhile, exploiting both holistic templates and local representations help overcome dramatic appearance changes, illumination and so on. Above all, sparse representation can learn effective information of tracking objects which is robust to appearance changes and heavy occlusion.

References

1. Gao, S., Tsang, I.W., Chia, L., Zhao, P.: Local features are not lonely—laplacian sparse coding for image classification. In: The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, pp. 3555–3561 (2010)
2. Jia, X., Lu, H., Yang, M.: Visual tracking via adaptive structural local sparse appearance model. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1822–1829 (2012)
3. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1409–1422 (2012)
4. Liu, B., Huang, J., Yang, L., Kulikowski, C.A.: Robust tracking using local sparse appearance model and k-selection. In: The 24th IEEE Conference on Computer Vision and Pattern Recognition, pp. 1313–1320 (2011)
5. Liu, B., Yang, L., Huang, J., Meer, P., Gong, L., Kulikowski, C.A.: Robust and fast collaborative tracking with two stage sparse optimization. In: 11th European Conference on Computer Vision, ECCV 2010, pp. 624–637 (2010)
6. Matthews, I.A., Ishikawa, T., Baker, S.: The template update problem. IEEE Trans. Pattern Anal. Mach. Intell. **26**(6), 810–815 (2004)
7. Mei, X., Ling, H.: Robust visual tracking using ℓ_1 minimization. In: IEEE 12th International Conference on Computer Vision, pp. 1436–1443 (2009)
8. Ross, D.A., Lim, J., Lin, R., Yang, M.: Incremental learning for robust visual tracking. Int. J. Comput. Vis. **77**(1–3), 125–141 (2008)
9. Santner, J., Leistner, C., Saffari, A., Pock, T., Bischof, H.: PROST: parallel robust online simple tracking. In: The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, pp. 723–730 (2010)

10. Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S., Ma, Y.: Robust face recognition via sparse representation. IEEE Trans. Pattern Anal. Mach. Intell. **31**(2), 210–227 (2009)
11. Wu, Y., Lim, J., Yang, M.: Object tracking benchmark. IEEE Trans. Pattern Anal. Mach. Intell. **37**(9), 1834–1848 (2015)
12. Zhong, W., Lu, H., Yang, M.: Robust object tracking via sparse collaborative appearance model. IEEE Trans. Pattern Anal. Mach. Intell. **23**(5), 2356–2368 (2014)

Chapter 3
Visual Tracking Based on Local Model

Many effective and efficient tracking methods cannot deal with partial occlusion and background clutter. To address these issues, several local-based models have been employed for designing robust tracking algorithms. In this chapter, we introduce two local-model-based trackers. One improves the discriminative ability by leaning discriminative local weights, and the other takes advantages of both structural and local information of target objects by averaging and alignment pooling.[1, 2]

3.1 Introduction

Developing an efficient and robust tracker is a challenging task due to the difficulties such as partial occlusion, appearance change, illumination variation, and background clutter. Among these challenging factors, occlusion is one of the most general yet crucial problems in visual tracking. Therefore, to obtain an appearance model which can cope with partial occlusion situation, several local-based models have been employed for designing robust tracking algorithms [1, 5, 6, 9, 13, 14].

In contrast to the holistic representation methods, local descriptor-based appearance models tend to preserve the object's structural information and capture changes in detail. This trait is exactly suitable for dealing with partial occlusion problems. Actually, many recent researches have proved the effectiveness of local models adopted in various visual tracking frameworks. Adam et al. [1] proposed a method robust to occlusion which divide the object region into several local fragments, and then locate the target's position by fusing the voting maps of these fragments. He et al. [3] extend the idea of fragment and present the concept of local sensitive histogram for dense matching. Kwon and Lee [5] represent an object by a fixed number of local patches and update the model according to prior knowledge of local patches during tracking. The tracking methods presented in [7] represent the tracked target

[1] ©2012 IEEE, Reprinted, with permission, from Ref. [4].

[2] ©2015 IEEE, Reprinted, with permission, from Ref. [14].

H. Lu and D. Wang, *Online Visual Tracking*,
https://doi.org/10.1007/978-981-13-0469-9_3

with a single histogram which is generated by a series of sparse codes of local image patches and the pooling techniques. Zhong et al. [21] encode each patch by the sparse representation strategy with a static dictionary, and construct a clean histogram by excluding the occluded patches. Xu et al. [18] divide the target into four patches, and compute the response of each patch via the color attributes kernelized correlation filter. In order to handle pose change and background clutter besides occlusion, a novel local part-based tracking method [20] is proposed by introducing spatial latent variables within a structured support vector machine framework. This work adopts a spatial tree structure to describe the appearance of the tracked object, in which an object is represented by a holistic patch and some sub-parts. The spatial configuration of different patches is fully considered by introducing the spatial relation vectors into the model training and updating process. Wang et al. [15] learn a discriminative model between the target and the context, and exploit an online SVM algorithm in hope of discriminating the target from the context patches. Recently, correlation filters (CFs)-based methods make a breakthrough on tracking task, Liu et al. [9] exploit the advantages of CFs to learn multiple patch-based correlation filters and obtain several response maps, then proposed an adaptive weighting method to fuse these response maps for object tracking.

In this chapter, we make two attempts to exploit the local models for solving the tracking problem. First, we consider tracking as a local cosine similarity matching problem between target template and candidates. Based on this configuration, we present an effective method to improve the discriminative ability by learning discriminative local weights. Second, we introduce an effective tracking algorithm based on coarse and fine structural local sparse appearance models with adaptive update. This method exploits both structural and local information of target objects by averaging and alignment pooling.

3.2 Visual Tracking Based on Local Similarity

Weighted Local Cosine Similarity (WLCS): Cosine similarity is a measure of similarity between two vectors of an inner product space, which measures the cosine of the angle between them. The cosine similarity is very common used in many fields such as information retrieval, pattern matching, data mining, and so on. For the tracking problem, we denote $\mathbf{t} \in \mathbb{R}^{d\times 1}$ as the template of the tracked object. Given an observed image vector of a candidate $\mathbf{y} \in \mathbb{R}^{d\times 1}$, the cosine similarity is defined as

$$s(\mathbf{y}, \mathbf{t}) = \frac{\langle \mathbf{y}, \mathbf{t} \rangle}{\|\mathbf{y}\| \, \|\mathbf{t}\|}, \tag{3.1}$$

where $\|.\|$ denotes ℓ_2-norm. We note that the cosine similarity is one of the global matching methods, thus it is usually sensitive to some impulse noise (e.g., partial occlusion).

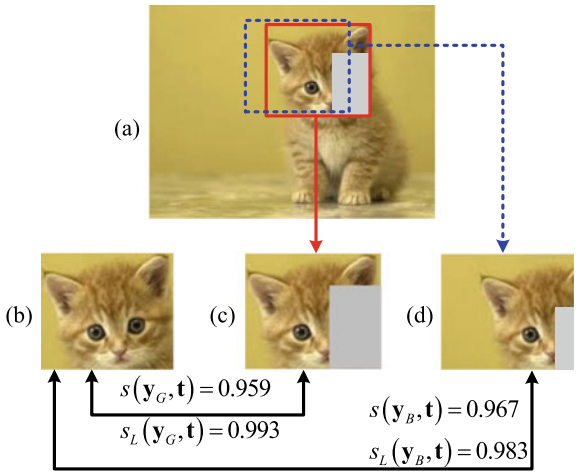


Fig. 3.1 A toy example of good and bad candidates for template matching

Motivated by the ideas of local models (Sect. II.B), we present a local cosine similarity to achieve a good matching. First, we reorganize the candidate vector $\mathbf{y}$ as the concatenation of M local feature vector $\mathbf{y} = \left[\mathbf{y}_1^\top, \mathbf{y}_2^\top, ..., \mathbf{y}_M^\top\right]^\top$, where $\mathbf{y}_i \in \mathbb{R}^{l\times 1}$ denotes a column vector that representing the i-th local block of the image and $M = d/l$. Likewise, the template $\mathbf{t}$ can be represented in the same way as a column vector by its blocks. Then the local cosine similarity can be defined as

$$s_L(\mathbf{y}, \mathbf{t}) = \frac{1}{M}\sum_{i=1}^{M} s(\mathbf{y}_i, \mathbf{t}_i) = \frac{1}{M}\sum_{i=1}^{M} \frac{\langle \mathbf{y}_i, \mathbf{t}_i \rangle}{\|\mathbf{y}_i\| \, \|\mathbf{t}_i\|}, \tag{3.2}$$

where $s(\mathbf{y}_i, \mathbf{t}_i)$ measures the local similarity between $\mathbf{t}_i$ and $\mathbf{y}_i$.

Figure 3.1 demonstrates a toy example of good and bad candidates for template matching with partial occlusion. A single template is shown in Fig. 3.1b. Figure 3.1c, d show good and bad candidates (shown in red and blue boxes respectively). We denote the template as $\mathbf{t}$, the good candidate as $\mathbf{y}_G$ and the bad candidate as $\mathbf{y}_B$ respectively. We can see that $s(\mathbf{y}_G, \mathbf{t}) < s(\mathbf{y}_B, \mathbf{t})$ in this example, which means the bad candidate is selected if the cosine similarity is adopted for template matching. On the other hand, the good candidate is selected ($s_L(\mathbf{y}_G, \mathbf{t}) > s_L(\mathbf{y}_B, \mathbf{t})$) when the local cosine similarity is used. Here, we use 4×4 (i.e., $M = 4 \times 4 = 16$) blocks for calculating the local cosine similarity. We note that the local cosine similarity is more effective than the cosine similarity in handling partial occlusion.

For general purpose, we introduce the weighted local cosine similarity (WLCS) that is defined as

$$s_{WL}(\mathbf{y}, \mathbf{t}) = \sum_{i=1}^{M} w_i s(\mathbf{y}_i, \mathbf{t}_i) = \sum_{i=1}^{M} w_i \frac{\langle \mathbf{y}_i, \mathbf{t}_i \rangle}{\|\mathbf{y}_i\| \, \|\mathbf{t}_i\|}, \tag{3.3}$$

where w_i is a positive weight of the i-th component that satisfies $\sum_{i=1}^{M} w_i = 1$. In **Remark**, we can see that both cosine similarity and local cosine similarity can be viewed as special cases of the weighted local cosine similarity by choosing different weights.

Remark: Both cosine similarity (Eq. 3.1) and local cosine similarity (Eq. 3.2) can be viewed as special cases of the weighted local cosine similarity (Eq. 3.3) by choosing different weights.

(1) The cosine similarity can be viewed as a special case of the weighted local cosine similarity by adopting local brightness ratios as weights. The following equation provides a detailed explanation.

$$\begin{aligned} s(\mathbf{y}, \mathbf{t}) &= \frac{\langle \mathbf{y}, \mathbf{t} \rangle}{\|\mathbf{y}\|\|\mathbf{t}\|} \\ &= \sum_{i=1}^{M} \frac{\|\mathbf{y}_i\|\|\mathbf{t}_i\|}{\|\mathbf{y}\|\|\mathbf{t}\|} \frac{\langle \mathbf{y}_i, \mathbf{t}_i \rangle}{\|\mathbf{y}_i\|\|\mathbf{t}_i\|} \\ &= \sum_{i=1}^{M} w_i \frac{\langle \mathbf{y}_i, \mathbf{t}_i \rangle}{\|\mathbf{y}_i\|\|\mathbf{t}_i\|} \\ &= \sum_{i=1}^{M} w_i s(\mathbf{y}_i, \mathbf{t}_i), \end{aligned} \tag{3.4}$$

where $w_i = \frac{\|\mathbf{y}_i\|\|\mathbf{t}_i\|}{\|\mathbf{y}\|\|\mathbf{t}\|} = r_i^y r_i^t, r_i^y = \|\mathbf{y}_i\|/\|\mathbf{y}\|$ and $r_i^t = \|\mathbf{t}_i\|/\|\mathbf{t}\|$. It can be seen that the cosine similarity intrinsically assigns larger weights to local components with higher brightness ratio, the imperfections of which are two folds. First, a brighter component does not mean it is more significant or discriminative for template matching. Second, if a candidate $\mathbf{y}$ suffers some impulse noises (e.g., partial occlusions and local illumination changes), all local brightness ratio $\left\{r_i^y\right\}_{i=1}^{M}$ may change significantly. As these changes are random and unpredictable, the cosine similarity is usually not effective in handling impulse noises.

(2) By choosing uniform weights (i.e., $w_i = \frac{1}{M}, i = 1, 2, ..., M$), the weighted local cosine similarity is reduced to the local cosine similarity. When some impulse noises occur, the changes in some local components do not affect those components that do not suffer from impulse noises. Thus, it can restrict the negative influence of a local component with errors rather than all components.

(3) The weighted local cosine similarity (WLCS) can be generalized by learning weights when it is used for some special purpose. The next subsection will introduce how to learn discriminative weights for the tracking problem.

(4) We note that the proposed local cosine similarity is very related to the local normalized squared ℓ_2 distance, the detailed illustration of which is as follows:

$$\begin{aligned}
&\sum_{i=1}^{M}\left(\frac{\mathbf{y}_i}{\|\mathbf{y}_i\|}-\frac{\mathbf{t}_i}{\|\mathbf{t}_i\|}\right)^2 \\
&=\sum_{i=1}^{M}\left[\left(\frac{\mathbf{y}_i}{\|\mathbf{y}_i\|}\right)^2+\left(\frac{\mathbf{t}_i}{\|\mathbf{t}_i\|}\right)^2-2\left\langle\frac{\mathbf{y}_i}{\|\mathbf{y}_i\|},\frac{\mathbf{t}_i}{\|\mathbf{t}_i\|}\right\rangle\right] \\
&=\sum_{i=1}^{M}\left[2-2\frac{\langle\mathbf{y}_i,\mathbf{t}_i\rangle}{\|\mathbf{y}_i\|\|\mathbf{t}_i\|}\right] \\
&=2M\left[1-s_L\left(\mathbf{y},\mathbf{t}\right)\right]
\end{aligned}$$

Learning Discriminative Weights via Quadratic Programming: Based on the discussions above, it is very important to determine (or learn) effective weights for the WLCS method. Thus, we develop an optimization method to calculate discriminative weights for the tracking problem. Once we obtain the optimal state in the current frame, we can collect some positive samples and negative samples. The positive samples are densely cropped $\Omega^+=\{\mathbf{y}|\left(\|c\left(\mathbf{y}\right)-c\left(\mathbf{y}^*\right)\|\leq\alpha\right)\}$ within a search radius α centering at the center location of the tracked object. Then the negative samples are randomly sampled from the set $\Omega^-=\{\mathbf{y}|\left(\beta\leq\|c\left(\mathbf{y}\right)-c\left(\mathbf{y}^*\right)\|\leq\gamma\right)\}$, where $\alpha<\beta<\gamma$.

Now, we consider exploiting these positive and negative samples to learn discriminative weights for the WLCS method. For this purpose, we define the following objective function:

$$\begin{aligned}
J\left(\mathbf{w}\right)=&\frac{1}{|\Omega^+|}\sum_{j\in\Omega^+}s_{WL}\left(\mathbf{t},\mathbf{y}^j\right)-\frac{1}{|\Omega^-|}\sum_{j\in\Omega^-}s_{WL}\left(\mathbf{t},\mathbf{y}^j\right) \\
&+\frac{\mu}{2}\sum_{i=1}^{M}\left(w_i-{w_i}'\right)^2
\end{aligned}, \tag{3.5}$$

where the first term $\frac{1}{|\Omega^+|}\sum_{j\in\Omega^+}s_{WL}\left(\mathbf{t},\mathbf{y}^j\right)$ denotes the average WLCS of positive samples, the second term $\frac{1}{|\Omega^-|}\sum_{j\in\Omega^+}s_{WL}\left(\mathbf{t},\mathbf{y}^j\right)$ stands for the average WLCS of negative samples. The last term is a regularization term in which $\mathbf{w}'=\left[w_1',w_2',...,w_M'\right]^\top$ acts as a reference vector. The physical meaning of the first two terms in Eq. (3.5) is to make the WLCS values of positive samples large and the WLCS values of negative samples small in the meanwhile. The regularization term aims to avoid model degradation, or to introduce some prior information (such as temporal consistency).

Thus, we can obtain the discriminative weights by solving the following optimization problem:

$$\begin{aligned}
&\max\ J\left(\mathbf{w}\right) \\
&s.t.\ \sum_{i=1}^{M}w_i=1 \\
&\quad w_i\geq 0,\ i=1,...,M
\end{aligned} \tag{3.6}$$

It is not difficult to prove that the maximization problem in Eq. (3.6) is equivalent to the following minimization problem,

$$\begin{aligned} &\min G(\mathbf{w}) \\ &s.t. \sum_{i=1}^{M} w_i = 1 \\ &\quad w_i \geq 0,\ i = 1, ..., M \end{aligned} \tag{3.7}$$

The objective function $G(\mathbf{w})$ is defined as

$$G(\mathbf{w}) = \sum_{i=1}^{M} w_i \left(S_i^- - S_i^+ + \mu w_i'\right) - \frac{\mu}{2} \sum_{i=1}^{M} w_i^2. \tag{3.8}$$

where $S_i^+ = \frac{1}{|\Omega^+|} \sum_{j \in \Omega^+} s\left(\mathbf{t}_i, \mathbf{y}_i^j\right)$ and $S_i^- = \frac{1}{|\Omega^-|} \sum_{j \in \Omega^-} s\left(\mathbf{t}_i, \mathbf{y}_i^j\right)$. For the optimization problem in Eq. (3.7), the objective function $G(\mathbf{w})$ is a quadratic function and the constraint functions are linear. So, this optimization problem is a quadratic program, which can be solved by many existing software and packages, such as "**quadprog.m**" in the Matlab optimization toolbox.

We note that the regularization parameter μ is very important for the proposed optimization problem and should be set as an appropriate value. If the value of μ is very large, the solution of $\mathbf{w}$ will be very close to the reference weight vector $\mathbf{w}'$ (especially, $\mu = +\infty$ will lead to $\mathbf{w} = \mathbf{w}'$). On the other hand, if the value of μ is very small (i.e., very close to 0), the solution of $\mathbf{w}$ will intend to select only one single local component, which may make the tracker less robust. When $\mu = 0$, the objective function can converted into $G(\mathbf{w}) = \sum_{i=1}^{M} w_i \left(S_i^- - S_i^+\right)$, and the solution of the problem in Eq. (3.7) can be simply obtained as

$$w_i = \begin{cases} 1, & if\ \left(S_i^+ - S_i^-\right) > \left(S_j^+ - S_j^-\right)\ \forall\ j \neq i \\ 0, & otherwise \end{cases}. \tag{3.9}$$

WLCS-based Tracking Framework: Here, we present the basic components of the proposed tracker, the flowchart of which is shown in Fig. 3.2.

By assuming that the variation of the indicator vectors between two consecutive frames is very small, we build our likelihood function based on the WLCS method with the weight vector $\mathbf{w}_{t-1} = \left[w_{t-1}^1, w_{t-1}^2, ..., w_{t-1}^M\right]$ obtained in the last frame. For each observed image vector corresponding to a predicted state, the observation likelihood can be measured by

$$p\left(\mathbf{y}^j | \mathbf{x}^j\right) = \sum_{i=1}^{M} w_{t-1}^i s\left(\mathbf{t}_i, \mathbf{y}_i^j\right), \tag{3.10}$$

where j denotes the j-th sample of the state $\mathbf{x}$ (the frame index t is dropped without loss of generality).

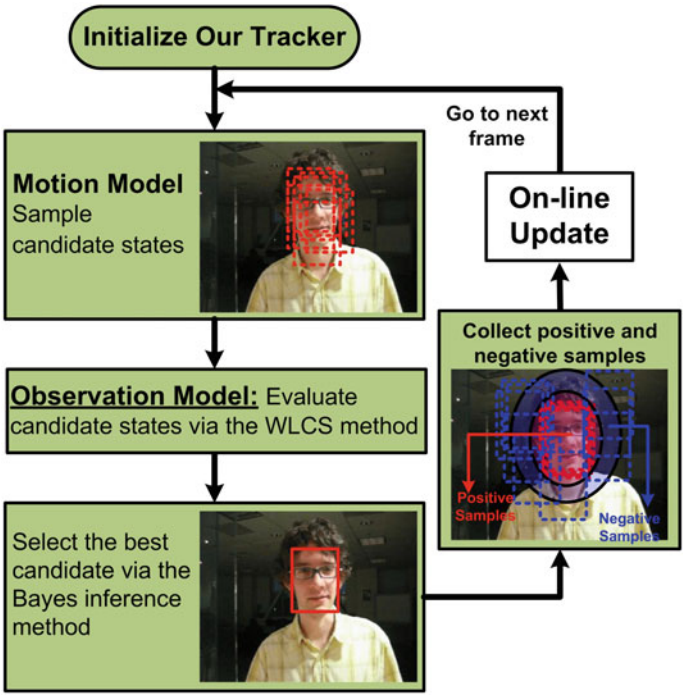


Fig. 3.2 The flowchart of the proposed tracking method

To capture the appearance variations during the tracking process, the online update scheme includes two aspects:

(1) online update the template: after obtaining the best candidate state of the tracked target in the current frame ($\mathbf{x}^*$, we drop the frame index t for clarity). Then we extract its corresponding image blocks $\mathbf{y}^* = \left[\left(\mathbf{y}_1^*\right)^\top, \left(\mathbf{y}_2^*\right)^\top, ..., \left(\mathbf{y}_M^*\right)^\top\right]^\top$ to update the target template ($\mathbf{t} = \left[\mathbf{t}_1^\top, \mathbf{t}_2^\top, ..., \mathbf{t}_M^\top\right]^\top$) as

$$\begin{cases} \mathbf{t}_i \leftarrow \eta \mathbf{t}_i + (1-\eta)\,\mathbf{y}_i^*, & if\ s\left(\mathbf{t}_i, \mathbf{y}_i^*\right) \geq \varepsilon \\ \mathbf{t}_i \leftarrow \mathbf{t}_i, & otherwise \end{cases}, \tag{3.11}$$

where $s\,(.,.)$ denotes the cosine similarity, i.e., $s\,(\mathbf{a}, \mathbf{b}) = \langle \mathbf{a}, \mathbf{b} \rangle / (\|\mathbf{a}\|\,\|\mathbf{b}\|)$. $\varepsilon = 0.85$ is a predefined threshold and η is a update rate that is set to 0.95 in this work.

(2) After updating the object template $\mathbf{t}$, we can collect the positive and negative samples and then solve the optimization problem in Eq. (3.8) to obtain the discriminative weight vector $\mathbf{w}_t$ in the current frame. It should be noted that we choose the discriminative weight vector obtained from the last frame as the reference weight, i.e., $\mathbf{w}' = \mathbf{w}_{t-1}$. This manner is able to consider the temporal consistency of the discriminative weights between consecutive frames. We also note that the discriminative weights in the first frame is initialized as equal values ($w_i = \frac{1}{M}, i = 1, 2, ..., M$).

3.3 Visual Tracking Based on Local Sparse Representation

In this section, we combine the local model and sparse coding representation to further improve the tracking performance. To be specific, we develop coarse and fine structural local appearance models based on sparse representation of both multiple templates and local appearance models. That is, we exploit the consistent local appearance of the object and highlight the role of these fragments for visual tracking. Figure 3.3 shows the main steps of feature formation in this work.

Object-Specific Dictionary: Object-specific dictionary models the target objects of interest better for visual tracking. It has been shown that dictionaries composed of local image patches extracted from the region of a target object in a fixed spatial layout perform well for visual tracking [4, 8, 16]. To construct the dictionary, we collect a set of n templates to model object appearance $\mathbf{T} = [\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n]$ where $\mathbf{T}_i$ is an image observation of a target object. Given $\mathbf{T}$, we extract a set of local image patches inside the target region using a fixed spatial layout as shown in Fig. 3.3. These local patches are used as the dictionary atoms to encode the local patches inside the possible candidate regions, i.e., $\mathbf{D} = \left[\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{(n\times N)}\right] \in \mathbb{R}^{d\times(n\times N)}$, where d is the dimension of the image patch vector, and N is the number of local patches sampled within the target region. Each column in $\mathbf{D}$ is obtained by ℓ_2 normalization of the vectorized local image patches extracted from $\mathbf{T}$. While each local patch represents one fixed part of the target object, the local patches altogether represent the holistic structure of the target. For a target candidate region, we extract the corresponding local patches as $\mathbf{Y} = \left[\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\right] \in \mathbb{R}^{d\times N}$.

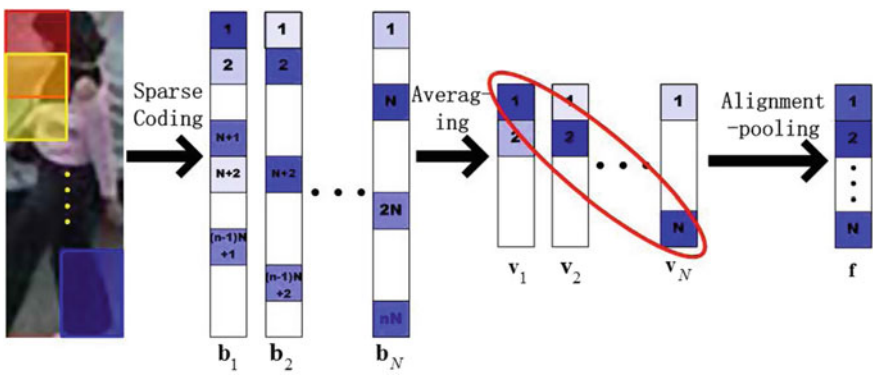


Fig. 3.3 Illustration of local patches and feature vectors formed by averaging and alignment pooling. Each local patch (where the first one is denoted in red, second one in yellow, and the last one in blue) is sparsely represented by the template set with a vector (where elements with larger values are denoted by darker color). These sparse coefficients are averaged and pooled to represent a target object

Sparse Coding and Averaging: With the sparsity assumption, each local patch within a target region can be represented as the linear combination of only a few basis elements of the dictionary by solving

$$\begin{aligned} &\min_{\mathbf{b}_i} \|\mathbf{y}_i - \mathbf{D}\mathbf{b}_i\|_2^2 + \lambda\|\mathbf{b}_i\|_1, \\ &\text{s.t.} \quad \mathbf{b}_i \succeq 0, \end{aligned} \tag{3.12}$$

where $\mathbf{y}_i$ denotes the i-th vectorized local image patch, $\mathbf{b}_i \in \mathbb{R}^{(n\times N)\times 1}$ is the corresponding sparse code of that local patch, and $\mathbf{b}_i \succeq 0$ enforces that all the elements of $\mathbf{b}_i$ are nonnegative. Note $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N]$ represents the sparse coefficients of patches within one candidate region. The proposed algorithm also accommodates another sparse coding method, that is, the elastic net method (Zou and Hastie [22]) based on ℓ_1/ℓ_2 regularization. Using the ℓ_1/ℓ_2 sparse coding method, each local patch is modeled by

$$\begin{aligned} &\min_{\mathbf{b}_i} \|\mathbf{y}_i - \mathbf{D}\mathbf{b}_i\|_2^2 + \lambda_1\|\mathbf{b}_i\|_1 + \tfrac{\lambda_2}{2}\|\mathbf{b}_i\|_2^2, \\ &\text{s.t.} \quad \mathbf{b}_i \succeq 0. \end{aligned} \tag{3.13}$$

The elastic net method combines the ℓ_1 and ℓ_2 regularization together with the hope of taking advantages of both. ℓ_1 regularization tends to find sparse solution but introduces a large Mean Square Error (MSE), while ℓ_2 is able to produce small MSE.

Since the template set contains varying appearances of a target object, the local patterns that frequently appear at the same position are more distinctive than others, and play an important role in robust representation and good alignment. For example, the appearance change on the upper body of a pedestrian is much less than that of the lower body. Thus, it is more effective to recognize this person by the patches from the upper body than other parts. The encoding vector of a local patch is divided into several segments according to the template that each element of the vector corresponds to, i.e., $\mathbf{b}_i^\top = \left[\mathbf{b}_i^{(1)\top}, \mathbf{b}_i^{(2)\top}, \ldots, \mathbf{b}_i^{(n)\top}\right]$, where $\mathbf{b}_i^{(k)} \in \mathbb{R}^{N\times 1}$ denotes the segment of encoding vector $\mathbf{b}_i$ corresponding to the k-th template. To enhance the stability of the sparse coding results, we use equally weighted averaging on different segments of encoding vectors. That is, these segmented coefficients are equally weighted to obtain $\mathbf{v}_i$ for the i-th patch,

$$\mathbf{v}_i = \frac{1}{C}\sum_{k=1}^{n} \mathbf{b}_i^{(k)}, \quad i = 1, 2, \ldots, N, \tag{3.14}$$

where vector $\mathbf{v}_i$ corresponds to the i-th local patch and C is a normalization term.

All the vectors $\mathbf{v}_i$ of local patches in a candidate region form a square matrix $\mathbf{V}$ and are further processed by the proposed pooling method.

Pooling: A single local patch can only capture some local appearance of the object. To model the whole object, it is necessary to pool the information contained in averaged coefficient vectors. We use the alignment pooling instead of the max pooling

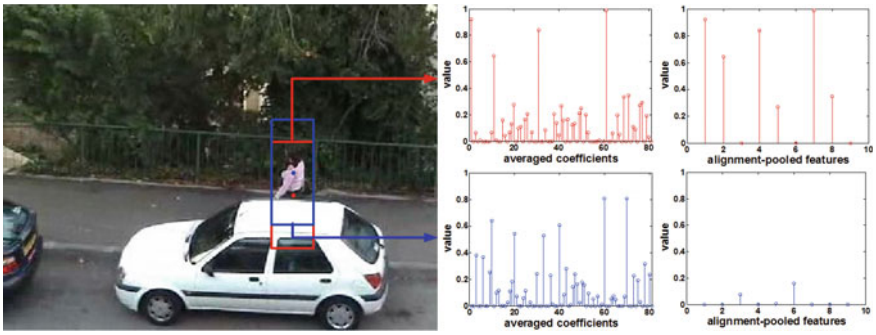


Fig. 3.4 Examples of pooled vectors. The proposed alignment pooling method facilitates to determine stable parts either due to large appearance change (top) or partial occlusion (bottom) as indicated by the feature values

(Riesenhuber and Poggio [10]; Serre et al. [12]; Yang et al. [19]) or directly concatenating these vectors, as it helps alleviate the influence of irrelevant patches. In addition, the alignment pooling method makes full use of the structural information contained in the dictionary and hence helps to locate target objects accurately. After $\mathbf{v}_i$ is computed, each local patch at a certain position within a candidate region is represented by patches at different positions of templates. The local appearance of a patch with small variation is best described by the blocks at the same positions of the templates (i.e., using the sparse codes with the aligned positions). For example, the top left corner patch of the target object in Fig. 3.3 should be best described by patches at the same position of templates. Therefore, the first element of $\mathbf{v}_1$ should have the largest coefficient value. We use the diagonal elements of the square matrix $\mathbf{V}$ as the pooled feature (see Fig. 3.3), i.e.,

$$\mathbf{f} = \text{diag}(\mathbf{V}), \tag{3.15}$$

where $\mathbf{f}$ is the vector of proposed alignment pooled features. After consistent local patterns are computed by the equally weighted averaging operation, this pooling method further aligns local patterns between target candidate and templates based on the locations of structural blocks. Figure 3.4 shows that the tracking result that aligns well with the target object has a higher score computed with the proposed pooling method than the poorly aligned one.

The aligned tracking results also facilitate the incremental subspace learning for template update as the alignment pooling operation enables the proposed appearance model to deal with partial occlusion. When occlusion occurs, encodings of the occluded local patches become dense due to significant appearance change, thereby leading to smaller values in $\mathbf{f}$. However, local patches which are not occluded have similar appearance to templates, and therefore can be described well by a few sparse coefficients and large values in $\mathbf{f}$. The similarity between target candidate regions and

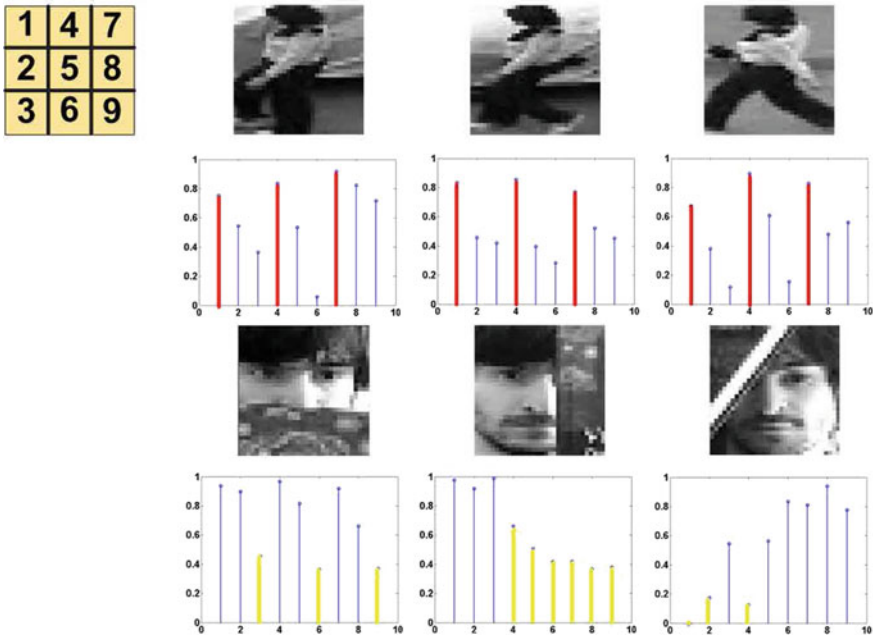


Fig. 3.5 Examples of pooled vectors. The proposed alignment pooling method facilitates determine stable parts either due to large appearance change (top) or partial occlusion (bottom) as indicated by the feature values

the set of target templates computed in this way is still higher than other candidates. Some examples of the pooled vectors are shown in Fig. 3.5.

In the first row, values correspond to the pooled vectors (denoted by red bars) of the upper body (patch 1, 4, and 7) are larger, and the tracking results align with the ground truth locations well. This shows the proposed alignment pooling method facilitates capturing stable structural parts of the target. In the second row, the proposed appearance model with alignment pooling helps handle partial occlusion by down-weighting the contribution of occluded local patches (denoted by yellow bars).

Coarse and Fine Representation: To further improve tracking performance, we adopt a coarse and fine representation to model target appearance. We extract local patches of different sizes within a target region and construct multiple dictionaries at coarse and fine scales. Based on these coarse and fine dictionaries, we obtain pooled features and compute coarse and fine similarities between candidate regions and a target model. The appearance model constructed at fine scale helps distinguish foreground targets from background regions whereas the appearance model at coarse scale facilitates account for appearance change due to large deformation. The final likelihood of a target region is computed by the combination of similarities computed at coarse and fine scales. We evaluate different weighting methods to compute the likelihood of using Eq. (3.16) including similarities computed based on finer

resolution patches are weighted more than those based on coarser ones; similarities computed based on coarser ones are weighted more; and they are equally weighted.

$$\eta_o = \sum_{m=1}^{M} \alpha^m \eta_o^m, \tag{3.16}$$

where η_o^m is the similarity computed on the m-th scale and α^m denotes the weight of the m-th scale.

Template Update: Template updating is an essential step in visual object tracking to deal with illumination and pose variation. To address this issue, we introduce an update method which exploits both multiple templates and local sparse representation to adapt templates to appearance change of target objects, and reduce the influence of the occluded target templates.

When un-occluded, local patterns of a target object in the current and previous frames can be well represented mutually. However, it is not the case when a target object is occluded. We use patches of the tracking result to encode each patch $\mathbf{d}_i$ of each template in $\mathbf{T}$,

$$\begin{aligned} &\min_{\mathbf{s}^{ij}} \left\| \mathbf{d}^{ij} - \hat{\mathbf{Y}}\mathbf{s}^{ij} \right\|_2^2 + \lambda \left\| \mathbf{s}^{ij} \right\|_1, \\ &\text{s.t.} \quad \mathbf{s}^{ij} \succeq 0, \end{aligned} \tag{3.17}$$

where $\mathbf{d}^{ij}$ is the i-th patch of the j-th template $\mathbf{T_j}$, $\hat{\mathbf{Y}}$ represents local patches of the tracking result. Since the template set covers a wide range of target appearance, a non-occluded and consistent local pattern on the target is likely to match its corresponding patch at the same position of at least template. Hence, for a non-occluded patch of a target region, it is likely to use the one within the tracking result to represent the one within a template (i.e., only a few coefficients are large). However, an occluded patch may not match a similar one in the template set due to the drastic difference in appearance and thereby contributes little to the representation of the patch within a template (i.e., the coefficients are small). For each patch within the tracking result, we use the maximum of sparse coding coefficients over all templates. A threshold γ is used to determine whether one local patch within the tracking result can be used to effectively model target appearance in the previous frame.

$$\begin{aligned} &\mathbf{t}_i = \max_j \{\mathbf{s}^{ij}\}, \\ &i = 1, 2, \ldots, N, \;\; j = 0, 1, \ldots, n-1, \end{aligned} \tag{3.18}$$

where $\mathbf{t}_i$ is the maximum response corresponding to the i-th patch of the tracking results. We compute $\mathbf{g}_i$ to determine whether a local patch is occluded or not,

$$\mathbf{g}_i(r_i, c_i) = \begin{cases} 1, & \mathbf{t}_i(i) > \gamma, \\ 0, & \text{otherwise}, \end{cases} \tag{3.19}$$

where r_i and c_i are image coordinates of the pixel in the i-th patch $\mathbf{p}_i$.

The occlusion of patches within the tracking result are accumulated to form a mask via Eq. (3.20) which can be used to describe the occlusion distribution within the whole target object.

$$\mathbf{M}(r, c) = \sum_{\{i|\mathbf{P}(r,c)=\mathbf{p}_i(r_i,c_i)\}} \mathbf{g}_i(r_i, c_i), \tag{3.20}$$

where $\mathbf{P}$ is the image patch of the tracking result, r and c are coordinates on that image patch.

In addition, we use the incremental subspace model (Ross et al. [11]) to represent a target object over a long duration. The tracking result is then reconstructed in terms of the occlusion extent of each local patch, and is represented as a weighted combination of its original appearance and its incremental subspace representation. This not only reduces the risk that the occlusion is updated into the dictionary, but also adapts the model to appearance change of a target object. Finally, to reduce the artifacts of reconstruction by piecing together overlapped patches, the guided image filter (He et al. [2]) is used for smoothing.

Hence, the new template is modeled as

$$\mathbf{T}_{new} = f(\mathbf{P} \odot \mathbf{M} + \mathbf{R} \odot (1 - \mathbf{M}), \mathbf{R}), \tag{3.21}$$

where f denotes the guided image filter, $\odot$ denotes element-wise multiplication, $\mathbf{T}_{new}$ represents the image patch of the new template, $\mathbf{R}$ is the reconstructed image patch using the incremental subspace reconstruction of the tracking result and also acts as guidance image. The new template $\mathbf{T}_{new}$ is then used to update both of the template set and the incremental subspace model.

3.4 Experimental Results and Discussions

In order to evaluate the performance of the proposed trackers, we conduct experiments on the benchmark dataset proposed in [17], which includes 100 challenging image sequences. The overall precision score plots and success rate plots are shown in Fig. 3.6 and the performance in different attributes is shown in Tables 3.1 and 3.2. We can see that ASLA [4] outperforms most of the well-known trackers at the time, meanwhile, WLCS [14] achieves a great result with novel methods.

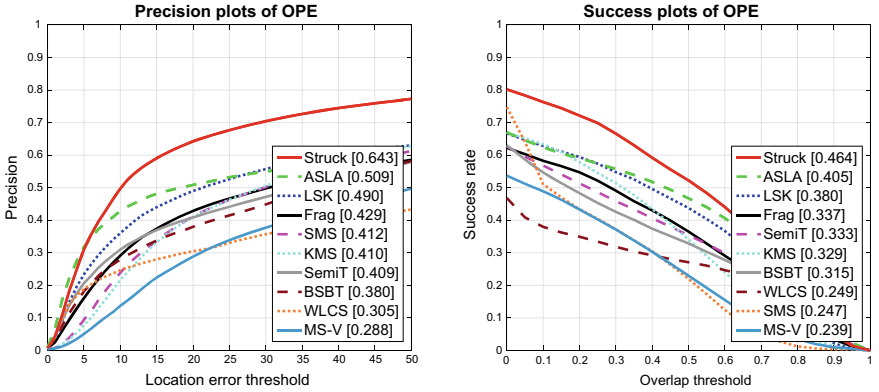


Fig. 3.6 Precision and success plots on OTB-2015 for the proposed algorithms

Table 3.1 The success comparison of mentioned trackers

Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
Struck	***0.440***	***0.383***	***0.455***	***0.451***	***0.428***	**0.347**	***0.459***	***0.391***	***0.425***	***0.374***	***0.411***
ASLA	**0.418**	**0.366**	0.269	**0.399**	**0.430**	*0.350*	0.275	**0.377**	**0.407**	0.297	**0.409**
LSK	*0.332*	*0.338*	**0.357**	*0.381*	*0.368*	*0.300*	**0.347**	*0.368*	*0.383*	**0.334**	*0.358*
Frag	0.298	0.325	0.314	0.323	0.264	0.206	0.325	0.310	0.336	0.302	0.298
SemiT	0.287	0.307	0.311	0.312	0.285	0.279	0.336	0.296	0.306	0.267	0.291
KMS	0.288	0.337	*0.326*	0.311	0.289	0.173	*0.343*	0.310	0.329	*0.322*	0.313
BSBT	0.254	0.266	0.290	0.309	0.290	0.212	0.317	0.281	0.300	0.260	0.269
WLCS	0.251	0.200	0.167	0.212	0.286	0.216	0.176	0.231	0.229	0.179	0.236
SMS	0.179	0.269	0.264	0.236	0.196	0.131	0.250	0.252	0.266	0.266	0.267
MS-V	0.207	0.227	0.254	0.230	0.208	0.107	0.272	0.212	0.243	0.267	0.254

Table 3.2 The precision comparison of mentioned trackers

Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
Struck	***0.584***	***0.527***	***0.609***	***0.633***	***0.559***	***0.628***	***0.592***	***0.532***	***0.587***	***0.487***	***0.610***
ASLA	**0.520**	**0.477**	0.281	**0.506**	**0.532**	**0.516**	0.295	*0.472*	**0.516**	0.348	**0.524**
LSK	*0.422*	0.436	**0.433**	*0.501*	*0.453*	0.422	**0.405**	**0.475**	*0.502*	**0.403**	*0.463*
Frag	0.394	0.404	*0.370*	0.415	0.316	0.307	0.359	0.386	0.432	*0.373*	0.391
SemiT	0.347	0.387	0.356	0.384	0.309	*0.450*	*0.373*	0.372	0.379	0.310	0.365
KMS	0.366	0.429	0.343	0.413	0.335	0.279	0.356	0.375	0.422	0.363	0.403
BSBT	0.305	0.329	0.312	0.383	0.326	0.308	0.343	0.336	0.366	0.306	0.331
WLCS	0.290	0.235	0.177	0.272	0.317	0.261	0.164	0.292	0.293	0.196	0.305
SMS	0.332	*0.447*	0.342	0.389	0.322	0.286	0.329	0.388	0.428	0.343	0.417
MS-V	0.256	0.262	0.262	0.293	0.238	0.186	0.280	0.238	0.300	0.288	0.317

3.5 Summary

In this chapter, we introduce two novel tracking algorithms based on local models. In the first method, the local cosine similarity helps to measure the similarities between the target template and candidates, and an objective function helps to improve the discriminative ability of the WLCS method. The second proposed method exploits both spatial and local information of the target by alignment pooling across the local patches with a spatial layout. Besides, sparse representation is combined with incremental subspace learning for template update. It not only adapts the tracker to account for appearance change of the target but also prevents incorrectly estimated or occluded observations from being put into the template set for update.

References

1. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 798–805 (2006)
2. He, K., Sun, J., Tang, X.: Guided image filtering. In: European Conference on Computer Vision, pp. 1–14 (2010)
3. He, S., Yang, Q., Lau, R.W., Wang, J., Yang, M.H.: Visual tracking via locality sensitive histograms. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2427–2434 (2013)
4. Jia, X., Lu, H., Yang, M.: Visual tracking via adaptive structural local sparse appearance model. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1822–1829 (2012)
5. Kwon, J., Lee, K.M.: Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping Monte Carlo sampling. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1208–1215 (2009)
6. Li, F., Jia, X., Xiang, C., Lu, H.: Visual tracking with structured patch-based model. Image Vis. Comput. **60**, 124–133 (2017)
7. Liu, B., Huang, J., Yang, L., Kulikowsk, C.: Robust tracking using local sparse appearance model and k-selection. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1313–1320 (2011)
8. Liu, B., Yang, L., Huang, J., Meer, P., Gong, L., Kulikowski, C.A.: Robust and fast collaborative tracking with two stage sparse optimization. In: European Conference on Computer Vision, pp. 624–637 (2010)
9. Liu, T., Wang, G., Yang, Q.: Real-time part-based visual tracking via adaptive correlation filters. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 4902–4912 (2015)
10. Riesenhuber, M., Poggio, T.A.: Hierarchical models of object recognition in cortex. Nat. Neurosci. **2**, 1019–1025 (1999)
11. Ross, D.A., Lim, J., Lin, R., Yang, M.: Incremental learning for robust visual tracking. Int. J. Comput Vis. **77**(1–3), 125–141 (2008)
12. Serre, T., Wolf, L., Bileschi, S.M., Riesenhuber, M., Poggio, T.A.: Robust object recognition with cortex-like mechanisms. IEEE Trans. Pattern Anal. Mach. Intell. **29**(3), 411–426 (2007)
13. Sun, C., Wang, D., Lu, H.: Occlusion-aware fragment-based tracking with spatial-temporal consistency. IEEE Trans. Image Process. **25**(8), 3814–3825 (2016)
14. Wang, D., Lu, H., Bo, C.: Visual tracking via weighted local cosine similarity. IEEE Trans. Cybern. **45**(9), 1838–1850 (2015)

15. Wang, F., Zhang, J., Guo, Q., Liu, P., Tu, D.: Robust visual tracking via discriminative structural sparse feature. In: Chinese Conference on Image and Graphics Technologies, pp. 438–446 (2015)
16. Wang, Q., Chen, F., Xu, W., Yang, M.: Online discriminative object tracking with local sparse representation. In: IEEE Workshop on Applications of Computer Vision, pp. 425–432 (2012)
17. Wu, Y., Lim, J., Yang, M.: Object tracking benchmark. IEEE Trans. Pattern Anal. Mach. Intell. **37**(9), 1834–1848 (2015)
18. Xu, Y., Wang, J., Li, H., Li, Y., Miao, Z., Zhang, Y.: Patch-based scale calculation for real-time visual tracking. IEEE Trans. Signal Process. **23**(1), 40–44 (2016)
19. Yang, J., Yu, K., Gong, Y., Huang, T.S.: Linear spatial pyramid matching using sparse coding for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1794–1801 (2009)
20. Yao, R., Shi, Q., Shen, C., Zhang, Y., van den Hengel, A.: Part-based visual tracking with online latent structural learning. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2363–2370 (2013)
21. Zhong, W., Lu, H., Yang, M.: Robust object tracking via sparsity-based collaborative model. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1838–1845 (2012)
22. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. Roy. Stat. Soc. Ser. B (Stat. Methodol.) **67**(2), 301–320 (2005)

Chapter 4
Visual Tracking Based on Model Fusion

Model fusion is an essential strategy to tackle the challenges in visual object tracking. Due to the unpredictable appearance changes of the target and background clusters, a single kind of feature or model cannot fit all situations. In this chapter, we introduce two representative methods based on model fusion. The first one is a traditional method focus on how to combine various handcraft features effectively and the second one is a deep-learning-based method which explores the attention information totally in visual tracking.[1, 2]

4.1 Introduction

The appearance model plays an important role in visual tracking, which is dependent on its adopted feature representation and description model. The adopted feature representation scheme generally adopts features such as color [7], intensity [21], texture information [2], to name a few. Robust feature is the guarantee of a robust tracker. To incorporate some high-level features, multiple feature extraction approaches are necessary, i.e., the standard three-channel RGB color space, LBP descriptors, SIFT descriptors, the color name information [23, 25] or HOG descriptors [1, 6]. Based on these features, feature fusion gains popularity in visual tracking.

Feature fusion focuses on effectively integrating different kinds of features, which can produce robust features against target's appearance changes. Apart from feature

[1] ©Reprinted from Neurocomputing, Vol 207, Huilan Jiang, Jianhua Li, Dong Wang, Huchuan Lu, Multi-feature tracking via adaptive weights, Pages No.189-201, Copyright (2016), with permission from Elsevier.

[2] ©Reprinted from Pattern Recognition, Vol 87, Boyu Chen, Peixia Li, Chong Sun, Dong Wang, Gang Yang, Huchuan Lu, Multi-attention module for visual tracking, Pages No.80-93, Copyright (2019), with permission from Elsevier.

H. Lu and D. Wang, *Online Visual Tracking*,
https://doi.org/10.1007/978-981-13-0469-9_4

fusion, model fusion is also adopted to other aspects such as multi-kernel fusion in trackers [5, 24] based on correlation filters, multi-classifier fusion in ensemble learning [11, 22] and so on. These methods try to utilize different models to deal with different problems. An effective fusion method generally needs workable attention mechanisms, because the unsuitable model would bring bad effects to the final performance if we just simply combine all models with equal weights. Some methods focus to calculate adaptive weights for integration, while others utilize switch mechanisms to select the best model among candidates. All these algorithms aim to pay attention to the most effective part of the whole model.

Recently, deep neural networks (DNNs) have demonstrated remarkable performance and have become the de facto standard methods in visual tracking. Features in higher layers of DNNs encode semantic concepts of object categories and are robust to significant target appearance changes, whereas features in lower layers preserve more spatial details but are sensitive to dramatic appearance changes. Thus, integrating features in both higher and lower layers can maximize their advantages. Video data are known to demonstrate a strong temporal coherence, where the appearance and motion information of the target seldom suffer from significant variations within consecutive frames. Recurrent neural networks (RNNs), which are effective in handling sequential data with correlations, have found limited applications in online visual tracking. In addition, the convolutional neural network (CNN) features pretrained on ImageNet are adopted to distinguish generic objects and treat all channels equally, which is not appropriate for the tracking task. Effective selection is capable of highlighting the target and suppressing response from the background. Another way to achieve this goal is to add spatial attention to features, which is contained in the inter-frame relationship.

4.2 Visual Tracking Based on Adaptive Multi-feature Combination

Here, we introduce a fast and effective tracking algorithm based on the STC tracker [31]. Based on the Bayesian framework, the proposed algorithm obtains a statistical correlation between the region of the tracked target and its local surrounding region in high-level feature spaces. Then, a confidence map is calculated based on the statistical correlation and features of the tracked object. Finally, the location of the target is estimated based on the computed confidence map. The basic idea of obtaining the confidence map is presented as follows:

$$\begin{aligned} c(\mathbf{x}) = \mathrm{P}(\mathbf{x}|o) &= \sum_{m(\mathbf{z})\in X^c} \mathrm{P}(\mathbf{x}, m(\mathbf{z})|o) \\ &= \sum_{m(\mathbf{z})\in X^c} \mathrm{P}(\mathbf{x}|m(\mathbf{z}), o)\mathrm{P}(m(\mathbf{z})|o), \end{aligned} \tag{4.1}$$

where $\mathrm{c}(\mathbf{x})$ denotes the estimated confidence map, $\mathbf{x} \in \mathbb{R}^2$ indicates the coordinate of the tracked object, $\mathbf{z} \in \mathbb{R}^2$ is the coordinates of the surrounding location, and o stands for the presence of an image in the current frame. X^c is the feature set defined by $X^c = \{m(\mathbf{z}) = (F(\mathbf{z}), \mathbf{z}) | \mathbf{z} \in \Omega_c(\mathbf{x}^*)\}$. $F(\mathbf{z})$ denotes the feature vectors at location $\mathbf{z}$, which represent the appearance of the tracked object by using high-level features. $\Omega_c(\mathbf{x}^*)$ stands for the surrounding region of center location $\mathbf{x}^*$ in the current frame.

Equation (4.1) models the relationships among the spatial model, features and confidence map. This equation shows that the joint probability $\mathrm{P}(\mathbf{x}, m(\mathbf{z})|o)$ can be transformed into the product of two conditional probabilities (i.e., $\mathrm{P}(\mathbf{x}|m(\mathbf{z}), o)$ and $\mathrm{P}(m(\mathbf{z})|o)$). We note that the conditional probability $\mathrm{P}(\mathbf{x}|m(\mathbf{z}), o)$ models the statistical correlation and $\mathrm{P}(m(\mathbf{z})|o)$ depicts the prior probability that corresponds to the local surrounding appearance model. The critical and challenging problem is to learn and update the conditional probability $\mathrm{P}(\mathbf{x}|m(\mathbf{z}), o)$.

Appearance Model: The prior probability aims to depict the appearance model, i.e., $\mathrm{P}(m(\mathbf{z})|o)=F(\mathbf{z})$. In this work, $F(\cdot)$ denotes the feature space that represents the appearance of the image frame by using three different features. The appearance model is a collaborative model, which is defined as

$$F_i(\mathbf{z}) = \begin{cases} \mathrm{I}(\mathbf{z}) & i = 1; \\ \mathrm{Cn}(\mathbf{z}) & i = 2; \\ \mathrm{S}(\mathbf{z}) & i = 3. \end{cases} \tag{4.2}$$

In Eq. (4.2), $\mathrm{I}(\mathbf{z})$ demonstrates the image intensity, which is a low-level features, $\mathrm{Cn}(\mathbf{z})$ denotes the color naming feature that aims to describe the color information of both the foreground and the background, and $\mathrm{S}(\mathbf{z})$ is the histograms of gradient (HOG) features which depicts the texture information of the image frame. In this work, the gray, color and texture features are used to calculate three confidence maps, which are then fused into a final confidence map that is used for object tracking.

Spatial Model: The statistical correlation term $\mathrm{P}(\mathbf{x}|m(\mathbf{z}), o)$ is used to build a spatial model, which aims to fully exploit the context information of both object and background and to handle partial occlusions. In this work, the conditional probability function can be defined as $\mathrm{P}(\mathbf{x}|m(\mathbf{z}), o) = \mathrm{h}(\mathbf{x} - \mathbf{z})$, in which $\mathrm{h}(.)$ is a function that represents the relationships between the target's location $\mathbf{x}$ and a local surrounding location $\mathbf{z}$. To obtain an effective confidence map, learning the function $\mathrm{h}(.)$ in an efficient is important. Thus, we rewrite Eq. (4.1) as:

$$\begin{aligned} \mathrm{c}(\mathbf{x}) &= \sum_{m(\mathbf{z}) \in X^c} \mathrm{P}(\mathbf{x}|m(\mathbf{z}), o)\mathrm{P}(m(\mathbf{z})|o) \\ &= \sum_{\mathbf{z} \in \Omega_c(\mathbf{x}^*)} \mathrm{h}(\mathbf{x} - \mathbf{z}) F(\mathbf{z}) \\ &= \mathrm{h}(\mathbf{x}) \otimes F(\mathbf{x}) \end{aligned} , \tag{4.3}$$

where $\otimes$ denotes the convolution operator. On the basis of the theory of signal processing, the computation of the confidence map in Eq. (4.3) can be transformed into the frequency domain, which can be efficiently solved by the fast Fourier transformation (FFT) method:

$$\begin{aligned} \mathscr{F}(c(\mathbf{x})) &= \mathscr{F}(h(\mathbf{x})) \odot \mathscr{F}(F(\mathbf{x})) \\ \mathrm{h}(\mathbf{x}) &= \mathscr{F}^{-1}(\frac{\mathscr{F}(c(\mathbf{x}))}{\mathscr{F}(F(\mathbf{x}))}) \end{aligned}, \tag{4.4}$$

where $\odot$ represents the element-wise product, $\mathscr{F}$ denotes the FFT operator, and $\mathscr{F}^{-1}$ stands for the inverse FFT operator. Thus, the spatial model can be efficiently learned based on Eq. (4.4).

Learning Model: In this work, we adopt a Gaussian-like function to model our expected confidence map for learning the spatial model and facilitating object tracking. The expected function is defined as:

$$\mathrm{c}(\mathbf{x}) = b \times e^{-\left|\frac{\mathbf{x}-\mathbf{x}^*}{\alpha}\right|^{\beta}}, \tag{4.5}$$

where α is a scale parameter, β is a shape parameter and b is a normalization constant. This function takes a value of 1 for the center location and smooth decays to 0 for the surrounding location, which is much smoother than the traditional binary classifier. Thus, we can obtain three spatial models with respect to three different feature channels by using the modeling confidence function (Eq. (4.5)) and FFT as follows:

$$\mathbf{h}_i(\mathbf{x}) = \begin{cases} \mathbf{h}_{gray}(\mathbf{x}) = \mathscr{F}^{-1}(\frac{\mathscr{F}(c(\mathbf{x}))}{\mathscr{F}(\mathrm{I}(\mathbf{x}))}) & i = 1; \\ \mathbf{h}_{cn}(\mathbf{x}) = \mathscr{F}^{-1}(\frac{\mathscr{F}(c(\mathbf{x}))}{\mathscr{F}(\mathrm{Cn}(\mathbf{x}))}) & i = 2; \\ \mathbf{h}_{hog}(\mathbf{x}) = \mathscr{F}^{-1}(\frac{\mathscr{F}(c(\mathbf{x}))}{\mathscr{F}(\mathrm{S}(\mathbf{x}))}) & i = 3. \end{cases} \tag{4.6}$$

Model Fusion

Update Scheme: To capture the appearance changes of both object and background, the representation model needs to be updated in an online fashion. In this work, we update the spatial model by $\mathrm{H}_{t+1}(\mathbf{x}) = (1-\rho)\mathrm{H}_t(\mathbf{x}) + \rho\mathrm{h}(\mathbf{x})$, where ρ is a learning parameter. To maintain an effective representation model, we exploit all the previous frames to update the spatial model, which is simply conducted in the frequency domain:

$$\begin{aligned} e^{jw}\mathrm{H}_w(\mathbf{x}) &= (1-\rho)\,\mathrm{H}_w(\mathbf{x}) + \rho\mathrm{h}_w(\mathbf{x}) \\ \mathrm{H}_w(\mathbf{x}) &= \mathrm{K_w}\mathrm{h}_w(\mathbf{x}) \\ \mathrm{K_w} &= \frac{\rho}{e^{jw} - (1-\rho)} \end{aligned} \tag{4.7}$$

Confidence Maps: To conduct tracking, we calculate confidence maps in different channels for the next frame by using the updated model $\mathrm{H}_{t+1}(\mathbf{x})$ and features of the next frame. The feature set is defined by $X^c_{t+1} = \{m(\mathbf{z}) = (F_{t+1}(\mathbf{z}), \mathbf{z}) | \mathbf{z} \in \Omega_c(\mathbf{x}_t{}^*)\}$.

The three confidence maps are computed as follows:

$$c_{t+1}(\mathbf{x})_{(i)} = \begin{cases} c_{t+1}(\mathbf{x})_{gray}, & i = 1 \\ c_{t+1}(\mathbf{x})_{cn}, & i = 2 \\ c_{t+1}(\mathbf{x})_{hog}, & i = 3 \end{cases}. \tag{4.8}$$

$$c_{t+1}(\mathbf{x})_{(i)} = \mathscr{F}^{-1}(\mathscr{F}(\mathrm{H}_{t+1}(\mathbf{x}))_{(i)} \odot \mathscr{F}(F_{t+1}(\mathbf{x}))_{(i)}) \tag{4.9}$$

Then, a final confidence map $c_{t+1}(\mathbf{x})$ is obtained by incorporating individual confidence maps $c_{t+1}(\mathbf{x})_{(i)}$ with adaptive weights ω_i $(i = 1, 2, 3)$. Finally, the location of the tracked target can be determined based on the largest response in the final confidence map.

Adaptive Weights: Here, we exploit an adaptive learning weight method to fuse different confidence maps for the tracking process. Some papers have already shown the good performance of using adaptive weights for detecting and tracking [26, 29]. First, we initialize the weight vector $\boldsymbol{\omega}$ in the first frame ($\omega_i = \frac{1}{3},\ i = 1, 2, 3$). In the t-th frame, after we obtain the location $\mathbf{x}^*$ of the tracked object, the fusion of different confidence maps can be formulated as the following regression problem:

$$\frac{\sum_{i=1}^{n} \omega_i c(\mathbf{x}_k)}{\sum_{i=1}^{n} \omega_i c(\mathbf{x}^*)} = \psi(\mathbf{x}_k, \mathbf{x}^*) + \varepsilon_k,\ \forall \mathbf{x}_k \in \Omega_c({\mathbf{x}_t}^*) \tag{4.10}$$

$$\begin{aligned} \omega^* &= \arg\min \sum_{k=1}^{M} \eta(\varepsilon_k) \\ s.t. \quad & \sum_{i=1}^{n} \omega_i = 1, \quad \omega_i \geq 0 \end{aligned} \tag{4.11}$$

where M is the number of pixels that belong to the search region for tracking, n denotes the number of channels ($n = 3$ in this work), and ε_k is a slack variable. $\psi(\mathbf{x}, \mathbf{x}^*)$ is considered a template function, which is Gaussian function. $\eta(\varepsilon_k)$ is a cost value function ($\eta(\varepsilon) = \max(0, \varepsilon)$), which ensures that the used slack variable $\varepsilon_k \geq 0$. Thus, we obtain the weight vector that has the minimum cost value in Eq. (4.11).

During the tracking process, we sample N particles randomly around ω_{t-1}, and then choose the best candidate weights to obtain the fusion map. Finally, the fused confidence map $c_{t+1}(\mathbf{x})$ is computed by incorporating individual confidence map $c_{t+1}(\mathbf{x})_{(i)}$ with adaptive weights ω_i, and the target object location is determined by maximizing the confidence map.

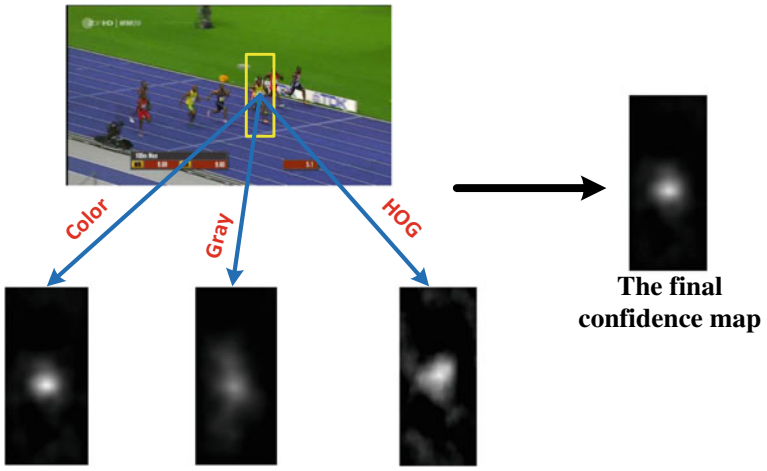


Fig. 4.1 The effectiveness of the weight learning scheme

$$c_{t+1}(\mathbf{x}) = \sum_{i=1}^{n} c_{t+1}(\mathbf{x})_{(i)}\omega_i \tag{4.12}$$

$$\mathbf{x}^*_{t+1} = \arg \max_{\mathbf{x}\in\Omega_c(\mathbf{x}^*_t)} c_{t+1}(\mathbf{x}) \tag{4.13}$$

Figure 4.1 demonstrates an illustration of the proposed adaptive weight scheme, from which we can see that the combined final map is clearer than the individual one. The overall tracking framework in this study is summarized in Table 4.1.

Implementation Details: The parameters of modeling the confidence map are set to $\alpha = 2.25$, $\beta = 1$ in all experiments. The learning parameter of model updating $\rho = 0.075$. The size of the search region is twice that of the tracked object. To reduce the noise effect, we apply a Hamming window in the feature extraction. As a trade-off between effectiveness and speed, we use N = 300 particles for inferencing the optimal combination weights.

4.3 Visual Tracking Based on Multi-attention Module

In the last section, a tracker based on feature fusion of handcraft features is introduced. Now, we move our attention to the deep-learning-based tracking which attempts to develop a multi-attention module to conduct model fusion and capture the appearance variations of the tracked object.

The proposed algorithm consists of four components: a feature extraction network; a multi-attention network that combines temporal, spatial, channel-wise and layer-wise attention; a tracking module for target localization; and an incorporation module.

Table 4.1 Overall flowchart of the proposed tracking method

Input: The t-th image o_t, the fixed rectangle size sz, the center location $\mathbf{x}_{t-1}{}^*$, the modeling confidence map $conf$, the updated model $\mathrm{H}_t(\mathbf{x})$, the parameter ρ and the fusion weight vector $\boldsymbol{\omega}$
1: Cropping out an image patch which is a rectangle region with fixed size sz centered on $\mathbf{x}_{t-1}{}^*$ in the o_t
2: Extracting the HOG feature map S, the color naming feature map Cn and the intensity feature map $\mathbf{I}$ of the image patch
3: Obtaining the three respective response $c_t(\mathbf{x})_{hog}$, $c_t(\mathbf{x})_{cn}$ and $c_t(\mathbf{x})_{gray}$ with the updated models $\mathrm{H}_t(\mathbf{x})_{hog}$, $\mathrm{H}_t(\mathbf{x})_{cn}$, $\mathrm{H}_t(\mathbf{x})_{gray}$ and feature space via $c_t(\mathbf{x}) = \mathscr{F}^{-1}(\mathscr{F}(\mathrm{H}_t(\mathbf{x})) \odot \mathscr{F}(F_t(\mathbf{x})))$
4: The final response map c_t is computed by the fusion of each confidence response via $c_t(\mathbf{x}) = \sum_{i=1}^{n} c_t(\mathbf{x})_{(i)}\omega_i$, and finds the location $\mathbf{x}_t^*$ with the largest response
5: Extracting new maps S, Cn and $\mathbf{I}$ from the image centered by $\mathbf{x}_t^*$
6: Learning the models $\mathrm{h}_{hog}(\mathbf{x})$, $\mathrm{h}_{cn}(\mathbf{x})$ and $\mathrm{h}_{gray}(\mathbf{x})$ with the modeling confidence map $conf$ and new feature space via $\mathrm{h}(\mathbf{x}) = \mathscr{F}^{-1}(\frac{\mathscr{F}(be^{-\left\|\frac{\mathbf{x}-\mathbf{x}^*}{\alpha}\right\|^{\beta}})}{\mathscr{F}(F(\mathbf{x}))})$
7: Updating the respective models $\mathrm{H}_{t+1}(\mathbf{x})_{hog}$, $\mathrm{H}_{t+1}(\mathbf{x})_{cn}$ and $\mathrm{H}_{t+1}(\mathbf{x})_{gray}$ with the parameter ρ for next frame
8: Updating the fusion weight vector $\boldsymbol{\omega}$ (4.11)
Output: The center location $\mathbf{x}_t^*$, the updated model $\mathrm{H}_{t+1}(\mathbf{x})$, and the updated fusion weight vector $\boldsymbol{\omega}$

Figure 4.2 presents the pipeline of our method. Given a new frame containing the target, we feed the whole image into the VGG16 network. Following prior works [20, 27], we adopt feature maps in both higher and lower convolutional layers to characterize the input image. The attention network takes both features as input and outputs for an attentional map as well as a layer-wise selection result. On the basis of the attention map and selected features, the tracking module samples a collection of candidates around the attentive region and computes their posterior probabilities of being the target. The final target state is refined by considering the object detection results.

Here, a multi-attention network is proposed to guide the object tracking, which effectively integrates temporal, spatial, channel-wise and layer-wise attention mechanisms. The detailed structure of the attention module is shown in Fig. 4.3. Given an input frame, we utilize VGG16 to extract the feature of the whole image and pass the features through RoI pooling, obtaining square search regions centered in the last target location. The length of the square search region is two times the maximum of the object's length and width. The concatenation of the cropped feature maps is then fed into the multi-attention network to calculate an attention map and layer-wise scores. In the following, we will describe the attention module in detail.

Temporal Attention: If we treat visual attention in different frames as independent tasks, then, the inference process may be inevitably influenced by input noise, leading to suboptimal results. Motivated by this observation, we treat visual attention in consecutive frames as a sequential problem; and adopt long short term memory(LSTM) [10] units, which are shown as Fig. 4.4 in our attention network to

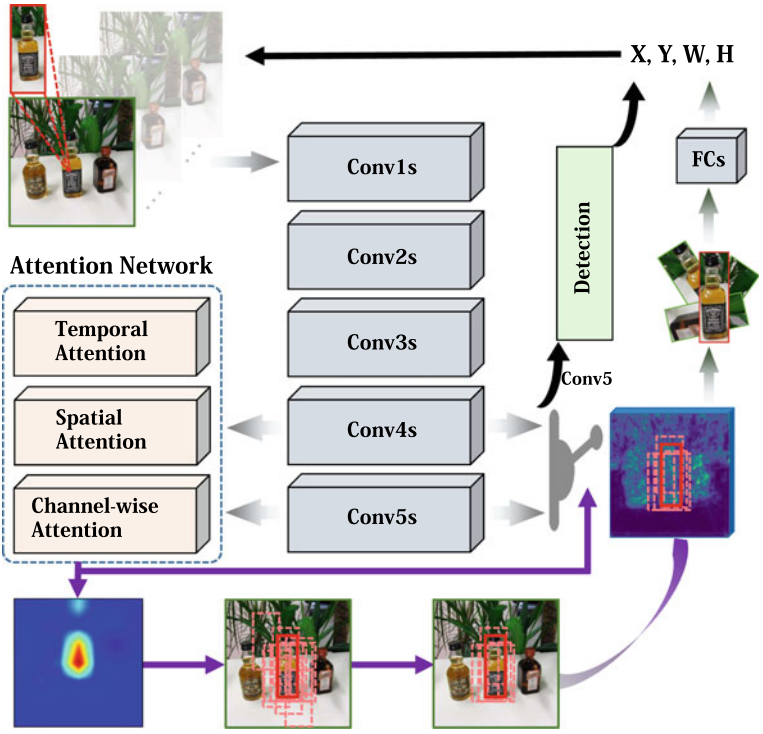


Fig. 4.2 Pipeline of our proposed algorithm. When a new frame appears, we send the whole picture into the feature extraction network, which consists of 16 convolutional layers. The features from Conv4-3 and Conv5-3 are center cropped and sent to the attention network, which will output an attention map and layer-wise scores. The attention map is then applied to guide the candidate sampling and the layer-wise scores are utilized to select the more suitable feature. Then, we utilize the selected feature for target location. The final result is decided by tracking and detection results

enforce temporal consistency. In the t-th time step, the computation in the LSTM is defined by the following equations:

$$\begin{aligned} \begin{pmatrix} f_t \\ i_t \\ o_t \\ g_t \end{pmatrix} &= \sigma \left(W_{hh} h_{t-1} + W_{ih} x_t + b \right), \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t, \\ h_t &= o_t \odot \tanh(c_t), \end{aligned} \tag{4.14}$$

where x_t is the input feature; h_t and c_t are the hidden and cell state, respectively; W_{hh} and W_{ih} are hidden-to-hidden and input-to-hidden kernel weights, respectively; f_t, i_t, and o_t are forget, input and output gate, respectively, and g_t denotes the content gate. The activation function σ for the first three gates adopts sigmoid function; and

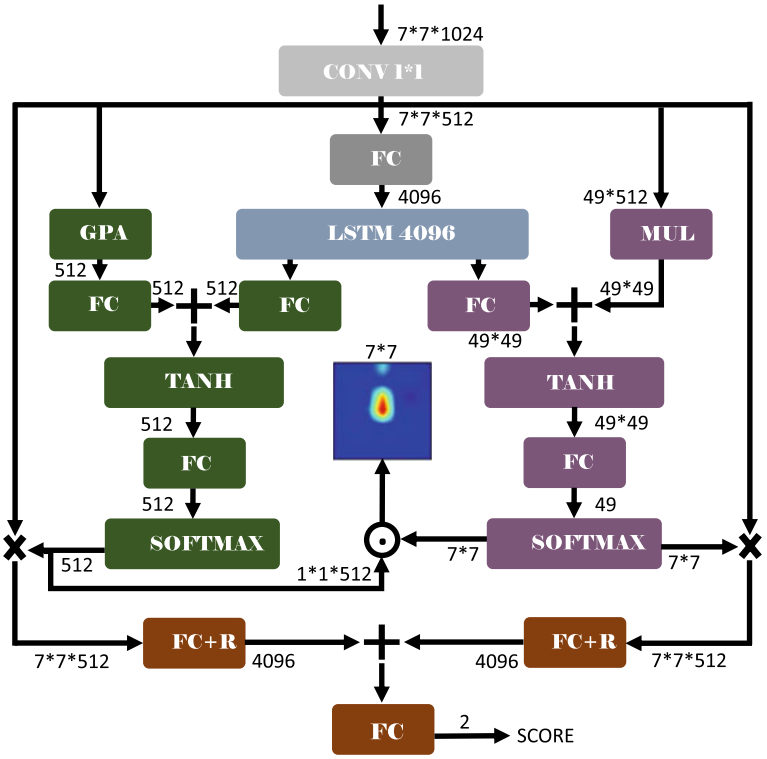


Fig. 4.3 Structure of the attention network. "CONV" means convolutional layer, "FC" means fully connected layer, "GPA" means global average pooling, "MUL" means matrix multiplication, "TANH" means tanh layer, "SOFTMAX" means softmax layers, and "FC+R" means fully connected layer with ReLU unit. The blue rectangles belong to temporal attention. The green rectangles belong to channel-wise attention. The purple rectangles belong to spatial attention. The brown rectangles belong to layer-wise attention

tanh function for the content gate. We treat the hidden state h_t in the t-th time step as the attentional features produced by temporal information, which is also utilized in the next spatial and channel-wise attention.

Spatial Attention: In general, to distinguish the target from the background, only partial regions of the image should be focused. Therefore, applying a global image feature map may lead to tracking drifts due to the interference of superfluous regions. Instead of treating every region of the image equally, spatial attention mechanism gives higher weights for the valuable regions of the task. Here, we utilize the temporal attention feature produced by LSTM to calculate the spatial attention map, encoding the information not only from the current frame but also from previous frames. Given a visual image feature $V \in \mathbb{R}^{W \times H \times C}$ and the hidden state h_t, we first reshape the feature to $V_S = [v_1, v_2, ..., v_n]$ by flattening the width and height of the original V, where $v_i \in \mathbb{R}^C$ and $n = W * H$, representing a spatial region i through a vector v_i.

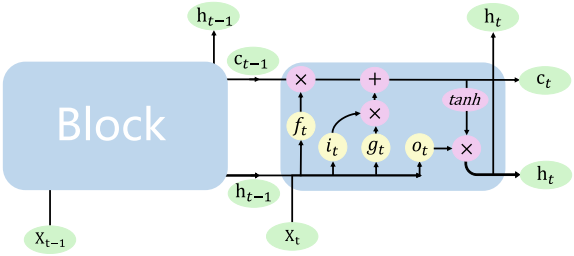


Fig. 4.4 Structure of a block and the association between $t-1$ and t in LSTM

We feed them through a single-layer neural network followed by a softmax function to generate the attention weight for the spatial attention over the $W * H$ regions:

$$\begin{aligned} A_a &= tanh\left((W_1 V_S + b_1) \oplus W_2 h_t\right) \\ A_s &= softmax(W_3 A_a + b_3) \end{aligned} \tag{4.15}$$

where $W_1 \in \mathbb{R}^{n \times C}$, $W_2 \in \mathbb{R}^{n \times d}$, $W_3 \in \mathbb{R}^n$ are weight parameters that are used to ensure a balance between features and hidden state dimensions. $b_1 \in \mathbb{R}^n$, $b_3 \in \mathbb{R}^1$ are model biases and $\oplus$ represents the addition of a matrix and a vector which is performed by adding each column of the matrix with the vector.

Channel-Wise Attention: As mentioned in [20], CNN filters can be seen as pattern detectors. Some are sensitive to color information, whereas others may be strongly responsive around object edge. We aim to select certain filters that fit well with the current tracking. To achieve this, we introduce a channel-wise attention mechanism to dynamically choose effective channels, given that each channel of the CNN features is exactly a response activation of the corresponding convolution filter. Similar to spatial attention, we utilize the temporal attention feature produced by LSTM. Given a visual image feature $V \in \mathbb{R}^{W \times H \times C}$ and the hidden state h_t, we first apply the global average pooling on the visual image feature V to obtain the channel feature $v^{'}$.

$$V_C = [v_1^{'}, v_2^{'}, ..., v_c^{'}], v^{'} \in \mathbb{R}^C \tag{4.16}$$

where $v_i^{'}$ is the output of the global average pooling on the 2D feature of the ith channel. A single-layer neural network followed by a softmax function is applied in the channel-wise attention.

$$\begin{aligned} A_b &= tanh\left(\left(W_1^{'} \otimes V_C + b_1^{'}\right) \oplus W_2^{'} h_{t-1}\right) \\ A_c &= softmax\left(W_3^{'} A_b + b_3^{'}\right) \end{aligned} \tag{4.17}$$

Similar to the spatial attention part, $W_1^{'} \in \mathbb{R}^{C\times C}$, $W_2^{'} \in \mathbb{R}^{C\times d}$, $W_3^{'} \in \mathbb{R}^{C}$ are weight parameters to achieve a balance between feature and hidden state dimensions. $b_1^{'} \in \mathbb{R}^{C}$, $b_3^{'} \in \mathbb{R}^{1}$ are model biases and $\otimes$ represents the outer product of vectors.

Layer-Wise Attention: Prior works [20, 27] suggest that deep features in different levels are suitable for handling different types of scenarios. However, it is nonscalable to fully analyze their characteristics and manually design practical rules to determine optimal features on the fly. Therefore, we propose a layer-wise attention model for a dynamic switch to the optimal features.

Given the spatial weight A_s, channel-wise weight A_c and features visual V, the feature F_s with spatial attention and F_c with channel-wise attention can be calculated by the following formula:

$$F_s = f_s(A_s, V), F_c = f_c(A_c, V) \tag{4.18}$$

where $f_s(.)$ is an element-wise multiplication between the spatial weight and the feature of each channel after reshaping $A_s \in \mathbb{R}^{W*H}$ to $A_s^{'} \in \mathbb{R}^{W\times H}$, while $f_c(.)$ represents the application of the weights to each channel. Then, we feed these features to independent fully connected layers followed by ReLU units. Finally, we combine two kinds of features and utilize a classifier to output the scores of input features, which also comprises a single fully connected layer followed by a softmax function. Notably, our selection method is applicable to any number of features, although we focus only on two different types of features, i.e., Conv5-3 and Conv4-3.

For each of the Conv4-3 and Conv5-3 features, we exploit one convolutional layer and three fully connected layers for target classification. The convolutional layer is used for dimension reduction, with a kernel size of 1×1 and produces a feature map with 256 channels. The next two fully connected layers have 512 output units and are combined with ReLUs. The last fully connected layer outputs two response values, which are normalized by the Softmax layer to obtain the foreground and background probabilities for different candidates.

Tracking with Multi-attention Module: We perform tracking under the tracking-by-detection framework.

Candidate Feature Extraction: Given the input image I^t at the t-th frame and the target location X^{t-1} in the last frame, we sample a collection of target candidates $\{X_i^t\}$ according to the Gaussian distribution $p(X^t|X^{t-1}) = \mathcal{N}(X^t; X^{t-1}, \Sigma)$, centered at X^{t-1} with covariance Σ. To identify the target location, one needs to characterize each candidate X_i with a representation f_i, and measures its probability of being the target by $p(X_i; f_i, \theta)$, where θ denotes the parameters in modeling the probability.

Instead of extracting the deep feature independently for each candidate, we feed forward the whole input image through the feature extraction network to obtain the feature map that is shared by all target candidates. RoI pooling is then performed on the feature map, which projects the window position of each candidate X_i of the original image onto the feature map and aggregates the response values of the feature map in the projected window into a fixed size ($7 \times 7 \times 512$) feature representation f_i.

As a result, only one forward propagation of the deep feature extractor suffices to generate feature representations of all the target candidates.

Guidance with Attention Map: To further reduce the number of candidates, we propose the attention map guidance method which is described as follows.

With the spatial weight $A_s \in \mathbb{R}^{W*H}$, channel-wise weight $A_c \in \mathbb{R}^C$ and visual features $V \in \mathbb{R}^{W\times H\times C}$, we obtain the attention map F_a via:

$$F_a = \frac{1}{C}\sum^{C} f_c(A_c, f_s(A_s, V)) \tag{4.19}$$

where f_s and $f_c(.)$ represent the same meaning as mentioned in Eq. (4.18). After that, all the candidates are sorted in descending order by the sum of attention score in the corresponding box and the top K candidates are kept. We choose the feature from the layer with a higher score, which is calculated according to layer-wise attention as mentioned. Then, we crop candidate features on that basis according to the candidate parameters and apply them for the next evaluation.

Target Location: Two subnetworks are designed for different layers, which share the same network architecture with different network parameters. We send the above candidate features to the corresponding subnetwork to obtain their probability of being the target and the one with the highest foreground probability is predicted as the target.

Initialization and Update: Given the initial location X^1 of the target in the first frame, we generate a positive and a negative training set $\mathscr{X}_+$ and $\mathscr{X}_-$, respectively. The above two subnetworks are initialized by minimizing the following objective function with Stochastic Gradient Descent (SGD):

$$\arg\min_{\theta} -\sum_{X\in\mathscr{X}_+} p_+(X|f;\theta) - \sum_{X\in\mathscr{X}_-} p_-(X|f;\theta). \tag{4.20}$$

To adapt the tracker to target appearance changes, we finetune the tracking modules in each frame based on the same method as the initialization and the same objective function as mentioned in Eq. (4.20) using the SGD method.

Incorporation with Detection Results: Object detection in static images is correlated to but highly different from online visual tracking in that the former aims to locate the object of predefined categories and does not differentiate instances of the same category. Therefore, applying object detectors to visual tracking is nontrivial. In this section, we explore three simple but effective strategies as potential solutions to this problem. We adopt the Faster-RCNN detector trained on the PASCAL-VOC VOC 2007 dataset with 20 categories. Given both the detector and the tracker rely on the VGG network for feature extraction, with shared deep features the additional computational overhead of introducing the detector to our tracking system is minimal.

The training data of the detector may not cover the target to track at test time. Consequently, whether the detection results are suitable for the current tracking task needs to be determined first. To this end, we compute the overlaps between the detected object windows and the ground truth target window in the first frame. If the highest overlap value is less than a threshold τ_1, which indicates that the detector fails to capture the target category, then we perform tracking without considering the detection results. Otherwise, we leverage detection results in the following three ways:

In our preliminary experiments, we observe that the off-line pretrained detector often delivers more accurate prediction than the online initialized tracker in handling scale variations of the target. However, its performance is not consistently reliable. Therefore, we primarily use the tracking results and refine the target location by using the detection results. To alleviate detection noise, we measure the overlap between the tracking result and all the detected windows. If the highest overlap is higher than a predefined threshold τ_2, then we select the detected window with the highest overlap as the final target location.

In each frame, the detector proposes a set of potential objects, which has a higher risk of confusing the tracker and leading to drifts. To further improve the discriminative power of the tracker, we treat all the detected object windows that do not overlap with the predicted target window as hard negatives and add them to the training sample set to update the tracker in the current frame.

Many challenging factors (e.g., severe occlusion, dramatic appearance variation, and out of view) may easily lead to tracking drifts. Recapturing the target after tracking failures is difficult for a tracker. In our case, this situation is mainly caused by the motion model, which restricts the candidate sampling to be performed near the last target location only. To address this issue, we augment the target candidate set by adding all the detected windows from the same category as the target after tracking failures.

Implementation Details: The first layer of the attention model is a convolutional layer with a kernel size of 1×1 and produces the visual feature map V of 512 channels. W and H in spatial and channel-wise attention are both set to 7. The LSTM hidden state dimension is set to 4096. The fully connected layers that combine with the ReLU and dropout layers after the F_s and F_c have 4096 units.

For tracking modules, one convolutional layer and three fully connected layers exist. The convolutional layer is for the purpose of dimension reduction, with a kernel size of 1×1 and produces a feature map with 256 channels. The next two fully connected layers have 512 output units and are combined with ReLUs. The last fully connected layer outputs two response values, which are normalized by the Softmax layer to obtain the foreground and background probabilities.

To estimate the target in each frame, we draw N(= 256) samples $X_t^i = \left(x_t^i, y_t^i, s_t^i\right), i = 1, \ldots, N$ whose translation and scale generate from a Gaussian distribution. The mean is the previous target state $X_{t-1}^{\star}$ and covariance is a diagonal matrix $diag\left(0.09l^2, 0.09l^2, 0.25\right)$, where l is the mean of the width and height of the target in the previous frame. The scale of each candidate is determined by multiplying

$1.05^{s_t^i}$ to the previous target scale. The attention map guidance keeps two-thirds of the candidates.

We use 67 video sequences from TC-128 [18] in which videos that overlap with OTB and VOT are excluded. Correspondingly, we trained the attention network using videos from OTB-2015 by excluding overlapping videos from TC-128.

To collect training samples, we run the two trackers based on the Conv5-3 and Conv4-3 features, on all the training video clips. In each frame, we measure the accuracy of the two trackers by the overlaps between the predicted and the ground truth target windows. The feature with higher accuracy is labeled as the optimal one for the current frame. We then store the concatenated feature maps of the search region and the label indicating the optimal feature, which together serves as a training sample pair for the attention network. To prevent tracking drifts when collecting training samples, we update the two trackers using the ground truth target locations.

At training stage, each time we randomly sample a sequence of $T = 10$ consecutive frames. The stored feature maps of these frames are sequentially fed into the attention network to produce the scores of the input features. The disagreement between the predicted scores and the ground truth labels is measured by the cross-entropy loss in each frame, which is then back-propagated through time to calculate gradients and minimized by SGD. At test time, we apply the attention network to sequences of arbitrary length. Although the number of total time steps at training and test time is inconsistent, we find it works well in practice. Notably, our selection method is applicable to any number of features, although we focus on only two different types of features, i.e., Conv5-3 and Conv4-3.

For the initialization of two tracking modules, we collect 500 positive samples and 5,000 negative samples which have ≥ 0.7 and ≤ 0.3 IoU overlap ratios with ground truth in the first frame. Similarly, for network updating, we collect 50 positive and 200 negative samples with ≥ 0.7 and ≤ 0.3 overlap with the predicted position of the target generally. Specially, for online hard negative mining, we collect 100 hard negative samples for every hard negative target, and add them into the negative samples. The number of positive samples is one-third the number of negative samples.

For attention network learning, we set the learning rate as 1e-3 at first and every 50,000 iterations down to the half of the original. The network converges at 150,000 iterations. For the tracking modules, we initialize the networks in the first frame for 30 iterations with a learning rate of 0.01. We update the network for one iteration with a learning rate of 0.003 in the following each frame. The weight decay and momentum parameters are fixed as 0.0005 and 0.5, respectively. We update the network when the foreground probability of the candidate is larger than 0.

For online tracking, we utilize the VGG-net finetuned on the PASCAL-VOC VOC 2007 dataset. τ_1 in the first frame and τ_2 for tracking results refinement are set to 0.5 and 0.7, respectively. When the predicted target location is out of the input image and the highest foreground probability of all candidates is less than 0, we consider it a tracking failure, and re-initialize the tracker as described in Sect. 4.3. All parameters are fixed throughout our experiment.

4.4 Experimental Results and Discussions

To evaluate the performance of the proposed trackers, we conduct experiments on the benchmark dataset proposed in [28], which includes 100 challenging image sequences. We compared our algorithms (MTAW [14] and MAM[4]) with 13 trackers including the top ten trackers (STRUCK [11], TLD [15], SCM [32], CXT [8], CSK [12], VTD [16], ASLA [13], VTS [17], OAB [9] and LSK [19]) on OTB-2015 dataset and three baseline trackers (MEEM [30], SiameFC [3] and FCNT [27]). The precision score plots and success rate plots are utilized to evaluate the tracker's performance.

Figure 4.5 shows the overall comparison of our algorithms with 13 other trackers. Tables 4.2 and 4.3 show the performance of mentioned trackers of different challenge factors. The traditional method MTAW gets a top performance among other trackers which are based on handcraft features. The comparison illustrates the effectiveness of feature fusion in MTAW. The other algorithm, named by multi-attention module (MAM), outperforms among state-of-the-art trackers significantly in both measures. These outstanding results are partly attributed to the strength of CNN features, which have a much better ability to describe a target than low-level handcrafted features. Comparing with FCNT [27], which also utilizes multilevel deep features, our algorithm performs much better due to the accurate feature selection through our attention network. Instead of integrating multiple features from different layers, we intelligently select the most suitable feature from a certain layer, suppressing noise interference and improving the tracking results.

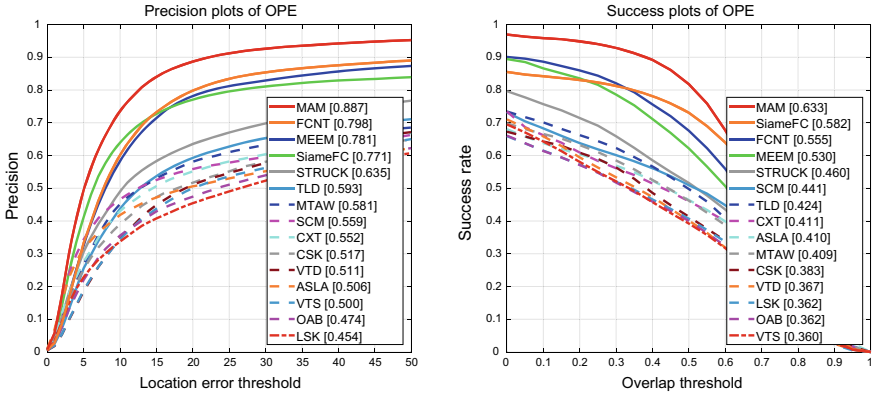


Fig. 4.5 Precision and success plots on OTB-2015 for the proposed algorithms

Table 4.2 The success comparison of mentioned trackers

Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
MAM	***0.677***	***0.600***	***0.622***	***0.625***	***0.666***	***0.596***	***0.634***	***0.592***	***0.621***	***0.613***	***0.620***
FCNT	**0.529**	**0.529**	*0.526*	**0.558**	*0.543*	*0.442*	*0.555*	*0.515*	*0.549*	0.470	*0.506*
MEEM	0.519	0.489	0.525	0.529	0.517	0.355	0.545	0.504	0.525	*0.488*	0.474
SiameFC	*0.523*	*0.506*	**0.569**	*0.557*	**0.568**	**0.573**	**0.568**	**0.543**	**0.558**	**0.506**	**0.557**
STRUCK	0.428	0.383	0.452	0.448	0.420	0.347	0.454	0.388	0.424	0.363	0.404
TLD	0.354	0.341	0.421	0.427	0.414	0.372	0.430	0.365	0.390	0.343	0.389
MTAW	0.434	0.374	0.368	0.446	0.436	0.270	0.381	0.375	0.401	0.283	0.334
SCM	0.457	0.381	0.311	0.412	0.486	0.347	0.320	0.420	0.421	0.324	0.432
CXT	0.352	0.297	0.403	0.442	0.376	0.366	0.403	0.332	0.391	0.325	0.382
CSK	0.416	0.338	0.331	0.379	0.368	0.251	0.322	0.331	0.355	0.262	0.326
VTD	0.396	0.341	0.280	0.391	0.379	0.235	0.273	0.356	0.398	0.336	0.360
ASLA	0.432	0.362	0.291	0.393	0.433	0.341	0.282	0.367	0.411	0.321	0.408
VTS	0.383	0.339	0.277	0.376	0.374	0.203	0.258	0.340	0.388	0.319	0.351
OAB	0.313	0.314	0.364	0.354	0.320	0.266	0.375	0.334	0.340	0.302	0.335
LSK	0.353	0.318	0.312	0.363	0.364	0.300	0.274	0.353	0.369	0.327	0.351

Table 4.3 The precision comparison of mentioned trackers

Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
MAM	***0.933***	***0.847***	***0.838***	***0.891***	***0.907***	***0.894***	***0.850***	***0.831***	***0.892***	***0.860***	***0.869***
FCNT	**0.750**	**0.760**	0.708	**0.830**	**0.777**	*0.755*	**0.726**	*0.731*	**0.800**	0.629	**0.763**
MEEM	*0.746*	*0.754*	*0.728*	*0.794*	*0.740*	0.605	0.722	**0.741**	*0.794*	**0.685**	*0.740*
SiameFC	0.690	0.690	**0.741**	0.742	0.736	**0.815**	*0.724*	0.722	0.756	*0.669*	0.739
STRUCK	0.550	0.527	0.603	0.628	0.549	0.628	0.583	0.527	0.593	0.468	0.599
TLD	0.459	0.484	0.550	0.606	0.559	0.552	0.541	0.527	0.570	0.463	0.566
MTAW	0.629	0.521	0.477	0.636	0.625	0.449	0.487	0.529	0.582	0.342	0.512
SCM	0.579	0.512	0.320	0.533	0.605	0.484	0.322	0.541	0.554	0.409	0.550
CXT	0.444	0.394	0.544	0.599	0.500	0.492	0.561	0.433	0.524	0.388	0.525
CSK	0.577	0.453	0.400	0.512	0.482	0.367	0.379	0.428	0.484	0.283	0.455
VTD	0.536	0.461	0.332	0.559	0.504	0.378	0.304	0.482	0.574	0.398	0.526
ASLA	0.531	0.453	0.307	0.495	0.524	0.489	0.284	0.434	0.515	0.384	0.515
VTS	0.512	0.462	0.326	0.534	0.502	0.347	0.281	0.448	0.558	0.360	0.513
OAB	0.410	0.410	0.444	0.461	0.413	0.380	0.443	0.426	0.452	0.336	0.455
LSK	0.444	0.411	0.350	0.476	0.446	0.422	0.274	0.454	0.483	0.391	0.447

4.5 Summary

It is hard to design a single model which is suitable for all challenges in visual tracking. In this part, we proposed two representative trackers (MTAW and MAM) based on model fusion and attention module. Extensive experimental results on OTB-2015 illustrate that the proposed methods could improve the tracking performance for both traditional and deep-learning-based methods. The improvements in success and accuracy scores mainly rely on the comprehensive property from various models and effective attention mechanisms.

References

1. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 798–805 (2006)
2. Avidan, S.: Ensemble tracking. IEEE Trans. Pattern Anal. Mach. Intell. **29**(2), 261–271 (2007)
3. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: European Conference on Computer Vision Workshop, pp. 850–865 (2016)
4. Chen, B., Li, P., Sun, C., Wang, D., Yang, G., Lu, H.: Multi attention module for visual tracking. Pattern Recognit. **87**, 80–93 (2019)
5. Choi, J., Chang, H.J., Jeong, J., Demiris, Y., Choi, J.Y.: Visual tracking using attention-modulated disintegration and integration. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 4321–4330 (2016)
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 886–893 (2005)
7. Danelljan, M., Shahbaz Khan, F., Felsberg, M., Van de Weijer, J.: Adaptive color attributes for real-time visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1090–1097 (2014)
8. Dinh, T.B., Vo, N., Medioni, G.G.: Context tracker: exploring supporters and distracters in unconstrained environments. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1177–1184 (2011)
9. Grabner, H., Bischof, H.: On-line boosting and vision. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 260–267 (2006)
10. Graves, A.: Long Short-Term Memory. Springer, Berlin, Heidelberg, pp. 1735–1780 (2012)
11. Hare, S., Saffari, A., Torr, P.H.S.: Struck: Structured output tracking with kernels. In: IEEE International Conference on Computer Vision, pp. 263–270 (2011)
12. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: Exploiting the circulant structure of tracking-by-detection with kernels. In: European Conference on Computer Vision, pp. 702–715 (2012)
13. Jia, X., Lu, H., Yang, M.: Visual tracking via adaptive structural local sparse appearance model. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1822–1829 (2012)
14. Jiang, H., Li, J., Wang, D., Lu, H.: Multi-feature tracking via adaptive weights. Neurocomputing **207**, 189–201 (2016)
15. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1409–1422 (2012)
16. Kwon, J., Lee, K.M.: Visual tracking decomposition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1269–1276 (2010)
17. Kwon, J., Lee, K.M.: Tracking by sampling trackers. In: IEEE International Conference on Computer Vision, pp. 1195–1202 (2011)

18. Liang, P., Blasch, E., Ling, H.: Encoding color information for visual tracking: algorithms and benchmark. IEEE Trans. Image Process. **24**(12), 5630–5644 (2015)
19. Liu, B., Huang, J., Kulikowski, C.A., Yang, L.: Robust visual tracking using local sparse appearance model and k-selection. IEEE Trans. Pattern Anal. Mach. Intell. **35**(12), 2968–2981 (2013)
20. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: IEEE International Conference on Computer Vision, pp. 3074–3082 (2015)
21. Ross, D., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. Int. J. Comput. Vis. **77**(1–3), 125–141 (2008)
22. Saffari, A., Leistner, C., Godec, M., Bischof, H.: Robust multi-view boosting with priors. In: European Conference on Computer Vision, pp. 776–789 (2010)
23. Shahbaz Khan, F., Anwer, R.M., van de Weijer, J., Bagdanov, A.D., Vanrell, M., Lopez, A.M.: Color attributes for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 3306–3313 (2012)
24. Tang, M., Feng, J.: Multi-kernel correlation filter for visual tracking. In: IEEE International Conference on Computer Vision, pp. 3038–3046 (2015)
25. Van De Weijer, J., Schmid, C., Verbeek, J., Larlus, D.: Learning color names for real-world applications. Image Process. IEEE Trans. **18**(7), 1512–1523 (2009)
26. Wang, D., Lu, H., Chen, Y.W.: Object tracking by multi-cues spatial pyramid matching. In: IEEE International Conference on Image Processing, pp. 3957–3960 (2010)
27. Wang, L., Ouyang, W., Wang, X., Lu, H.: Visual tracking with fully convolutional networks. In: IEEE International Conference on Computer Vision, pp. 3119–3127 (2015)
28. Wu, Y., Lim, J., Yang, M.: Object tracking benchmark. IEEE Trans. Pattern Anal. Mach. Intell. **37**(9), 1834–1848 (2015)
29. Yang, M., Lv, F., Xu, W., Gong, Y.: adaptive multi-cue integration for multiple human tracking. In: IEEE International Conference on Computer Vision, pp. 1554–1561 (2009)
30. Zhang, J., Ma, S., Sclaroff, S.: MEEM: robust tracking via multiple experts using entropy minimization. In: European Conference on Computer Vision, pp. 188–203 (2014)
31. Zhang, K., Zhang, L., Liu, Q., Zhang, D., Yang, M.H.: Fast visual tracking via dense spatio-temporal context learning. In: European Conference on Computer Vision, pp. 127–141 (2014)
32. Zhong, W., Lu, H., Yang, M.H.: Robust object tracking via sparsity-based collaborative model. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1838–1845 (2012)

Chapter 5
Tracking by Segmentation

Current efforts of single-target tracking mainly focus on building robust bounding box-based trackers to effectively discriminate the target from the background. However, these trackers describe the targets from a global view and are not robust to cope with challenging situations like nonrigid appearance change or partial occlusion. In this chapter, we introduce three segmentation-based tracking algorithms aimed at nonrigid object tracking, which is a more challenging task because this kind of tracking requires obtaining accurate target-background separations rather than coarse bounding boxes.[1, 2, 3]

5.1 Introduction

As traditional segmentation methods are slow, often requiring several seconds per frame, directly applying them into tracking algorithms leads to inferior performance. To obtain an acceptable speed, BCD does segmentation based on the proposed color feature mining manner which exploits log color feature distribution ratio (LCFDR). To adapt to online variations, a novel online gentle boost (OGB) algorithm is further developed. It aims to minimize log-likelihood function and absorbs the weight of a sample into weak classifier. This strategy avoids a complicated mechanism for evaluating the importance weight of a sample and thereby speeding up learning processing. It treats training samples as independent examples rather than some instances, which benefit some particular applications (such as pixel-based classification). By combining OGB algorithm and log color features, BCD proposes a fast yet effective color-based object tracking algorithm. For a new given frame, the pixels within a

[1] ©2014 IEEE, Reprinted, with permission, from Ref. [30].

[2] ©Reprinted by permission from Springer Nature: [Springer] [Pattern Analysis and Applications] [Fast and effective color-based object tracking by boosted color distribution, Dong Wang, Huchuan Lu, Ziyang Xiao, Yen Wei Chen, 2013].

[3] ©2011 IEEE, Reprinted, with permission, from Ref. [27].

H. Lu and D. Wang, *Online Visual Tracking*,
https://doi.org/10.1007/978-981-13-0469-9_5

search window are tested by the classifier and then a probability map is built, where the value of probability map at a given pixel is in direct proportion to the probability of classifying this pixel into the object of interest. Based on the probability map created by the classifier, the objects accurate position is given by the mean shift method. Extensive results show that BCD achieves competitive tracking and segmentation results while maintaining a speed of 6–10 frames per second.

Numerous trackers have exploited low-level cues for visual tracking. Such cues are effective for feature tracking and scene analysis, but are not efficient in the context of object tracking. As mid-level visual cues have achieved great success in numerous vision problems including object segmentation, recognition, and pose estimation, SPT introduces mid-level features to develop adaptive appearance model to account for large appearance variation in tracking. It presents a discriminative appearance model based on superpixels, thereby facilitating a tracker to distinguish the target and the background by segmentation results. The tracking task is then formulated by computing a target-background confidence map, and obtaining the best candidate by maximum a posterior estimate. During the training stage, the segmented superpixels are grouped for constructing a discriminative appearance model to distinguish foreground objects from cluttered backgrounds. In the test phase, a confidence map at superpixel level is computed using the appearance model to obtain the most likely target location with maximum a posteriori (MAP) estimates. The appearance model is constantly updated to account for variation caused by change in both the target and the background. It also includes a mechanism to detect and handle occlusion in the proposed tracking algorithm for adaptively updating the appearance model without introducing noise. Experimental results on various sequences show that the proposed algorithm performs favorably against traditional state-of-the-art methods.

In recent years, the tracking algorithms based on deep learning show great performance in many challenges, this mainly depends on the powerful ability of neural networks on feature extraction and image classification. These methods are suitable for the bounding box-based object tracking, but can not produce the segmentation-based outputs for non-rigid object tracking. Being inspired by saliency detection, which produces pixel-wise segmentation outputs that distinguish the objects of interest from its surrounding background, TFCN proposes a novel nonrigid object tracking method based on spatial-temporal consistent discriminative saliency detection. The proposed method can extract the accurate regions of the tracked target as tracking output, which achieves better description of the nonrigid objects while reduces background pollution to the target model. More specifically, it first develops a tailored fully convolutional neural network (TFCN), which is pretrained on a well-constructed saliency detection dataset to predict the saliency map for a given image region. Then, the proposed TFCN model utilizes local image regions with various scales and spatial configurations as inputs, resulting in multiple local saliency maps. Based on the weighted entropy method, these local saliency maps are effectively fused to produce a final discriminative saliency map for online tracking. In addition, a structural output target-background classifier is built on the accumulated discriminative saliency maps, which effectively utilize the spatial-temporal information to

generate pixel-wise outputs for depicting the state of the tracked object. Finally, the proposed algorithm extracts the regions of interest (ROIs) and fine-tune the TFCN to obtain the local saliency map in the next frame.

Segmentation-based trackers mentioned above will be introduced in details in the following sections.

5.2 Color-Based Object Tracking Based on Boosted Color Distribution

To investigate effective nonrigid tracking algorithm, we initially develop a boosted color distribution model using pixel-wise classification information.

Tracking with Boosted Color Distribution: To begin with, we briefly review the overall tracking framework based on pixel-wise classification, as illustrated in Fig. 5.1. Generally, a tracking system requires three components: image representation, appearance model, and motion model.

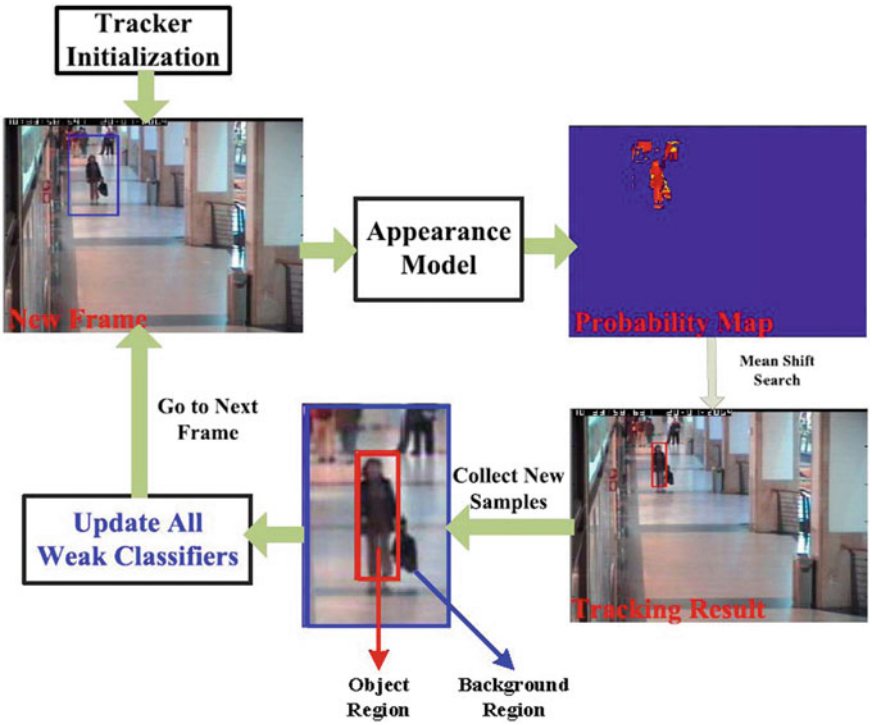


Fig. 5.1 Flowchart of our pixel-wise nonrigid tracking system

Fig. 5.2 Motion model in our tracking system

First, image representation contains two aspects, that is, features (e.g., color, texture, and Haar-like feature) and the fashion of using them (e.g., histogram). Our image representation consists of a set of linearized RGB color space and their histogram representation.

Second, the appearance model is composed of a discriminative classifier, the output of which is a probability ($P\ (y = 1|x)$) rather than a binary value, where x represents an individual pixel and y is a binary value indicating the pixel belonging to the object of interest ($y = 1$) or its surrounding background ($y = 0$). Using the discriminative classifier at a given frame, every pixel is mapped to a probability value, and a probability map is then built. The peak of the probability map convolved with an Epanechnikov kernel [5], which is related to the mean shift kernel, is selected as the center of the moving object.

Finally, our motion model is a simple search window (Fig. 5.2), where the location of object of interest (l_t) at time t is equally likely to appear within a bounding box (also called search window ($s\ (l_{t-1})$) centered by the location of object at time $t - 1$.

Online Gentle Boost (OGB): Since Avidan [2] introduced the idea of ensemble learning into the tracking field, many researchers [3, 10, 11, 16, 21] have devoted to developing online versions of AdaBoost algorithms for achieving robust tracking.

The most influential work, online boosting method (also denoted as online AdaBoost (OAB)), is proposed by Grabner et al. [10]. The essential idea is to apply online boosting not directly to the weak classifiers but to the selectors. The importance of a sample can be estimated by propagating it through a series of selectors consisting of a set of weak learners. However, the imperfection of OAB is that it requires a complicated mechanism for evaluating the importance weight of a sample. As discussed in [8], the boosting algorithm can be considered as a gradient decent algorithm in function space, which attempts to minimize a specific loss function. Motivated by [3], we adopt the negative log-likelihood function as our loss function, because it helps avoid the evaluation of weights of samples and provides a probability interpretation easily. The following is our OGB algorithm (see Table 5.1).

Table 5.1 OGB algorithm

OGB Algorithm
Input: Newly added data $\{x_i, y_i\}_{i=1}^{N}$, where $y_i \in \{0,1\}$
Goal: Select no more than K weak classifiers from M weak classifier pool for building a strong classifier to label pixels that belong to an object or background $(M > K)$
1: Update all M weak classifiers in the pool with new data $\{x_i, y_i\}$
2: Initialize $L = +\infty$ and $H_i = 0$ for all x_i (H_i: cumulative constant)
3: **for** k = 1 to K **do**
4: **for** m = 1 to M **do**
5: $p_i^m = \sigma(H_i + h_m(x_i))$
6: $L^m = -\sum_i (y_i \log(p_i^m) + (1 - y_i)\log(1 - p_i^m))$
7: **end for**
8: $m^* = \arg\min_m L^m$
9: **if** $L^{m*} > L$
10: break;
11: **end if**
12: $L = L^{m*}$
13: $h_k(x) = h_{m^*}(x)$
14: $H_i = H_i + h_k(x_i)$
15: **end for**
Output: strong classifier $H(x) = \sum_k h_k(x), P(y = 1|x) = \frac{1}{1+e^{-H(x)}}$

To achieve online boosting learning for visual tracking, M weak classifiers are maintained during the tracking process. Once newly added data are obtained, all M weak classifiers are updated in parallel. Some of them are selected by our OGB algorithm (Table 5.1, Steps 3–15) to minimize the negative log-likelihood function (5.1).

$$L^m = -\sum_i (y_i \log(p_i^m) + (1 - y_i)\log(1 - p_i^m)), \tag{5.1}$$

where L^m denotes the loss after m round iteration, y_i stands for the label of the training sample x_i ($y_i \in \{0, 1\}$,"1" for a positive sample and "S0" for a negative sample), and p_i^m represents the probability $P(y_i = 1|x_i)$ after m round iteration. By assuming that training examples are independently sampled from a binomial distribution, to minimizing the negative log-likelihood function (5.1) is equal to enjoying all the associated asymptotic optimality weak classifiers of maximum likelihood estimation.

Then the selected weak classifiers can be combined by an additive model (5.2), where $h_k(x)$ indicates the kth weak classifier, and $H(x)$ indicates the final strong classifier.

$$H(x) = \sum_k h_k(x) \tag{5.2}$$

Finally, the output margin of the strong classifier can be interpreted as a probability $P(y = 1|x)$ using a sigmoid function (5.3).

$$P(y=1|x)=\sigma\left(H(x)\right), \quad \sigma\left(x\right)=\frac{1}{1+e^{-x}} \tag{5.3}$$

This operation not only provides a probability interpretation but also normalizes the output of weak classifier into the same level.

Notably, our OGB algorithm is a general framework; thus, the type of weak classifiers and the corresponding update mechanism should be designed by users for their specific applications. In the next section, we will introduce our proposed weak classifier, and the corresponding update mechanism for visual tracking.

Log Color Feature Distribution Ratio (LCFDR): How to design the weak classifiers is also a key issue for boosting algorithms. To achieve fast yet effective performance, our principle of developing weak classifiers is related to the distribution of color features. Generally, the 3D joint RGB histogram has been popular in object tracking [5] because of its rotation and scale invariance to a certain extent. However, it has two major disadvantages. First, the computation of joint histograms is extremely slow. Second, histograms with a large number of bins may lead to unstable features, as discussed in [19]. Many studies (such as [1]) demonstrate that using a set of 1D histograms by dividing the foreground region of the object of interest into a few subregions will potentially be a cheaper, yet slightly invariant approach of encoding features for rotation and scale. Recently, the manner in which all pixel colors has mapped all pixel colors into a set of 1D lines in RGB color space and how a set of 1D color histograms is built has drawn many researchers' attention [4, 16, 28]. In [4, 28], the authors adopt 49 and 13 different linear combinations of camera R, G, B pixel values to approximate 3D RGB color space, respectively. As shown in [4, 16, 28], the manner of linearizing RGB color space is a fast yet effective way of using color features.

The color features we adopt in this study are similar to that in [28]. The set of seed candidate features is composed of linear combinations of camera R, G and B pixel values (each seed candidate feature can be viewed as one line of RGB color space). In our experiments, we choose the following set of feature candidates:

$$\mathscr{F} \equiv \{w_1 R + w_2 G + w_3 B \,|\, w_* = [-1, 0, 1]\}. \tag{5.4}$$

The total number of such candidates will be 3^3. However, we only select 13 lines by pruning redundant coefficients where $\left(w_1', w_2', w_3'\right) = k\left(w_1, w_2, w_3\right)$ and $(w_1, w_2, w_3) = (0, 0, 0)$ and normalize them, such that all features are in the range of 0–255 (as shown in Fig. 5.3a). Then, we project all pixel colors on each line, and build a set of 1D histogram of them (Fig. 5.3b). Thus, the 13 lines through RGB color space can be considered as 13 types of color features. This feature pool contains many common features, such as raw "R","G","B" values, intensity feature "R+G+B", and approximate chrominance features (e.g., "R–B").

Thirteen weak classifiers are formed on the basis of the 13 lines in RGB color space (13 color features). We denote p_m and q_m as the normalized object and background histograms, respectively, which are related with the mth line (Fig. 5.3c). Thus, 13

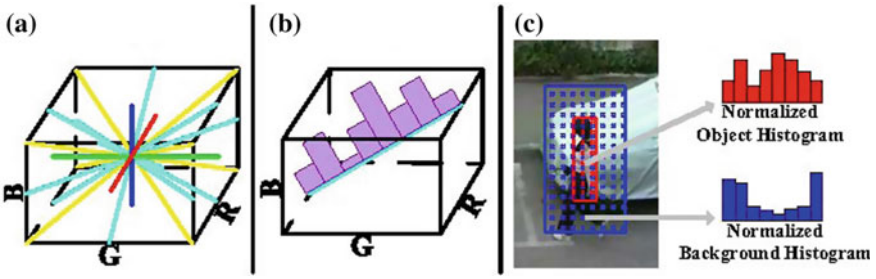


Fig. 5.3 Color features. **a** 13 lines in RGB color space. **b** Building 1D histogram on a specific line. **c** Extraction the normalized object background histograms

pairs of color feature distributions exist. Given a novel sample x (an individual color pixel with RGB values), the classifier margin of it follows Eq. (5.5). We refer to this novel weak classifier as LCFDR.

$$h_m(x) = \log \frac{\max\{p_m([x_m]), \delta\}}{\max\{q_m([x_m]), \delta\}}, \tag{5.5}$$

where x_m denotes the value given by projecting color pixel x on the m-th line (the m-th color feature), and $[x_m]$ is the index number that a special bin x_m belongs to. δ is a small constant (0.001 in this study) that prevents numerical problem (division by 0).

A similar work is available in [4], where the authors use the log likelihood ratio of a feature value (Eq. (5.5)) to obtain tuned features. However, in this study, we use it as the output margin of weak classifiers to achieve fast and effective training, updating, and testing because the boosting method is inclined to select complementary features. Hence, the novel weak LCFDR classifier can achieve excellent results.

Figure 5.4 provides an interpretation of LCFDR. Figure 5.4b shows three color features, which are calculated by "R+G+B", "R–G", and "R+G–B". The normalized object and background histograms are shown in Fig. 5.4c. On the basis of those histograms, the LCFDR values are calculated by Eq. (5.5) and shown in Fig. 5.4d. Figure 5.4e shows the local probability maps, which are calculated by Eq. (5.3). As shown in Fig. 5.4d, e, treating the LCFDR as a weak classifier is reasonable. For Fig. 5.4, the features "R+G+B" and "R–G" are good candidates, whereas the feature "R+G–B" is not a good choice.

By treating LCFDR as a weak classifier, we can present a fast implementation by combining Eqs. (5.3) and (5.5). Given that Eq. (5.3) contains an exponent (exp) operator and Eq. (5.5) contains a logarithm (log) operator, we introduce

$$h'_m(x) = \frac{\max\{p_m([x_m]), \delta\}}{\max\{q_m([x_m]), \delta\}}, \tag{5.6}$$

($h_m(x) = \log h'_m(x)$), and the output probability of the strong classifier can then be modified into

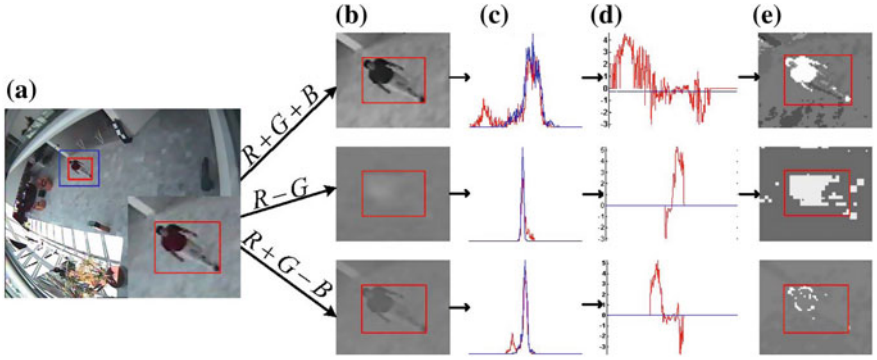


Fig. 5.4 Interpretation of LCFDR. **a** Specific frame, where the red and blue bounding boxes indicate the object of interest and its surrounding, respectively. **b** Three color features. **c** Distribution of the object and its surrounding, where red and blue curves indicate the object and background histograms, respectively. **d** LCFDR of the three color features. **e** Local probability maps

$$
\begin{aligned}
p\left(y=1|x\right) &= \frac{1}{1+e^{-H(x)}} = \frac{1}{1+e^{-\sum\limits_{K} h_k(x)}} \\
&= \frac{\prod\limits_{K} e^{h_k(x)}}{1+\prod\limits_{K} e^{h_k(x)}} = \frac{\prod\limits_{K} h'_k\left(x\right)}{1+\prod\limits_{K} h'_k\left(x\right)} = \theta\left(H'\left(x\right)\right),
\end{aligned}
\tag{5.7}
$$

where $H'\left(x\right) = \prod\limits_{K} h'_k\left(x\right)$, and $\theta\left(x\right) = \frac{x}{1+x}$.

Accordingly, Steps 5 and 14 in Table 5.2 can be changed into $p_i^m = \theta\left(H'_i \cdot h'_m\left(x_i\right)\right)$ and $H'_i = H'_i \cdot h'_k\left(x_i\right)$, respectively, with an initialization of $H'_i = 1$.

Therefore, we present a fast implementation of the OGB algorithm in Table 5.2 by combining OGB and LCFDR. As previously discussed, the proposed algorithm adopts addition and multiplication operators instead of exponent and logarithm operators, which leads the algorithm to be faster than the old version.

As previously mentioned, the output of the final strong classifier (5.7) indicates the probability that a pixel belongs to the tracked object. After the probability map is generated, we adopt the mean shift method with an Epanechnikov kernel [5] to determine the optimal location of the moving object. Figure 5.5 provides an explanation of locating the tracked object on the probability map. By initializing with the location of the previous frame l_{t-1} (the center of the blue bounding box in Fig. 5.5), the mean shift method seeks the optimal location l_t (the center of the red bounding box in Fig. 5.5) of the current frame by Eq. (5.8) iteratively.

$$
\begin{aligned}
l'_t &\leftarrow l_t + \frac{\sum\limits_{l_t^i \in \Omega(l_t)} \left(l_t^i - l_t\right) p\left(l_t^i\right)}{\sum\limits_{l_t^i \in \Omega(l_t)} p\left(l_t^i\right)} \\
l_t &\leftarrow l'_t
\end{aligned}
\tag{5.8}
$$

Table 5.2 Fast OGB algorithm

OGB (a fast implementation)

Input: Newly added data $\{x_i, y_i\}_{i=1}^{N}$, where $y_i \in \{0, 1\}$
Goal: Select no more than K weak classifiers from M weak classifier pool for building a strong classifier to label pixels that belong to an object or background ($M > K$)
1: Update all M weak classifiers in the pool with new data $\{x_i, y_i\}$
2: Initialize $L = +\infty$ and $H_i' = 1$ for all x_i (H_i': cumulative constant)
3: **for** k = 1 to K **do**
4: **for** m = 1 to M **do**
5: $p_i^m = \theta\left(H_i' \cdot h_m'(x_i)\right)$
where $\theta(x) = \frac{x}{1+x}$
6: $L^m = -\sum_i \left(y_i \log(p_i^m) + (1 - y_i)\log(1 - p_i^m)\right)$
7: **end for**
8: $m^* = \arg\min_m L^m$
9: **if** $L^{m*} > L$
10: break;
11: **end if**
12: $L = L^{m*}$
13: $h_k'(x) = h_{m^*}'(x)$
14: $H_i' = H_i' \cdot h_k'(x_i)$
15: **end for**
Output: strong classifier $H'(x) = \prod_k h_k'(x)$, $P(y = 1|x) = \frac{H'(x)}{1+H'(x)}$

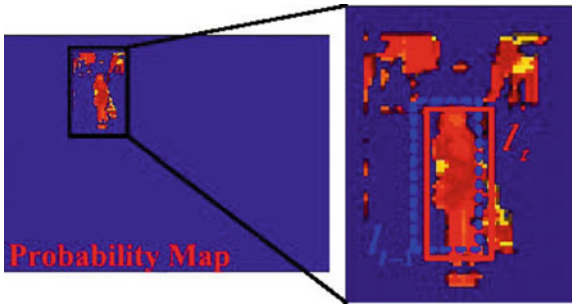


Fig. 5.5 Object location on the probability map. The solid red box and the dashed blue box indicate the location of the object in the current and previous frames, respectively

$\Omega(l_t)$ denotes a local region centered by l_t (its size is equal to the object region), $p(l)$ indicates the probability value at position l, and $\frac{\sum_{l_t^i \in \Omega(l_t)} (l_t^i - l_t) p(l_t^i)}{\sum_{l_t^i \in \Omega(l_t)} p(l_t^i)}$ is the mean shift vector.

The mean shift method converges after a few iterations (only requires 3–4 steps empirically). Then, the location of the object of interest is obtained.

For LCFDR, the output margin of each classifier is merely related with its corresponding color feature distribution; thus, the cost of updating the classifiers is extremely low. Once the location of the object of interest is given at the current

frame, the normalized object and background histograms about the current samples can be extracted (Fig. 5.3c) and then used to update the previous normalized object and background histograms, respectively.

$$p_m \leftarrow \left(1-\eta_p\right) p_m + \eta_p p'_m, \quad m = 1, 2, ..., 13 \tag{5.9}$$

$$q_m \leftarrow \left(1-\eta_q\right) q_m + \eta_q q'_m, \quad m = 1, 2, ..., 13 \tag{5.10}$$

For the m-th color feature, p'_m and q'_m are the current normalized object and background histograms, respectively. p'_m and q'_m are derived only from the current frame. η_p and η_q are the update rates of the object and background distributions, respectively. Notably, this manner of updating classifiers can also be immune to noisy samples to a certain extent because the weak classifiers are related with distributions rather than individual pixels.

5.3 Visual Tracking Based on Superpixel Segmentation

In this section, we attempt to use mid-level information for nonrigid object tracking, the basic idea of which is to mine the visual descriptions of superpixels.

This algorithm is formulated within the Bayesian framework in which the maximum a posterior estimate of the state given the observations up to time t is computed by

$$\begin{aligned} p(X_t|Y_{1:t}) = \alpha p(Y_t|X_t) \\ \int p(X_t|X_{t-1}) p(X_{t-1}|Y_{1:t-1}) dX_{t-1}, \end{aligned} \tag{5.11}$$

where X_t is the state at time t, $Y_{1:t}$ denotes all the observations up to time t, and α is a normalization term. In this work, the target state is defined as $X_t = (X_t^c, X_t^{sx}, X_t^{sy})$, where X_t^c represents the center location of the target, X_t^{sx} and X_t^{sy} denote its scale in x-axis and y-axis, respectively. As demonstrated by numerous works in the object tracking literature, it is critical to construct an effective observation model $p(Y_t|X_t)$ and an efficient motion model $p(X_t|X_{t-1})$.

In our formulation, a robust discriminative appearance model is constructed which, given an observation, computes the likelihood of it belonging to the target or the background. Thus the observation estimate of a certain target candidate X_t is proportional to its confidence:

$$p(Y_t|X_t) \propto \hat{C}(X_t), \tag{5.12}$$

where $\hat{C}(X_t)$ represents the confidence of an observation at state X_t being the target. The state estimate of the target $\hat{X}_t$ at time t can be obtained by the MAP estimate over the N samples at each time t. Let $X_t^{(l)}$ denote the l-th sample of the state X_t,

$$\hat{X}_t = \arg\max_{X_t^{(l)}} p(X_t^{(l)}|Y_{1:t}) \; \forall \, l = 1, ..., N. \tag{5.13}$$

In the following, the superpixel-based discriminative appearance model for tracking is introduced in Sect. 5.3, followed by construction of the confidence map based on this model in Sect. 5.3. The observation and motion models are presented in Sect. 5.3, and then the update scheme.

Superpixel-Based Discriminative Appearance Model: To construct an appearance model for the target and background, prior knowledge regarding the label of each pixel can be learned from a set of m training frames. That is, for a certain pixel at location (i, j) in the tth frame $pixel(t, i, j)$, we have:

$$y_t(i, j) = \begin{cases} 1 \text{ if } pixel(t, i, j) \in \text{target} \\ -1 \text{ if } pixel(t, i, j) \in \text{background}, \end{cases} \tag{5.14}$$

where $y_t(i, j)$ denotes the label of $pixel(t, i, j)$. Assume that the target object can be represented by a set of superpixels without considerably destroying the boundaries between the target and background (i.e., only few superpixels contain nearly equal amount of target and background pixels), prior knowledge regarding the target and background appearance can be modeled by

$$y_t(r) = \begin{cases} 1 \text{ if } sp(t, r) \in \text{target} \\ -1 \text{ if } sp(t, r) \in \text{background}, \end{cases} \tag{5.15}$$

where $sp(t, r)$ is the r-th superpixel in the t-th frame, and $y_t(r)$ denotes its corresponding label. However, such prior knowledge is not at our disposal in most tracking scenarios, and one feasible approach to achieve this is to infer prior knowledge from a set of samples, $\{X_t\}_{t=1}^m$, prior to the tracking process that begins with Eq. (5.15) from a small set of samples.

Segmentation Procedures: First, we segment the surrounding region of the target in the t-th training frame into N_t superpixels. The surrounding region is a square area centered at the location of target X_t^c, and its side length is equal to $\lambda_s[S(X_t)]^{\frac{1}{2}}$, where $S(X_t)$ represents the area size of target area X_t. We use a squared region for simplicity, which works well in practice, although rectangular or more sophisticated regions can be used at the expense of using a larger state space. Parameter λ_s, which controls the size of this surrounding region, is constant and is set to 3 in all the experiments. Therefore, the surrounding region is sufficiently large to cover the entire object in the last frame and include sufficient background region around the object for better discrimination. Each superpixel $sp(t, r)$ $(t = 1, ..., m, \; r = 1, ..., N_t)$ is represented by a feature vector f_t^r (see Fig. 5.6a–c). The mean shift clustering algorithm, with a single parameter controlling the bandwidth of the kernel function, is shown to capture the relationship among superpixels rather than other methods (e.g., k-means). Thus, in this work, we apply the mean shift clustering algorithm on the feature pool $F = \{f_t^r | t = 1, ..., m; \; r = 1, ..., N_t\}$ and obtain n different clusters. In the feature

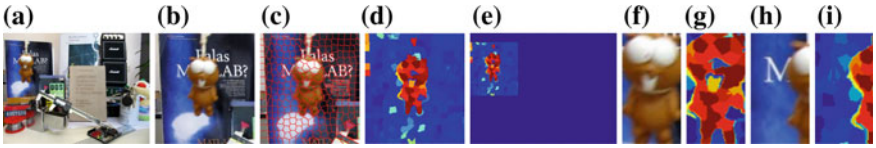


Fig. 5.6 Illustration of confidence map for state prediction. **a** A new frame at time t. **b** Surrounding region of the target in the last frame, i.e., at state $X_t^{(1)}$. **c** Superpixel segmentation from **b**. **d** Calculated confidence map of superpixels using Eqs. 5.18 and 5.17. The superpixels colored with red indicate a strong likelihood of belonging to the target. **e** Confidence map of the entire frame. **f**, **g** and **h**, **i** show two target candidates with high and low confidence, respectively

space, each cluster $clst(i)$ $(i = 1, ..., n)$ is represented by its cluster center $f_c(i)$, its cluster radius $r_c(i)$, and its own cluster members $\{f_t^r | f_t^r \in clst(i)\}$.

Confidence Computation: Every cluster $clst(i)$ corresponds to its own image region $S(i)$ in the training frames (image regions that superpixel members of cluster $clst(i)$ cover); thus, we count two scores for each cluster $clst(i)$, $S^+(i)$ and $S^-(i)$. The former denotes the size of cluster area $S(i)$ that overlaps with the target area at state X_t in the corresponding training frames, and the latter denotes the size of $S(i)$ outside the target area. Intuitively, the ratio $S^+(i)/S^-(i)$ indicates the likelihood that superpixel members of $clst(i)$ appear in the target area. Consequently, we assign each cluster a target-background confidence value between 1 and -1 to indicate whether its superpixel member belongs to the target or the background.

$$C_i^c = \frac{S^+(i) - S^-(i)}{S^+(i) + S^-(i)}, \ \forall\, i = 1, ..., n, \tag{5.16}$$

where larger positive values indicate high confidence to assign the cluster to the target, and vice versa.

Our superpixel-based discriminative appearance model is constructed on the basis of four factors, cluster confidence C_i^c, cluster center $f_c(i)$, cluster radius $r_c(i)$ and cluster members $\{f_t^r | f_t^r \in clst(i)\}$, which are used to determine the cluster for a certain superpixel, which will be described in the following sections. By applying the confidence values of each cluster to superpixels in the training frames, similar prior knowledge as Eq. (5.15) can be learned from a set of training images.

The merits of the proposed superpixel-based discriminative appearance model are illustrated in Fig. 5.8. That is, few background superpixels that appear in the target area (as a result of drifts or occlusions) are likely to be clustered into the same group with other background superpixels. Thus, the background pixels within the target region (enclosed by a rectangle) have negligible effect on our appearance model during training and update.

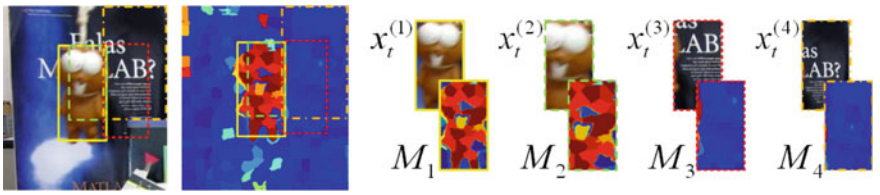


Fig. 5.7 Confidence map. Four target candidate regions corresponding to states $X_t^{(i)}$, $i = 1, \ldots, 4$ are shown in warped image and the confidence map. The confidence region of these candidates M_i, $i = 1, \ldots, 4$ have the same canonical size (upper right) after normalization. On the basis of Eq. 5.20, candidates $X_t^{(1)}$, $X_t^{(2)}$ have similar positive confidence (C_1, C_2), whereas $X_t^{(3)}$, $X_t^{(4)}$ have similar negative confidence (C_3, C_4). However, candidate $X_t^{(2)}$ covers less target area than $X_t^{(1)}$, and $X_t^{(4)}$ covers more background area than $X_t^{(3)}$. Intuitively, the target-background confidence of $X_t^{(1)}$ should be higher than $X_t^{(2)}$, whereas the confidence of $X_t^{(4)}$ should be lower than $X_t^{(3)}$. These two factors are considered in computing the confidence map, as described in Sect. 5.3

Confidence Map

When a new frame arrives, we initially extract a surrounding region[4] of the target and segment it into N_t superpixels (see Fig. 5.6b, c). To compute a confidence map for the current frame, we evaluate every superpixel and compute its confidence value. The confidence value of a superpixel depends on two factors, that is, the cluster it belongs to, and the distance between this superpixel and the corresponding cluster center in the feature space. The rationale for the first criterion is that if a certain superpixel belongs to cluster $clst(i)$ in the feature space, then the target-background confidence of cluster $clst(i)$ indicates how likely it belongs to the target or background. The second term is a weighting term that considers the distance metric. The farther the feature of a superpixel f_t^r lies from the corresponding cluster center $f_c(i)$ in feature space, the less likely this superpixel belongs to cluster $clst(i)$. Figure 5.7 illustrates how these factors are used for a confidence map. The confidence value of each superpixel is computed as follows:

$$C_r^s = w(r, i) \times C_i^c, \ \forall\, r = 1, ..., N_t, \tag{5.17}$$

and

$$\begin{aligned} &w(r, i) = \exp(-\lambda_d \times \tfrac{||f_t^r - f_c(i)||_2}{r_c(i)}), \\ &\forall\, r = 1, ..., N_t, \quad i = 1, ..., n, \end{aligned} \tag{5.18}$$

where $w(r, i)$ denotes the weighting term based on the distance between f_t^r (the feature of $sp(t, r)$, the r-th superpixel in the t-th frame) and $f_c(i)$ (the feature center of the cluster that $sp(t, r)$ belongs to). Parameter $r_c(i)$ denotes the cluster radius of cluster $clst(i)$ in the feature space, and λ_d is a normalization term (set to 2 in

[4] A square area centered at X_{t-1}^c with a side length of $\lambda_s[S(X_{t-1})]^{\frac{1}{2}}$.

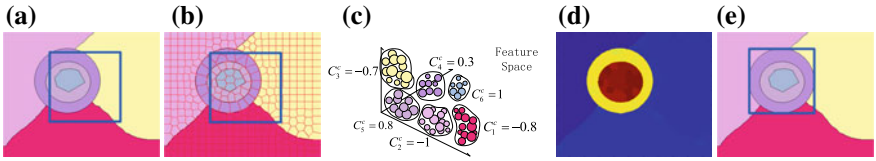


Fig. 5.8 Recovering from drifts. **a** A target object with tracking drifts. **b** Surrounding region of the target is segmented into superpixels. **c** Clustering results of **b** in the feature space and the target-background confidence of each cluster. **d** Confidence map in a new frame computed with clustering results. **e** MAP estimate of the target area where the tracker recovers from drifts. This illustration shows even if our tracker experiences drifts during tracking (see **a**), our appearance model obtains sufficient information from the surrounding background area by update, and provides the tracker with more discriminative strength against drifts than holistic appearance models

all experiments). By considering these two terms, C_r^s is the confidence value for superpixel r at the t-th frame, $sp(t, r)$.

We obtain a confidence map for each pixel on the entire current frame as follows. Every pixel in the superpixel $sp(t, r)$ is assigned with confidence C_r^s, and every pixel outside this surrounding region with confidence value -1. Figure 5.6a–e show the steps how the confidence map is computed with a new frame arriving at time t. This confidence map is computed based on our appearance model described in Sect. 5.3. In turn, the following steps for identifying the likely locations of the target in object tracking are based on this confidence map.

Observation and Motion Models

The motion (or dynamical) model is assumed to be Gaussian distributed, as shown as follows:

$$p(X_t|X_{t-1}) = \mathcal{N}(X_t; X_{t-1}, \Psi), \tag{5.19}$$

where Ψ is a diagonal covariance matrix whose elements are the standard deviations for location and scale (i.e., σ_c and σ_s). The values of σ_c and σ_s dictate how the proposed algorithm accounts for motion and scale change.

We normalize all these candidate image regions into canonical sized maps $\{M_l\}_{l=1}^N$ (the size of the target corresponding to X_{t-1} is used as the canonical size). We denote $v_l(i, j)$ as the value at location (i, j) of the normalized confidence map M_l of $X_t^{(l)}$. Then we accumulate $v_l(i, j)$ to obtain the confidence C_l for the state $X_t^{(l)}$, as shown as follows:

$$C_l = \sum_{(i,j)\in M_l} v_l(i, j). \tag{5.20}$$

However, this target-background confidence value C_l does not considers scale change. To make the tracker robust to the scale change of the target, we weigh C_l with respect to the size of each candidate as follows:

$$\hat{C}_l = C_l \times [S(X_t^{(l)})/S(X_{t-1})], \ \forall\, l = 1, ..., N, \tag{5.21}$$

where $S(X_{t-1})$ and $S(X_t^{(l)})$ represents the area sizes of target state X_{t-1} and candidate state $X_t^{(l)}$. For the target candidates with positive confidence values (i.e., indicating they are likely to be targets), the ones with a larger area size should be weighted more. For the target candidates with negative confidence values, the ones with a larger area size should be weighted less. This weighting scheme ensures that our observation model $p(Y_t|X_t^s)$ is adaptive to scale change. Figure 5.7 illustrates this weighting scheme.

We normalize the final confidence of all targets $\{\hat{C}_l\}_{l=1}^N$ within the range of [0, 1] for computing likelihood of $X_t^{(l)}$ for our observation model, as shown as follows:

$$p(Y_t|X_t^{(l)}) = \underline{\hat{C}_l}, \ \forall\, l = 1, ..., N, \tag{5.22}$$

where $\underline{\hat{C}_l}$ denotes the normalized confidence value for each sample. With the observation model $p(Y_t|X_t^{(l)})$ and the motion model $p(X_t^{(l)}|X_{t-1})$, the MAP state $\hat{X}_t$ can be computed with Eq. 5.19. Figure 5.6f–i show two samples and their corresponding confidence maps. From the figures, the confidence maps facilitate the process of determining the most likely target location.

5.4 Tracking via End-to-End Fully Convolutional Networks (FCNs)

In this section, we begin by describing the architectures of the famous FCNs [17] and the proposed tailored FCN network(TFCN). Then we give the details of how to generate the discriminative saliency map is generated on the basis of our TFCN, which is competent in handling tracking problems.

FCN Architectures: The incipient FCN architecture [17] is an end-to-end, pixel-to-pixel learning model that can produce a pixel-wise prediction and has been widely used for dense labeling tasks. The model differs from traditional CNN model because it essentially converts all fully connected layers into convolution operators and use transposed convolutions for upsampling feature maps. Specifically, the output of a convolutional operator is calculated by

$$\mathbf{Y}^j = f(\sum_i \mathbf{X}^i * \mathbf{W}^{i,j} + b^j), \tag{5.23}$$

where the operator * represents 2D convolution, $f(\cdot)$ is an element-wise nonlinear activation function, e.g., ReLu ($f(x) = \max(0, x)$). $\mathbf{X}^i$ is the i-th input feature map and $\mathbf{Y}^j$ is the jth output feature map. $\mathbf{W}^{i,j}$ is a filter with a size of $k \times k$, and b^j is the corresponding bias term. The transposed convolutions perform the transformation in the opposite direction of a normal convolution. In the FCN model, transposed convolutions are used to project feature maps to a higher dimensional space.

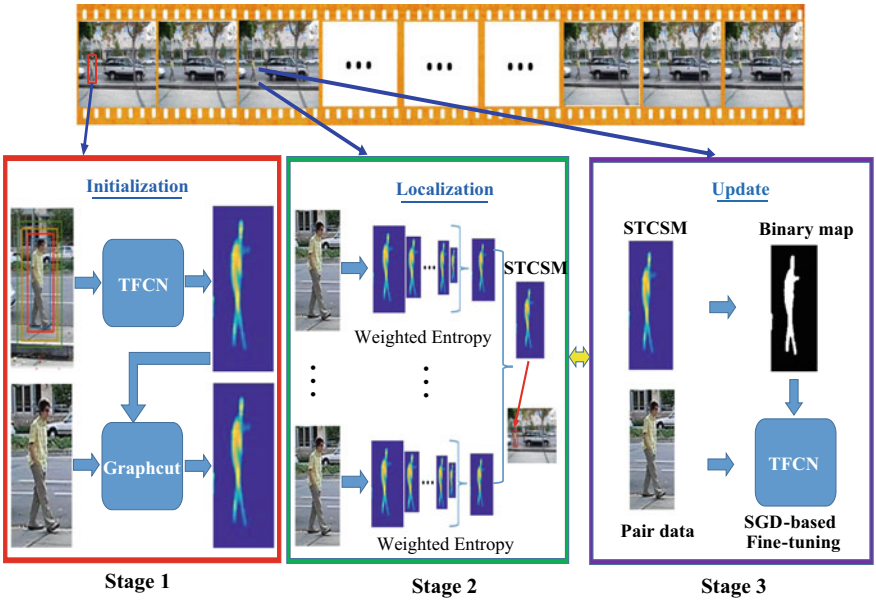


Fig. 5.9 Overall framework of the proposed tracking approach. In the first stage, we use the proposed multiscale multi-region image representation methods and the pretrained TFCN to predict the local saliency maps in the start frame. The Graphcut [22] is used in the start frame to increase the robustness of saliency region predictions. Then, we use the weighted entropy to fuse the local saliency maps, thereby resulting in a discriminative saliency map for the target localization. During tracking, we use the STCSM model to incorporate spatial-temporal information into the location inference. After the localization, we perform thresholding on the STCSM and update the TFCN with the binary saliency map by stochastic gradient descent (SGD) methods

As shown in Fig. 5.10a, the FCN model initially performs several layers of convolution and pooling on the image or feature maps to extract multiscale feature representations of the image. Then, the back-end layers perform several transposed convolutions that increase the resolution-reduced feature maps to the image size. Finally, the prediction is achieved by applying the pixel-wise classification with a Softmax function. In [17], the authors introduce several skip connections, which add high-level prediction layers to intermediary layers to generate prediction results at multiple resolutions. The skip connections considerably improve the semantic segmentation performance.

TFCN: The proposed TFCN model is largely inspired by the FCN-8s [17] semantic segmentation model due to the two common characteristics between saliency detection and semantic segmentation. First, the goals of saliency detection and semantic segmentation are relatively close. The goal of saliency detection is to extract the salient region from the background, whereas semantic segmentation aims to distinguish different objects from the background. Second, both tasks produce pixel-level outputs. Each pixel of the input image requires to be categorized into two or multiple

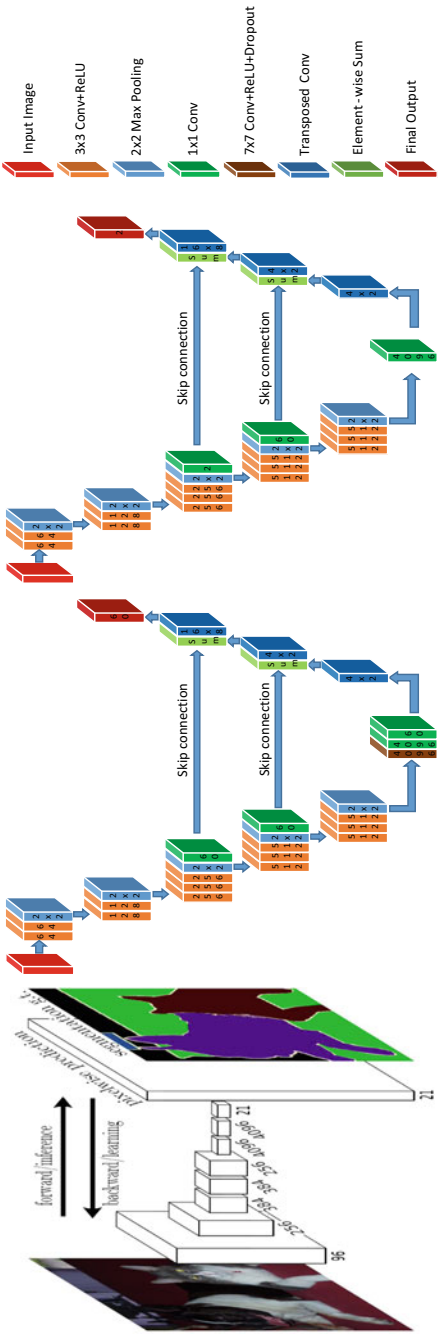


Fig. 5.10 Detailed comparisons of the FCN-8s [17] and our proposed TFCN model. **a** FCN. **b** State-of-the-art FCN-8s sematic segmentation model. **c** Proposed TFCN model for region saliency detection. Conv stands for convolution. The number inside each layer indicates the amount of convolutional kernels

classes. Thus, the pretrained FCN-8s model can be utilized to provide prior information on generic objects. The original FCN-8s model introduces two skip connections and adds high-level prediction layers to intermediary layers to generate prediction results at the resolution of image size.

We introduce scale considerations and modify the terminal structure of the original FCN-8s model to adapt to the saliency detection task and accelerate the training process. The major modifications include: (1) changing the filter size from 7×7 to 1×1 in the $fc6$ layer to enlarge resolutions of feature maps and keep abundant details; (2) discarding the $drop6$, $fc7$, $relu7$, and $drop7$ layers because of their insignificant contribution for our tasks, and connecting the $fc6$ and $score59$ layers directly; (3) setting the num_out as 2 in the $upsample\text{-}fused\text{-}16$, $score\text{-}pool3$, and $upsample$ layers because the TFCN model is expected to predict the scores for two classes (salient foreground and general background); and (4) keeping the num_out value unchanged in $score59$, $upscore2$, and $score\text{-}pool4$ layers to integrate moderate semantic information. To be more precise, Fig. 5.10b, c illustrate the detailed differences of the original FCN-8s and the proposed TFCN models, respectively.

Local Region Saliency Maps and Their Fusion: The proposed TFCN model is adopted to generate local saliency maps, which will be used for visual tracking. The overall procedure comprises pretraining the TFCN, extracting scale-dependent regions, and fusing discriminative saliency maps.

The TFCN model is derived from FCN-8s designed for the semantic segmentation task. Thus, the direct application on salient object detection may lead to a negative transfer [20]. To deal with this issue, we first pretrain the TFCN model on the basis of a well-collected saliency dataset before conducting local saliency detection. Formally, given the salient object detection training dataset $S = \{(X_n, Y_n)\}_{n=1}^{N}$ with N training pairs, where $X_n = \{x_j^n,\ j = 1, ..., T\}$ and $Y_n = \{y_j^n,\ j = 1, ..., T\}$ are the input image and the binary ground truth image with T pixels, respectively. $y_j^n = 1$ and $y_j^n = 0$ denote the foreground and background pixels, respectively. For notional simplicity, we subsequently drop the subscript n and consider each image independently. We denote $\mathbf{W}$ as the parameters of the TFCN. For the pretraining, the loss function can be expressed as

$$\begin{aligned} \mathscr{L}_f(\mathbf{W}) = &-\beta \sum_{j \in Y_+} \mathbf{log\ Pr}(y_j = 1|X; \mathbf{W}) \\ &-(1-\beta) \sum_{j \in Y_-} \mathbf{log\ Pr}(y_j = 0|X; \mathbf{W}), \end{aligned} \tag{5.24}$$

where Y_+ and Y_- denote the foreground and background label sets, respectively. The loss weight $\beta = |Y_+|/(|Y_+| + |Y_-|)$. $|Y_+|$ and $|Y_-|$ denote the number of foreground and background pixels, respectively. $\Pr(y_j = 1|X; \mathbf{W}) \in [0, 1]$ is the confidence score that measures how likely the pixel belong to the foreground. The ground truth of each image in the saliency dataset is a 0–1 binary map, which perfectly matches the channel output of the TFCN model. For the saliency inference, outputs

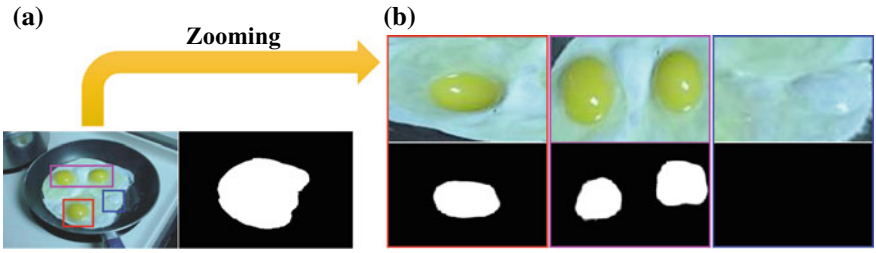


Fig. 5.11 Illustration of the effect of different received visual fields. **a** Original image with the salient region. **b** Top: Three selected typical regions, which are almost included by the overall salient region. Bottom: Corresponding saliency maps

of the last convolutional layer are utilized to distinguish the saliency foreground from the general background.

Before putting the training images into the TFCN, each image is subtracted with the ImageNet mean [6] and resized into the same size (500×500). For the correspondence, we also resize the 0–1 binary maps to the same size. Our pretraining uses SGD with a momentum, learning rate decay schedule.

The human visual system suggests that the size of the received visual fields affects the fixating mechanisms considerably [25]. Figure 5.11a shows a natural image with structural and hierarchical characteristics. Human visual system focuses on the fried sunny-side up eggs and makes these regions salient as shown in the right. If we zoom in a specific region (Fig. 5.11b), then we may select the area of the egg yolks as the most salient region or obtain an inconspicuous activation. This sensibility of visual perception motivates us to present a novel image representation method and develop a saliency detection model based on the selected regions with different spatial layouts and scale variations.

More specifically, we exploit a multi-region target representation scheme (Fig. 5.12). We divide each image region into seven parts and calculate seven saliency maps over these regions. The detailed configuration is as follows. (1) The first saliency map is obtained from the entire image region. (2) The subsequent four saliency maps are calculated on the four equal parts to introduce spatial information. (3) The last two saliency maps are extracted from the inside and outside areas to highlight the scale support. In addition, we introduce a multiscale mechanism into each part to enhance the diversity of the region representation. N scales are sampled to generate multiple regions with different scales for a given part of the centered region (l_x, l_y) with size (w_0, h_0). In this study, we select the sizes $(n \times w, n \times h)$, where $w = \frac{3}{4}w_0$, $h = \frac{3}{4}h_0$, and $n = 1, 2, ..., N$.

On the basis of the proposed multiscale multi-region scheme, we can obtain $M \times N$ saliency maps ($\mathbf{S}_{m,n}, m = 1, ..., M, n = 1, ..., N$) in total, where M denotes the number of regions for describing spatial layouts, and N is the number of sampled scales. Each saliency map $\mathbf{S}_{m,n}$ is calculated based on the TFCN model, that is,

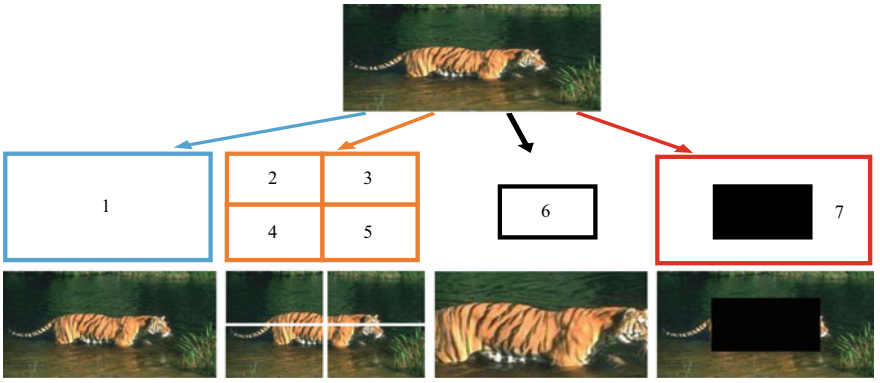


Fig. 5.12 Illustration of the multi-region representation

$$\mathbf{S}_{m,n} = padding(\mathbf{M}^E_{m,n} - \mathbf{M}^I_{m,n}), \tag{5.25}$$

where $\mathbf{M}^E_{m,n}$ and $\mathbf{M}^I_{m,n}$ are the exaction and inhibition maps obtained by the TFCN outputs, respectively. $padding(.)$ guarantees that different saliency maps are of equal sizes. $\mathbf{M}^E$ and $\mathbf{M}^I$ have reciprocal properties and the proposed strategy can eliminate several noises introduced by transposed convolution operations (Fig. 5.13b–d). Then, we adopt an additive rule to integrate the information of different regions for each scale, as shown as follows:

$$\mathbf{S}_n = \max(\sum_{m=1}^{M} \mathbf{S}_{m,n}, 0), \tag{5.26}$$

where the $\max(.)$ operator avoids model degradation.

Finally, we use a weighted strategy to combine multiple saliency maps with different scales effectively, as shown as follows:

$$\mathbf{S} = \sum_n w_n \mathbf{S}_n, \tag{5.27}$$

where $\mathbf{S}$ is the fused saliency map, w_n is the weight of the n-th scale ($w_n \geq 0$, $\sum_n w_n = 1$). Generally, a more important saliency map has a large weight assigned to it. Thus, we utilize the weighted entropy to measure the discriminative power of the fused map. Let $\mathbf{w} = [w_1, w_2, .., w_n]^T$. Then, the weighted entropy is defined as,

$$H(\mathbf{w}) = -\Gamma(\mathbf{w}) \sum_i s_i^{\alpha+1} \ln s_i, \tag{5.28}$$

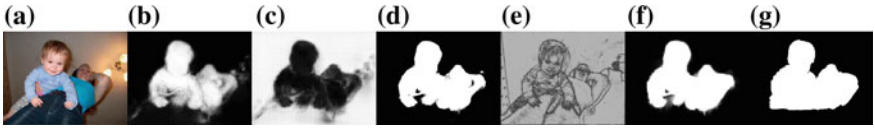


Fig. 5.13 Illustration of saliency maps. From left to right: **a** Original image; **b** Exaction map; **c** Inhibition map; **d** Saliency map; **e** Texture map generated by fuzzy logical filters; **f** Final saliency map after domain transform; and **g** Ground truth

where $\Gamma(\mathbf{w}) = 1/\sum_i s_i^\alpha$ denotes a normalization term, α denotes a constant, and s_i is a function of $\mathbf{w}$. Here, we select the function in Eq. 5.27. To reduce computational complexity, we set $\alpha = 1$ and obtain $\Gamma(\mathbf{w}) = 1/\sum_i s_i^\alpha = 1$.

According to [12], a small weighted entropy represents that the saliency map is highly different from others, indicating more discrimination. The optimal weight vector $\mathbf{w}$ can be obtained by minimizing the objective function (Eq. 5.28), which can be effectively solved based on the iterative gradient descent method in [18]. After obtaining the fused saliency map $\mathbf{S}$, we utilize the domain transform technique to enhance its spatial consistency. Specifically, a high-quality edge preserving filter [9] is performed on $\mathbf{S}$ with the texture map generated by fuzzy logical filters. As shown in Fig. 5.13d–f, several holes or disconnected regions can be filled after the domain transform.

Applying the TFCN Model for Tracking: We present the proposed nonrigid object tracking method based on the resulting saliency map. We show the manners in which the state of target objects with the discriminative saliency map is initialized and the spatial-temporal saliency information is utilized for object tracking. The overall framework of the proposed method is shown in Fig. 5.9. The critical components and discussions are presented as follows.

Given the center location (x_1, y_1) and specified region R_1 of the target object in the first frame, we initially crop an image patch P_1 centered at the target location with the 1.5 times target size ($W_1 \times H_1$), which is calculated on the specified region. Subsequently, the image patch P_1 is placed into the pretrained TFCN model to generate a saliency map $\mathbf{S}_1$ through feed-forward propagation. To improve the robustness of the saliency map, Grabcut [23] is conducted to obtain a foreground mask $\mathbf{M}_1$ using the map $\mathbf{S}_1$ as a prior. Finally, the TFCN model is fine-tuned based on the intersection region of $\mathbf{S}_1$ and $\mathbf{M}_1$ maps in the first frame, which provides accurate information of the target.

For target localization, the shape and deformation information of the tracked object in previous frames can be utilized to predict the new state in the current frame because of the assumption of spatial-temporal consistency. After cropping the same region in the t-th frame, we determine the state of the tracked object on the basis of the STCSM model, which is defined as

$$\mathbf{S}_t^{STC} = \mathbf{S}_t + \sum_{k=t-\tau-1}^{t-1} \beta(k)\mathbf{S}_k^{STC}, \beta(k) = \frac{1}{c^k}, \tag{5.29}$$

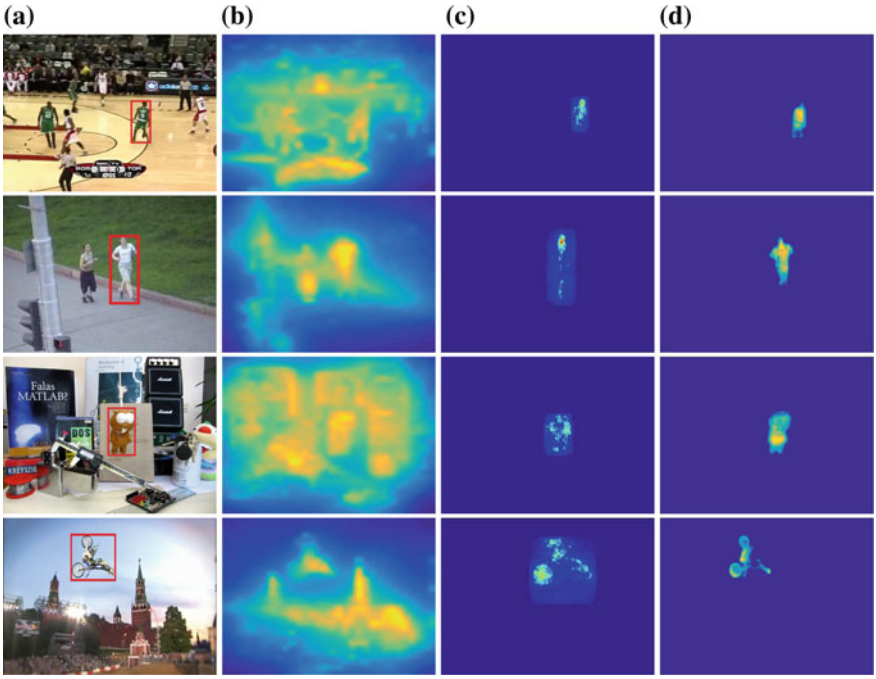


Fig. 5.14 Saliency maps of different methods. From left to right: **a** Input frames; **b** Video saliency detection [15]; **c** Target-specific saliency map [14]; and **d** TFCN

where $\mathbf{S}_t$ denotes the saliency map in the current frame, which can be obtained by the saliency detection method presented in Section III.C. $\mathbf{S}_t^{STC}$ is the STCSM model up to the t-th frame, τ is the accumulated time interval, and $\beta(k)$ corresponds to the weights of previous STCSM models with a decay factor c. $\beta(k)$ imposes higher weights for recent frames and lower weights for previous frames. In addition, thanks to the availability of the TFCN model, $\mathbf{S}_t$ can be used to generate the accumulated saliency map before determining the location of the target object. This is different from most of existing trackers which only consider previous information and directly detect the location on the current frame. The proposed accumulated saliency maps are also pixel-wise maps that possess the structural property and computational scalability. The structural property maintains discriminative information between backgrounds and the target object.

For target localization, we can directly treat the saliency region obtained by the STCSM method as the tracking result in the current frame. The obtained saliency region provides not only accurate tracking states but also detailed segmentation masks concentrating the tracked object (Fig. 5.14d). This strategy provides more accurate tracking states which is very appropriate for nonrigid object tracking. In addition, we can also use a compact rectangle that includes the overall saliency region and consider the center of this rectangle as the location of the tracked object. The latter

strategy facilitates the fair comparisons between our method and many bounding box-based trackers.

To update the proposed tracker for online adaptation, we first convert the STCSM model into binary maps using a thresholding operator and treat this binary map as ground truth in the current frame. Subsequently, we fine-tune the TFCN model from the $upsample\text{-}fused\text{-}16$ layer to the $loss$ layer with the SGD algorithm to enhance the adaptiveness of the tracker effectively. We have one labeled image pair in each frame, fine-tuning the TFCN only with this image pair tends to result in overfitting. Thus, we employ data augmentation by mirror reflection and rotation techniques. Simultaneously we also use the tracked regions of the recent 20 frames for fine-tuning.

5.5 Experimental Results and Discussions

The most recent dataset of nonrigid object tracking [24] includes 11 challenging image sequences with pixel-wise annotations in each frame. The bounding box annotations are generated by computing the tightest rectangular boxes containing all target pixels. Based on this dataset, we compare the methods(BSD, SPT and TFCN) introduced in the previous sections with some popular trackers (OGBT [24], PT [7], Struck [13] and FCNT [26]). These trackers based on hard-crafted [7, 13, 24] or deep [26] features achieve top performance on recent large-scale tracking benchmarks [29].

Table 5.3 Quantitative results on the nonrigid object tracking dataset [?]. All numbers are presented in terms of percentage (%). For each row, the best and second-best results are shown in bolditalic and bold values respectively

	(a) Bounding box overlap ratio							(b) Segmentation overlap ratio				
	SPT	PT	OGBT	Struck	BCD	FCNT	TFCN	SPT	PT	OGBT	BCD	TFCN
Cliff-dive1	66.5	29.9	**75.9**	62.4	61.8	62.3	***77.2***	54.6	60.1	67.6	64.5	***71.4***
Cliff-dive2	30.3	13.4	49.3	34.0	30.0	36.2	***54.7***	41.8	16.0	36.7	43.1	***50.8***
Diving	35.2	12.3	***50.6***	33.6	15.1	24.3	**44.8**	21.2	25.5	44.1	33.2	***42.4***
Gymnastics	42.6	26.0	70.4	53.7	13.9	56.6	***74.6***	10.6	52.0	**69.8**	53.2	***72.4***
High-jump	5.3	0.6	***51.5***	15.4	8.4	44.2	**49.4**	***52.1***	0.9	42.8	32.5	**46.6**
Motocross1	11.2	3.6	62.4	29.2	10.8	67.6	***70.5***	8.9	1.4	53.1	54.2	***59.2***
Motocross2	46.1	21.3	**72.1**	70.6	41.7	67.9	***74.9***	37.1	39.7	64.5	56.8	***69.7***
Mtn-bike	53.0	12.9	61.2	66.2	71.4	68.5	**71.0**	43.0	32.1	54.9	48.2	***57.4***
Skiing	27.7	21.3	39.0	3.2	7.4	***54.1***	**53.2**	37.3	43.0	32.1	38.6	***47.5***
Transformer	55.8	13.9	**86.6**	57.7	55.6	72.8	***89.4***	2.8	5.5	74.0	54.3	***76.8***
Volleyball	26.8	15.2	46.2	36.2	13.2	45.3	**49.2**	6.5	25.1	41.1	25.1	***45.2***
Average	36.4	15.5	**60.5**	42.0	29.9	54.5	***64.5***	28.7	27.4	52.8	45.8	***58.1***

Similar to [24], we adopt two overlap ratio rules to evaluate the proposed methods and other competing ones. The bounding box overlap ratio is used to compare all trackers; whereas the segmentation overlap ratio is adopted for comparing nonrigid tracking algorithms with segmentation outputs. The average overlap ratio results are demonstrated in Table 5.3. From Table 5.3, we have two fundamental observations: (1) other deep learning-based trackers have not taken considered segmentation, however, their performance on this nonrigid dataset are still competitive; (2) TFCN outperforms all compared algorithms in both bounding box and segmentation overlap ratios of most sequences. Therefore, TFCN is more suitable for tracking deformable and articulated objects.

5.6 Summary

This section mainly introduces three segmentation-based tracking algorithms. BCD can capture the position of object of interest and update itself in an online fashion, which just needs to be initialized with the first labeled frame. SPT shows that the use of superpixels provides flexible and effective mid-level cues, which are incorporated in an appearance model to distinguish the foreground target and the background. TFCN presents a novel nonrigid object tracking method based on STCSM. All these three segmentation-based tracking algorithms demonstrate the potential of segmentation in visual tracking.

References

1. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: CVPR, pp. 798–805 (2006)
2. Avidan, S.: Ensemble tracking. IEEE Trans. Pattern Anal. Mach. Intell. **29**(2), 261–271 (2007)
3. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: CVPR, pp. 983–990 (2009)
4. Collins, R.T., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1631–1643 (2005)
5. Comaniciu, D., Ramesh, V., Meer, P.: Real-time tracking of non-rigid objects using mean shift. In: CVPR, pp. 142–149 (2000)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, pp. 248–255 (2009)
7. Duffner, S., Garcia, C.: Pixeltrack: a fast adaptive algorithm for tracking non-rigid objects. In: International Conference on Computer Vision, pp. 2480–2487 (2013)
8. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Ann. Stat. **28**, 337–407 (2009)
9. Gastal, E.S., Oliveira, M.M.: Domain transform for edge-aware image and video processing. In: ACM Transactions on Graphics, vol. 30, p. 69 (2011)
10. Grabner, H., Bischof, H.: On-line boosting and vision. In: CVPR, pp. 260–267 (2006)
11. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised on-line boosting for robust tracking. In: ECCV, pp. 234–247 (2008)

12. Guiaşu, S.: Weighted entropy. Rep. Math. Phys. **2**(3), 165–179 (1971)
13. Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M.M., Hicks, S.L., Torr, P.H.: Struck: structured output tracking with kernels. IEEE Trans. Pattern Anal. Mach. Intell. **38**(10), 2096–2109 (2016)
14. Hong, S., You, T., Kwak, S., Han, B.: Online tracking by learning discriminative saliency map with convolutional neural network. In: International Conference on Machine Learning, pp. 597–606 (2015)
15. Kim, H., Kim, Y., Sim, J.Y., Kim, C.S.: Spatiotemporal saliency detection for video sequences based on random walk with restart. IEEE Trans. Image Process. **24**(8), 2552–2564 (2015)
16. Liu, R., Cheng, J., Lu, H.: A robust boosting tracker with minimum error bound in a co-training framework. In: ICCV, pp. 1459–1466 (2009)
17. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
18. Ma, L., Lu, J., Feng, J., Zhou, J.: Multiple feature fusion via weighted entropy for visual tracking. In: International Conference on Computer Vision, pp. 3128–3136 (2015)
19. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 971–987 (2002)
20. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010)
21. Parag, T., Porikli, F., Elgammal, A.M.: Boosting adaptive linear weak classifiers for online learning and tracking. In: CVPR, pp. 1–8 (2008)
22. Rother, C., Kolmogorov, V., Blake, A.: "GrabCut": interactive foreground extraction using iterated graph cuts. TOG **23**(3), 309–314 (2004)
23. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. **23**, 309–314 (2004)
24. Son, J., Jung, I., Park, K., Han, B.: Tracking-by-segmentation with online gradient boosting decision tree. In: International Conference on Computer Vision, pp. 3056–3064 (2015)
25. Tsotsos, J.K., Culhane, S.M., Wai, W.Y.K., Lai, Y., Davis, N., Nuflo, F.: Modeling visual attention via selective tuning. Artif. Intell. **78**(1–2), 507–545 (1995)
26. Wang, L., Lu, H., Wang, D.: Visual tracking via structure constrained grouping. Signal Process. Lett. **22**(7), 794–798 (2015)
27. Wang, S., Lu, H., Yang, F., Yang, M.: Superpixel tracking. In: ICCV, pp. 1323–1330 (2011)
28. Wei, Y., Sun, J., Tang, X., Shum, H.Y.: Interactive offline tracking for color objects. In: ICCV, pp. 1–8 (2007)
29. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: a benchmark. In: Computer Vision and Pattern Recognition, pp. 2411–2418 (2013)
30. Yang, F., Lu, H., Yang, M.: Robust superpixel tracking. IEEE Trans. Image Process. **23**(4), 1639–1651 (2014)

Chapter 6
Correlation Tracking

The correlation filter (CF) has become one of the most widely used formulas in visual tracking for its computation efficiency. With low computational load, the CF-based tracker can exploit large numbers of cyclically shifted samples for learning, thus showing superior performance. However, existing attempts usually focus on the former one while pay less attention to reliability learning, which makes the learned filters dominated by the unexpected salient regions on the feature map, thereby resulting in model degradation. In this chapter, a novel CF-based optimization problem is introduced to jointly model the discrimination and reliability information.[1]

6.1 Introduction

The early CF-based trackers exploit a single feature channel as input, and thus usually have very impressive tracking speed. The MOSSE tracker [2] exploits the adaptive correlation filter, which optimizes the sum of squared error. Henriques [10] introduce the kernel trick into the correlation filter formula. By exploiting the property of the circulant matrix, they provide an efficient solver in the Fourier domain. The KCF [11] tracker further extends the method [10], and shows improved performance can be achieved when multichannel feature maps are used. Similarly, the color naming features are also introduced to achieve robust tracking in color video clips [8]. The DSST [5], SAMF [14] and IBCCF [13] track ers address the scale adaptation problem using multi-scale searching strategies.

With great success of deep convolutional neural network (CNN) in object detection and classification, more and more CF-based trackers resort to the pretrained CNN model for robust target representation [4, 23, 24]. Since most of CNN models

[1] ©2018 IEEE, Reprinted, with permission, from ref. [21].

H. Lu and D. Wang, *Online Visual Tracking*,
https://doi.org/10.1007/978-981-13-0469-9_6

are pretrained with respect to the task of object classification or detection, they tend to retain the features useful for distinguishing different categories of objects, and lose much information for instance level classification. Thus, the responses of the feature map are usually sparsely and nonuniformly distributed, which makes the learned filter weights inevitably highlight the high response regions. In this case, the tracking results are dominated by such high response regions, while these regions are in fact not always reliable (see Fig. 6.1 for an example).

Deeply investing the representation property of different convolution layers in the CNN model, Ma [18] propose to combine feature maps generated by three layers of convolution filters, and introduce a coarse-to-fine searching strategy for target localization. Henriques et al. [10] propose to use the continuous convolution filter for combinations of feature maps with different spatial resolutions. As fewer model parameters are used in the model, the tracker [10] is insusceptible to the over-fitting problem, and thus has superior performance than [18]. Another research hotspot for the CF-based methods is how to suppress the boundary effects. Typical methods include the trackers [9, 12]. In the SRDCF tracker [9], a spatial regular term is exploited to penalize the filter coefficients near the boundary regions. Different from [9], the BACF tracker [12] directly multiplies the filter with a binary matrix. This tracker can generate real positive and negative samples for training while at the same time share the computation efficiency of the original CF method. Compared to our method, these trackers have not attempted to suppress the background regions inside the target bounding box, and their learned filter weights tend to be dominated by the salient regions in the feature map.

Patch-based correlation filters have also been widely exploited [16, 17]. Liu [17] propose an ensemble of part trackers based on the KCF method, and use the peak-to-sidelobe ratio and the smooth constraint of confidence map for combinations of different base trackers. In the method [16], the authors attempt to learn the filter coefficients of different patches simultaneously under the assumption that the motions of sub-patches are similar. Li [15] detect the reliable patches in the image, and propose to use the Hough voting-like strategy to estimate the target states based on the sub-patches. Most of the previous patch-based methods intend to address the problems of deformation and partial occlusion explicitly.

However, as the correlation filter takes the entire image as the positive sample and the cyclically shifted images as negative ones, the learned filters are likely to be influenced by the background regions. Existing methods (e.g., [4, 9, 10]) address this problem by incorporating a spatial regularization on the filter, so that the learned filter weights focus on the central part of the target object. In [12], the authors prove that the correlation filter method can be used to simulate the conventional ridge regression method. By multiplying the filter with a binary mask, the tracker is able to generate the real training samples having the same size as the target object, and thus better suppressing the background regions. However, this method has two limitations: first, it exploits the augmented Lagrangian method for model learning, which limits the model extension; second, even though the background region outside the bounding box is suppressed, the tracker may also be influenced by the background region inside the bounding box.

In this chapter, we present a novel CF-based optimization problem to clearly learn the discrimination and reliability information, and then develop an effective tracking method (denoted as DRT [22]). The concept of the base filter is proposed to focus on discrimination learning. To do this, we introduce the local response consistency constraint into the traditional CF framework. This constraint ensures that the responses generated by different subregions of the base filter has small difference, thereby emphasizing the similar importance of each subregion. The reliability weight map is also considered in our formula. It is online jointly learned with the base filter and is aimed at learning the reliability information. The base filter and reliability term are jointly optimized by the alternating direction method, and their element-wise product produces effective filter weights for the tracking task. Finally, we conduct extensive experiments on three benchmark datasets to demonstrate the effectiveness of our method.

6.2 Correlation Tracking via Joint Discrimination and Reliability Learning

We first briefly revisit the correlation filter (CF) formula. Let $\mathbf{y}=[y_1, y_2, ..., y_K]^\top \in \mathbb{R}^{K\times 1}$ denote gaussian shaped response, and $\mathbf{x}_d \in \mathbb{R}^{K\times 1}$ be the input vector (in the two-dimensional case, it should be a feature map) for the d-th channel, then the correlation filter learns the optimal $\mathbf{w}$ by optimizing the following formula:

$$\dot{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{k=1}^{K} \left(y_k - \sum_{d=1}^{D} \mathbf{x}_{k,d}^\top \mathbf{w}_d \right)_2^2 + \lambda \|\mathbf{w}\|_2^2, \tag{6.1}$$

where $\mathbf{x}_{k,d}$ is the k-step circular shift of the input vector $\mathbf{x}_d$, y_k is the k-th element of $\mathbf{y}$, $\mathbf{w}=\left[\mathbf{w}_1^\top, \mathbf{w}_2^\top, ..., \mathbf{w}_D^\top\right]^\top$ where $\mathbf{w}_d \in \mathbb{R}^{K\times 1}$ stands for the filter of the d-th channel. For circular matrix in CF, K stands for both the dimension of features and the number of training samples. An analytical solution can be found to efficiently solve the optimization problem (6.1) in the Fourier domain.

To address this issue, we propose a novel CF-based optimization problem to jointly model the discrimination and reliability information.

Joint Discrimination and Reliability Model: Different from the previous CF-based methods, we treat the filter weight $\mathbf{w}_d$ of the d-th feature channel as the element-wise product of a base filter $\mathbf{h}_d$ and a reliability weight map $\mathbf{v}_d$,

$$\mathbf{w}_d = \mathbf{h}_d \odot \mathbf{v}_d, \tag{6.2}$$

where $\odot$ is the hadamard product, $\mathbf{h}_d \in \mathbb{R}^{K\times 1}$ is used to denote the base filter, $\mathbf{v}_d \in \mathbb{R}^{K\times 1}$ is the reliability weight for each target region, the values of $\mathbf{v}_d$ corresponding to the non-target region are set to zeros (illustrated in Fig. 6.2).

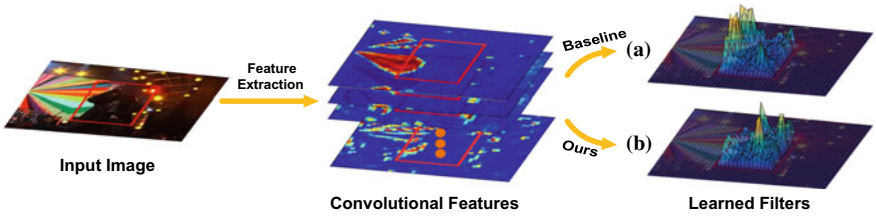


Fig. 6.1 Example showing that our learned filter coefficients are insusceptible to the response distribution of the feature map. In **a** and **b**, we compute the square sum of filter coefficients across the channel dimension, and obtain a spatial energy distribution for the learned filter. **a** The baseline method, which does not consider the local consistency regular term and set $\beta_m, m = \{1, ..., M\}$ as 1. **b** The proposed joint learning formula. Compared to our method, the baseline method learns large coefficients on the background (the left-side region in the bounding box)

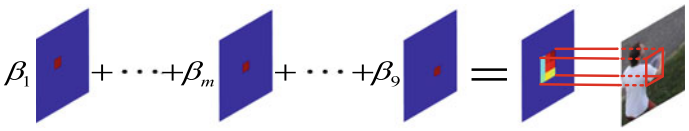


Fig. 6.2 Illustration showing how we compute the reliability map $\mathbf{v}_d$. The computed reliability map only has nonzeros values corresponding to the target region, thus the real positive and negative samples can be generated when we circularly shift the input image

To learn a compact reliability map, we divide the target region into M patches, and use a variable $\beta_m, m \in \{1, ..., M\}$ to denote the reliability for each patch (β_m is shared across the channels), thus $\mathbf{v}_d$ can be written as

$$\mathbf{v}_d = \sum_{m=1}^{M} \beta_m \mathbf{p}_d^m, \tag{6.3}$$

where $\mathbf{p}_d^m \in \mathbb{R}^{K\times 1}$ is a binary mask (see Fig. 6.2) which crops the filter region for the m-th patch.

Based on the previous descriptions, we attempt to jointly learn the base filter $\mathbf{h} = \left[\mathbf{h}_1^\top, ..., \mathbf{h}_D^\top\right]^\top \in \mathbb{R}^{KD\times 1}$ and the reliability vector $\boldsymbol{\beta} = [\beta_1, ..., \beta_M]^\top$ by using the following optimization problem:

$$\begin{aligned} \left[\dot{\mathbf{h}}, \dot{\boldsymbol{\beta}}\right] &= \arg\min_{\mathbf{h},\boldsymbol{\beta}} f\left(\mathbf{h}, \boldsymbol{\beta}; \mathbf{X}\right) \\ s.t.\ \ &\theta_{\min} \le \beta_m \le \theta_{\max},\ \forall m \end{aligned}, \tag{6.4}$$

where the objective function $f\left(\mathbf{h}, \boldsymbol{\beta}; \mathbf{X}\right)$ is defined as

$$f\left(\mathbf{h}, \boldsymbol{\beta}; \mathbf{X}\right) = f_1\left(\mathbf{h}, \boldsymbol{\beta}; \mathbf{X}\right) + \eta f_2\left(\mathbf{h}; \mathbf{X}\right) + \gamma \left\|\mathbf{h}\right\|_2^2. \tag{6.5}$$

In this equation, the first term is the data term with respect to the classification error of training samples, the second term is a regularization term to introduce the local response consistency constraint on the filter coefficient vector $\mathbf{h}$, and the last one is a squared ℓ_2-norm regularization to avoid model degradation. In the optimization problem (6.4), we also add some constraints on the learned reliability coefficients $\beta_1, ..., \beta_M$. These constraints prevent all reliability weights being assigned to a small region of the target especially when the number of training samples is limited, and encourage our model to obtain an accurate weight map. We note that the optimization problem (6.5) encourages learning more reliable correlation filters (see Fig. 6.4 for example).

Data Term: The data term $f_1(\mathbf{h}, \boldsymbol{\beta}; \mathbf{X})$ is indeed a loss function which ensures that the learned filter has a Gaussian shaped response with respect to the circulant sample matrix. By introducing our basic assumption in Eq. (6.3) into the standard CF model, $f_1(\mathbf{h}, \boldsymbol{\beta}; \mathbf{X})$ can be rewritten as

$$\begin{aligned} & f_1(\mathbf{h}, \boldsymbol{\beta}; \mathbf{X}) \\ &= \sum_{k=1}^{K} \left(y_k - \sum_{d=1}^{D} \mathbf{x}_{k,d}^{\top} (\mathbf{v}_d \odot \mathbf{h}_d) \right)^2 \\ &= \sum_{k=1}^{K} \left(y_k - \sum_{d=1}^{D} \mathbf{x}_{k,d}^{\top} \mathbf{V}_d \mathbf{h}_d \right)^2 \\ &= \left\| \mathbf{y} - \sum_{d=1}^{D} \mathbf{X}_d^{\top} \mathbf{V}_d \mathbf{h}_d \right\|_2^2 \\ &= \left\| \mathbf{y} - \mathbf{X}^{\top} \mathbf{V} \mathbf{h} \right\|_2^2 \end{aligned}, \tag{6.6}$$

where $\mathbf{V}_d = \operatorname{diag}(\mathbf{v}_d(1), \mathbf{v}_d(2), ..., \mathbf{v}_d(K)) \in \mathbb{R}^{K \times K}$ is a diagonal matrix, $\mathbf{X}_d = \left[\mathbf{x}_{1,d}, \mathbf{x}_{2,d}, ..., \mathbf{x}_{K,d}\right] \in \mathbb{R}^{K \times K}$ is the circulant matrix of the d-th channel. In addition, $\mathbf{X} = \left[\mathbf{X}_1^{\top}, \mathbf{X}_2^{\top}, ..., \mathbf{X}_D^{\top}\right]^{\top} \in \mathbb{R}^{KD \times K}$ stands for a contactated matrix of all circulant matrices from different channels, and $\mathbf{V} = \mathbf{V}_1 \oplus \mathbf{V}_2 \oplus \cdots \oplus \mathbf{V}_D \in \mathbb{R}^{DK \times DK}$ denotes a block diagonal matrix where $\mathbf{V}_d$ is the d-the diagonal block.

Local Response Consistency: The regularization term $f_2(\mathbf{h}; \mathbf{X})$ constrains that the base filter generates consistent responses for different fragments of the cyclically shifted sample. By this means, the base filter learns equal importance for each local region, and reliability information is separated from the base filter. The term $f_2(\mathbf{h}; \mathbf{X})$ can be defined as

$$\begin{aligned} & f_2(\mathbf{h}; \mathbf{X}) \\ &= \sum_{k=1}^{K} \sum_{m,n}^{M} \left(\sum_{d=1}^{D} \left(\mathbf{P}_d^m \mathbf{x}_{k,d}\right)^{\top} \mathbf{h}_d - \sum_{d=1}^{D} \left(\mathbf{P}_d^n \mathbf{x}_{k,d}\right)^{\top} \mathbf{h}_d \right)^2 \\ &= \sum_{m,n}^{M} \left\| \sum_{d=1}^{D} \mathbf{X}_d^{\top} \mathbf{P}_d^m \mathbf{h}_d - \sum_{d=1}^{D} \mathbf{X}_d^{\top} \mathbf{P}_d^n \mathbf{h}_d \right\|_2^2 \\ &= \sum_{m,n}^{M} \left\| \mathbf{X}^{\top} \mathbf{P}^m \mathbf{h} - \mathbf{X}^{\top} \mathbf{P}^n \mathbf{h} \right\|_2^2 \end{aligned}, \tag{6.7}$$

where $\mathbf{P}_d^m = \text{diag}(\mathbf{p}_d^m(1), \mathbf{p}_d^m(2), ..., \mathbf{p}_d^m(K)) \in \mathbb{R}^{K\times K}, \mathbf{P}^m = \mathbf{P}_1^m \oplus \mathbf{P}_2^m \oplus \cdots \oplus \mathbf{P}_D^m \in \mathbb{R}^{DK\times DK}$. For each cyclically shifted sample $\mathbf{x}_{k,d}$, $\left(\mathbf{P}_d^m \mathbf{x}_{k,d}\right)^\top \mathbf{h}_d$ is the response for the m-th fragment of $\mathbf{x}_{k,d}$.

Joint Discrimination and Reliability Learning: Based on the discussions above, the base filter and the reliability vector can be jointly learned by solving the optimization problem (6.4), which is a non-convex but differentiable problem for both $\mathbf{h}$ and $\boldsymbol{\beta}$. However, it can be converted into a convex differentiable problem if either $\mathbf{h}$ or $\boldsymbol{\beta}$ is known. Thus, in this work, we attempt to solve the optimal $\dot{\mathbf{h}}$ and $\dot{\boldsymbol{\beta}}$ via the alternating direction method.

Solving h: To solve the optimal $\mathbf{h}$, we first compute the derivative of the objective function (6.5) , then by setting the derivative to be zero, we obtain the following normal equations:

$$\mathbf{A}\mathbf{h} = \mathbf{V}^\top \mathbf{X}\mathbf{y}. \tag{6.8}$$

The matrix $\mathbf{A}$ is defined as

$$\begin{aligned}\mathbf{A} = \mathbf{g}(\mathbf{V}, \mathbf{X}) + 2\eta \sum_{m=1}^{M} M\mathbf{g}(\mathbf{P}^m, \mathbf{X}) \\ - 2\eta \mathbf{g}(\sum_{m=1}^{M} \mathbf{P}^m, \mathbf{X}) + \gamma \mathbf{I}\end{aligned}, \tag{6.9}$$

where $\mathbf{g}(\mathbf{\Lambda}, \mathbf{R}) = \mathbf{\Lambda}^\top \mathbf{R}\mathbf{R}^\top \mathbf{\Lambda}$, $\mathbf{R}$ is a circulant matrix and $\mathbf{\Lambda}$ is a diagonal matrix.

In this work, we exploit the conjugate gradient descent method due to its fast convergence speed. The update process can be performed via the following iterative steps [20]:

$$\begin{aligned}\alpha^{(i)} &= \mathbf{r}^{(i)\top}\mathbf{r}^{(i)} / \mathbf{u}^{(i)\top}\mathbf{A}\mathbf{u}^{(i)} \\ \mathbf{h}^{(i+1)} &= \mathbf{h}^{(i)} + \alpha^{(i)}\mathbf{u}^{(i)} \\ \mathbf{r}^{(i+1)} &= \mathbf{r}^{(i)} + \alpha^{(i)}\mathbf{A}\mathbf{u}^{(i)} \\ \mu^{(i+1)} &= \left\|\mathbf{r}^{(i+1)}\right\|_2^2 / \left\|\mathbf{r}^{(i)}\right\|_2^2 \\ \mathbf{u}^{(i+1)} &= -\mathbf{r}^{(i+1)} + \mu^{(i+1)}\mathbf{u}^{(i)}\end{aligned}, \tag{6.10}$$

where $\mathbf{u}^{(i)}$ denotes the search direction at the i-th iteration, $\mathbf{r}^{(i)}$ is the residual after the i-th iteration. Clearly, the computational load lies in the update of $\alpha^{(i)}$ and $\mathbf{r}^{(i+1)}$ since it requires to compute $\mathbf{u}^{(i)\top}\mathbf{A}\mathbf{u}^{(i)}$ and $\mathbf{A}\mathbf{u}^{(i)}$ in each iteration. As shown in Eq. (6.9), the first three terms have the same form. For clarity, we take the first term as an example to show how we compute $\mathbf{u}^{(i)\top}\mathbf{A}\mathbf{u}^{(i)}$ and $\mathbf{A}\mathbf{u}^{(i)}$ efficiently. Let $\mathbf{A}_1$ denote the first term of Eq. (6.9), then

$$\begin{aligned}\mathbf{u}^{(i)\top}\mathbf{A}_1\mathbf{u}^{(i)} &= \mathbf{u}^{(i)\top}\mathbf{V}^\top\mathbf{X}\mathbf{X}^\top\mathbf{V}\mathbf{u}^{(i)} \\ &= \left\|\sum_{d=1}^{D}\mathbf{X}_d^\top\mathbf{V}_d\mathbf{u}_d^{(i)}\right\|_2^2 \\ &= \frac{1}{K}\left\|\sum_{d=1}^{D}\widehat{\mathbf{X}}_d^H \odot \mathscr{F}\left(\mathbf{V}_d\mathbf{u}_d^{(i)}\right)\right\|_2^2\end{aligned}, \tag{6.11}$$

where $\widehat{\mathbf{X}}_d = \mathscr{F}(\mathbf{x}_d)$ is the Fourier transform of the base vector $\mathbf{x}_d$ (corresponding to the input image without shift), $\mathbf{u}_d^{(i)}$ is the subset of $\mathbf{u}^{(i)}$ corresponding to the d-th channel, $(\cdot)^H$ denotes the conjugate of a vector. Because $\mathbf{V}_d\mathbf{u}_d^{(i)}$ is a vector and $\mathbf{X}_d$ is the circulant matrix, the operation $\mathbf{X}_d^\top(\mathbf{V}_d\mathbf{u}_d^{(i)})$ can be viewed as a circular correlation process and can be efficiently computed in the Fourier domain.

Similarly, $\mathbf{A}_1\mathbf{u}^{(i)}$ can be computed as

$$\mathbf{A}_1\mathbf{u}^{(i)} = \mathbf{V}^\top\mathbf{X}\mathbf{X}^\top\mathbf{V}\mathbf{u}^{(i)} = \begin{bmatrix} \mathbf{V}_1^\top\mathbf{X}_1 \sum_{j=1}^{D} \mathbf{X}_j^\top\mathbf{V}_j\mathbf{u}_j^{(i)} \\ \mathbf{V}_2^\top\mathbf{X}_2 \sum_{j=1}^{D} \mathbf{X}_j^\top\mathbf{V}_j\mathbf{u}_j^{(i)} \\ \cdots \\ \mathbf{V}_D^\top\mathbf{X}_D \sum_{j=1}^{D} \mathbf{X}_j^\top\mathbf{V}_j\mathbf{u}_j^{(i)} \end{bmatrix}. \tag{6.12}$$

The d-th term $\mathbf{V}_d^\top\mathbf{X}_d \sum_{j=1}^{D} \mathbf{X}_j^\top\mathbf{V}_j\mathbf{u}_j^{(i)}$ can be computed as

$$\begin{aligned} &\mathbf{V}_d^\top\mathbf{X}_d \sum_{j=1}^{D} \mathbf{X}_j^\top\mathbf{V}_j\mathbf{u}_j^{(i)} \\ &= \mathbf{V}_d^\top\mathscr{F}^{-1}\left(\widehat{\mathbf{X}}_d \odot \left(\sum_{j=1}^{D} \widehat{\mathbf{X}}_j^H \odot \mathscr{F}(\mathbf{V}_j\mathbf{u}_j^{(i)})\right)\right), \end{aligned} \tag{6.13}$$

where $\widehat{\mathbf{X}}_j^H \odot \mathscr{F}(\mathbf{V}_j\mathbf{u}_j^{(i)})$ has been computed in Eq. (6.11) and can be directly used. The computational complexities of (6.11) and (6.12) are therefore $\mathscr{O}(DK\log K)$.

Solving $\boldsymbol{\beta}$: If the filter vector $\mathbf{h}$ is given, the reliability weight vector $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_M]^\top$ can be obtained by solving the following optimization problem:

$$\begin{aligned} &\dot{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\| \mathbf{y} - \sum_{d=1}^{D} \mathbf{X}_d^\top\mathbf{V}_d\mathbf{h}_d \right\|_2^2, \\ &s.t. \quad \theta_{\min} \le \beta_m \le \theta_{\max}, \ \forall m \end{aligned} \tag{6.14}$$

where the term $f_2(\mathbf{h}; \mathbf{X})$ is ignored as it does not include $\boldsymbol{\beta}$. With some derivations, the problem (6.14) can be converted as follows:

$$\begin{aligned} &\dot{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \boldsymbol{\beta}^\top\mathbf{C}^\top\mathbf{C}\boldsymbol{\beta} - 2\boldsymbol{\beta}^\top\mathbf{C}^\top\mathbf{y}, \\ &s.t. \quad \theta_{\min} < \beta_m < \theta_{\max}, \ \forall m \end{aligned} \tag{6.15}$$

where $\mathbf{C} = \left[\mathbf{C}^1, ..., \mathbf{C}^M\right] \in \mathbb{R}^{K\times M}$, and $\mathbf{C}^m$ is computed as $\mathbf{C}^m = \mathscr{F}^{-1}(\sum_{d=1}^{D} \widehat{\mathbf{X}}_d^H \odot \mathscr{F}(\mathbf{P}_d^m\mathbf{h}_d))$, whose computational complexity is $\mathscr{O}(DK\log(K))$. This optimization

problem is a convex quadratic programming method, which can be efficiently solved via standard quadratic programming.

Model Extension
We note the proposed model and its optimization method are defined and derived based on the training sample in one frame. Recent studies (like [10]) demonstrate that it is more effective to learn the correlation filter using a set of training samples from multiple frames. Thus, we can extend our optimization problem (6.4) to consider multi-frame information as follows:

$$\begin{aligned} \left[\dot{\mathbf{h}}, \dot{\boldsymbol{\beta}}\right] &= \arg\min_{\mathbf{h},\boldsymbol{\beta}} \sum_{t=1}^{T} \kappa_t f(\mathbf{h}, \boldsymbol{\beta}; \mathbf{X}^t), \\ s.t. \quad & \theta_{\min} < \beta_m < \theta_{\max}, \quad \forall m \end{aligned} \tag{6.16}$$

where $\mathbf{X}^t$ denotes the sample matrix in the t-th frame, κ_t is a temporal weight to indicate the contribution of the t-th frame. It is not difficult to prove that the previous derivations can be applicable for solving the optimization problem (6.16).

In Fig. 6.3, we provide examples showing that our tracker can accurately learn the reliability value for each patch region. In the first row, the left part of the frisbee is frequently occluded, our method learns a small reliability value for such regions. The example in the second row demonstrates that our method can accurately determine that the fast moving legs are not reliable. In the last example, the background regions are associated with small weights via the proposed model, thereby facilitating the further tracking process.

Joint Discrimination and Reliability Model Update: Most correlation filter based tracking algorithms perform model update in each frame, which results in high computation load. Recently, the ECO method proves that the sparse update mechanism is a better choice for the CF-based trackers. Following the ECO method, we also exploit the sparse update mechanism in our tracker. In the update frame, we use the conjugate gradient descent method to update the base filter coefficient vector $\mathbf{h}$ and then we update $\boldsymbol{\beta}$ based on the updated base filter by solving a quadratic programming problem. In each frame, we initialize the weight for the training frame as ω and weights of previous training samples are decayed as $(1-\omega)\kappa_t$. When the number of training samples exceeds the predefined value $T_{\max}$, we follow the ECO method and use the Gaussian Mixture Model (GMM) for sample fusion. We refer the readers to [4] for more details.

Target Localization: In the detection process at the t-th frame, we use a multi-scale search strategy [4, 10] for joint target localization and scale estimation. We extract the ROI regions with different scales centered in the estimated position of last frame, and obtain the multichannel feature map $\mathbf{x}_d^s$, $d = \{1, ..., D\}$, $s = \{1, ..., S\}$ for the ROI region, where s is the scale index. Then we compute the response for the target localization in scale s as

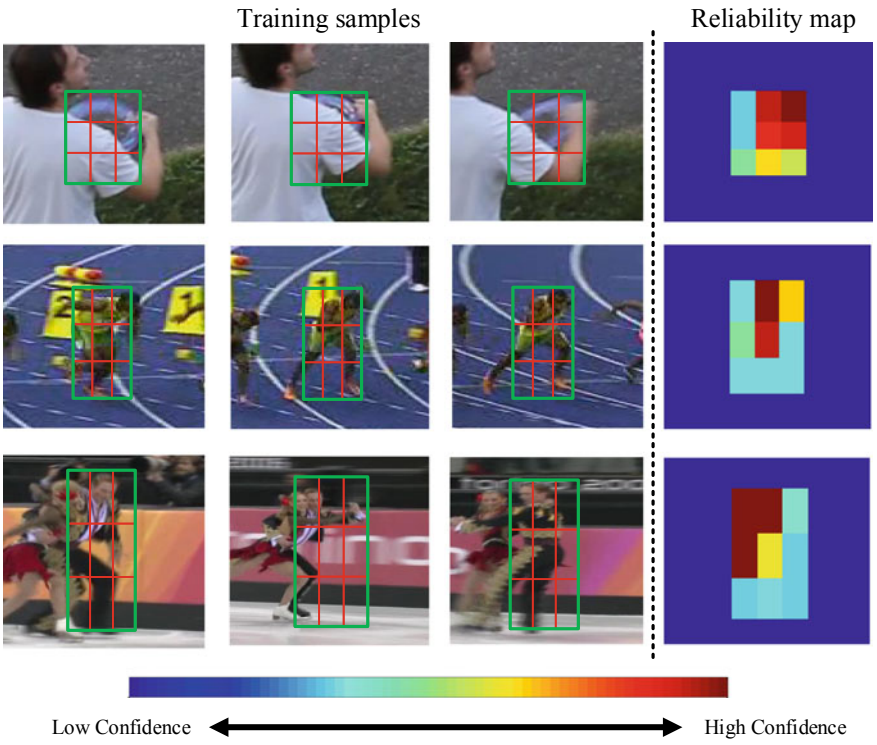


Fig. 6.3 Illustration showing that reliable regions can be determined through the joint learning formula. We show three example training samples on the left three columns, and show the learned reliable weight maps on the fourth column

$$\mathbf{r}_s = \sum_{d=1}^{D} \mathscr{F}^{-1}(\mathscr{F}(\mathbf{w}_d) \odot (\mathscr{F}(\mathbf{x}_d^s))^H). \tag{6.17}$$

The target location and scale are then jointly determined by finding the maximum value in the S response maps. This joint estimation strategy shows better performance than the previous methods, which first estimate the target position and then refine the scale based on the estimated position.

Implementation Details: The proposed DRT method exploits an ensemble of deep (Conv1 from VGG-M, Conv4-3 from VGG-16 [3]) and handcrafted (HOG and Color Names) features for target representation. In our tracker, we use a relatively small learning rate ω (0.011) for first 10 frames to avoid model degradation with limited training samples, and use a larger one (0.02) in the following tracking process. The maximum number of training samples $T_{\max}$ and the number of fragments as set as 50 and 9, respectively. As to the online joint learning formula, the trade-off parameter η for the local consistency term is set as 1 by default and $\theta_{\min}$

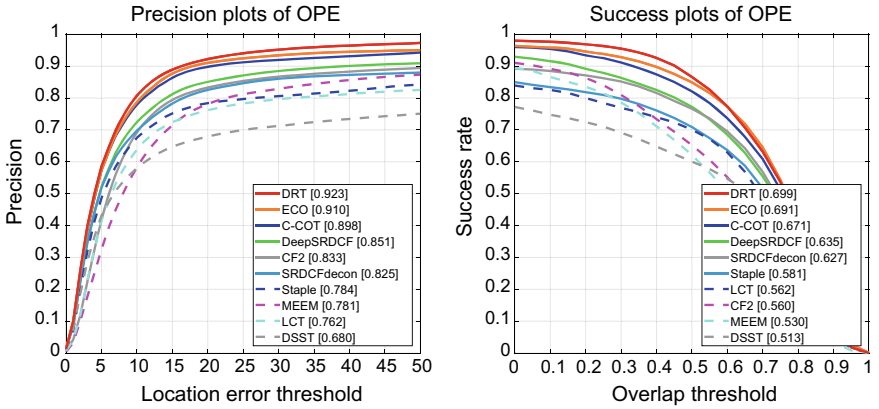


Fig. 6.4 Precision and success plots of different trackers on the OTB-2015 datasets in terms of the OPE rule. This figure only shows the plots of top 10 trackers for clarity. In the legend behind of name of each tracker, we show the distance precision score at the threshold on 20 pixels and the area under curve (AUC) score

and $\theta_{\max}$ are set as 0.5 and 1.5 respectively. One implementation of our tracker can be found in https://github.com/cswaynecool/DRT.

6.3 Experimental Results and Discussions

Performance Evaluation OTB-2015 Dataset: The OTB-2015 dataset [25] is an extension of the OTB-2013 dataset, and contains 50 more video sequences. We also evaluate the performance of the proposed DRT method over all 100 videos in this dataset. In our experiment, we compare with 29 default trackers in [25] and other 9 state-of-the-art trackers including ECO [4], C-COT [10], Staple [1], CF2 [18], DeepSRDCF [6], SRDCFDecon [7], LCT [19], DSST [5] and MEEM [26].

Figure 6.4 reports both precision and success plots of different trackers in terms of the OPE rule. Overall, our DRT method provides the best result with a distance precision rate of 92.3% and with an AUC score of 69.9%, which again achieves a substantial improvement of several outstanding trackers (e.g., ECO, C-COT and DeepSRDCF).

We also evaluate the performance of trackers in various attributes. The OTB-2013 dataset is divided into 11 attributes, each of which corresponds to a challenging factor (e.g., illumination, deformation and scale change). Tables 6.1 and 6.2 illustrate the overlap success scores and precision scores of the top 10 algorithms on 9 attributes. We can see that our tracker achieves the best performance in all these attributes. Specially, the proposed method improves the success scores of the second best tracker ECO by 0.8%, 1.7%, 2.1% and 1.8% in the attributes of deformation, background clutter, illumination variation and scale variation, respectively. These results validate

Table 6.1 The success comparison of mentioned trackers

Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
DRT	***0.717***	***0.641***	***0.687***	***0.664***	***0.734***	***0.655***	***0.719***	***0.687***	***0.684***	*0.651*	***0.687***
ECO	**0.700**	**0.633**	**0.678**	**0.655**	**0.713**	**0.617**	**0.718**	**0.680**	**0.673**	***0.660***	**0.669**
C-COT	*0.645*	*0.615*	*0.672*	*0.624*	*0.676*	*0.608*	*0.714*	*0.671*	*0.645*	**0.655**	*0.659*
SRDC Fdecon	0.641	0.553	0.604	0.573	0.646	0.518	0.653	0.589	0.591	0.510	0.611
CF2	0.580	0.530	0.549	0.557	0.537	0.424	0.568	0.523	0.532	0.474	0.486
Deep SRDCF	0.627	0.566	0.625	0.589	0.621	0.474	0.656	0.601	0.607	0.553	0.609
Staple	0.574	0.554	0.535	0.552	0.589	0.411	0.558	0.548	0.534	0.481	0.529
MEEM	0.519	0.489	0.525	0.529	0.517	0.355	0.545	0.504	0.525	0.488	0.474
LCT	0.550	0.499	0.522	0.557	0.566	0.330	0.532	0.507	0.538	0.452	0.492
DSST	0.523	0.420	0.453	0.502	0.558	0.379	0.488	0.453	0.470	0.386	0.475

Table 6.2 The precision comparison of mentioned trackers

Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
DRT	***0.952***	***0.877***	***0.870***	***0.902***	***0.938***	***0.897***	***0.905***	***0.918***	***0.922***	*0.874*	***0.901***
ECO	**0.942**	*0.860*	*0.865*	**0.892**	**0.914**	**0.888**	**0.904**	**0.908**	**0.907**	***0.913***	**0.881**
C-COT	*0.866*	**0.872**	**0.867**	*0.867*	*0.874*	*0.876*	*0.903*	*0.902*	*0.892*	**0.891**	*0.875*
SRDC Fdecon	0.850	0.754	0.768	0.776	0.835	0.672	0.826	0.768	0.797	0.641	0.808
CF2	0.830	0.791	0.782	0.846	0.806	0.787	0.783	0.759	0.801	0.677	0.796
Deep SRDCF	0.841	0.783	0.805	0.818	0.791	0.702	0.837	0.825	0.835	0.781	0.822
Staple	0.766	0.748	0.696	0.770	0.791	0.609	0.726	0.726	0.738	0.661	0.731
MEEM	0.746	0.754	0.728	0.794	0.740	0.605	0.722	0.741	0.794	0.685	0.740
LCT	0.734	0.689	0.667	0.782	0.746	0.490	0.673	0.682	0.746	0.592	0.686
DSST	0.704	0.542	0.562	0.691	0.721	0.550	0.595	0.597	0.644	0.481	0.643

that our method is effective in handling such challenges. When the object suffers from large deformations, parts of the target object will be not reliable. Thus, it is crucial to conduct accurate reliability learning in dealing with this case. Since our joint learning formula is insusceptible to the feature map response distributions, it can learn the reliability score for each region more accurately. Similarly, influenced by the cluttered background and abrupt illumination change, the feature map inevitably highlights the background or unreliable regions in the image. Existing CF-based algorithms learn large filter weights in such regions, thereby resulting in the tracking failure. In addition, these trackers usually assign most filter weights to the learned dominant regions and ignore certain parts of the target object, which leads to inferior scale estimation performance. By joint discrimination and reliability learning, the

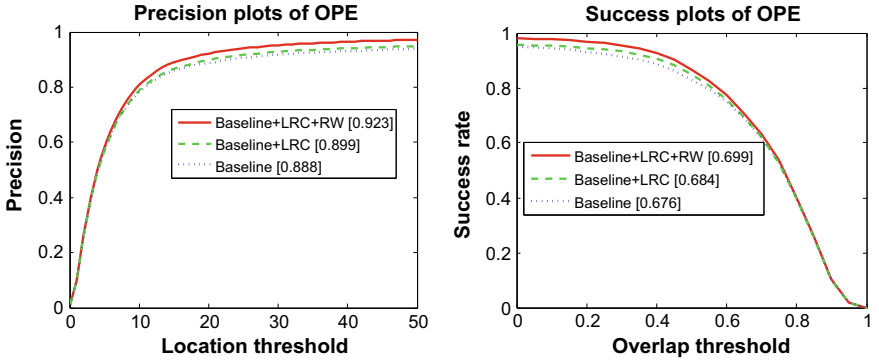


Fig. 6.5 Performance evaluation for each component of the proposed method

proposed DRT method is robust to numerous challenges and therefore achieves a remarkable performance in comparison with other ones.

Ablation Studies: In this section, we test effectiveness for each component of the proposed joint learning formula on both the OTB-2015 datasets. First, we use the notation "Baseline" to denote the baseline method which does not exploit the local consistency constraint and the reliability map ($\beta_m = 1, m \in \{1, ..., M\}$). Like the conventional correlation filter, the baseline method does not separate the discrimination and reliability information. In addition, we also use the notation "Baseline+LRC" to denote the modified baseline tracker with the local response consistency constraint. The "Baseline+LRC" method focuses on learning the discrimination information while ignoring the reliability information of the target. The abbreviation "RW" stands for reliability weight map and "Baseline+LRC+RW" denotes the proposed joint learning method. In Fig. 6.5, we show that the proposed joint learning formula improves the baseline method by 3.5 and 2.3% on the OTB-2015 dataset in terms of the distance precision rate and the AUC score. By comparing our method with "Baseline+LRC", we show the effectiveness of the reliability learning process. Considering the reliability learning, our tracker improves the "Baseline+LRC" method by 1.5% in terms of AUC score on the OTB-2015 dataset.

Failure Cases: We show some failure cases of the proposed tracker in Fig. 6.6. In the first and third columns, the cluttered background regions contain numerous distractors, which causes the proposed method to drift off the targets. In the second column, the proposed method does not track the target object well as it undergoes large deformations and rotations in a short span of time. These tracking failures can be partially addressed when the information of the optical flow is considered, which will be the focus of our future work.

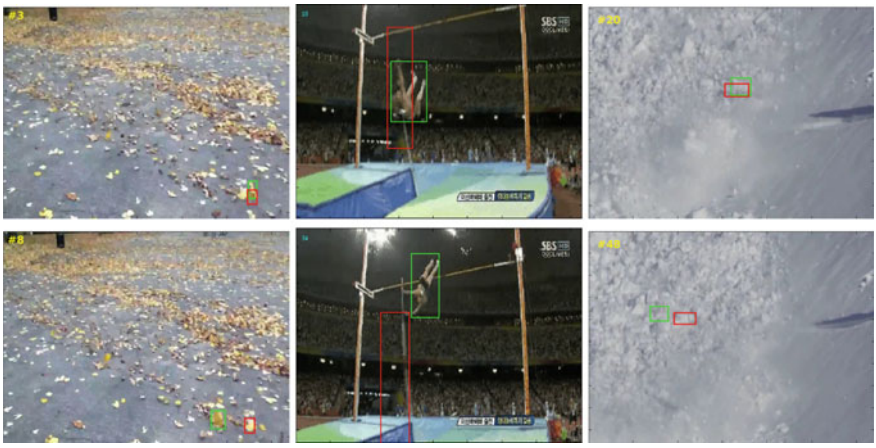


Fig. 6.6 Failure cases of the proposed method, where we use red and green bounding boxes to denote our results and ground truths

6.4 Summary

We consider the discrimination and reliability information in the correlation filter (CF) formula and rewrite the filter weight as the product of a base filter and a reliability weight map. A local response consistency constraint is introduced for the base filter, which constrains that each subregion of the target has similar importance. By this means, the reliability information is separated from the base filter. Besides, the reliability information in the filter is jointly learned with the base filter. Compared to the existing CF-based methods, our tracker is insusceptible to the nonuniform distributions of the feature map, and can better suppress the background regions. The joint learning of the base filter and reliability term can be performed by solving the proposed optimization problem and being speeded up in the Fourier domain.

References

1. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.: Staple: Complementary learners for real-time tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
2. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: IEEE Conference on Computer Vision and Pattern Recognition (2010)
3. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: delving deep into convolutional nets. In: British Machine Vision Conference (2014)
4. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: Eco: Efficient convolution operators for tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
5. Danelljan, M., Häger, G., Khan, F., Felsberg, M.: Accurate scale estimation for robust visual tracking. In: British Machine Vision Conference (2014)

6. Danelljan, M., Hager, G., Shahbaz Khan, F., Felsberg, M.: Convolutional features for correlation filter based visual tracking. In: IEEE Conference on International Conference on Computer Vision Workshops (2015)
7. Danelljan, M., Hager, G., Shahbaz Khan, F., Felsberg, M.: Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
8. Danelljan, M., Khan, F.S., Felsberg, M., van de Weijer, J.: Adaptive color attributes for real-time visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1090–1097 (2014)
9. Danelljan, M., Robinson, A., Khan, F.S., Felsberg, M.: Beyond correlation filters: learning continuous convolution operators for visual tracking. In: European Conference on Computer Vision (2016)
10. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: Exploiting the circulant structure of tracking-by-detection with kernels. In: European Conference on Computer Vision (2012)
11. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Trans. Pattern Anal. Mach. Intell. **37**(3), 583–596 (2015)
12. Kiani Galoogahi, H., Fagg, A., Lucey, S.: Learning background-aware correlation filters for visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
13. Li, F., Yao, Y., Li, P., Zhang, D., Zuo, W., Yang, M.: Integrating boundary and center correlation filters for visual tracking with aspect ratio variation. In: IEEE International Conference on Computer Vision Workshop (2017)
14. Li, Y., Zhu, J.: A scale adaptive kernel correlation filter tracker with feature integration. In: European Conference on Computer Vision Workshop, pp. 254–265 (2014)
15. Li, Y., Zhu, J., Hoi, S.C.: Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
16. Liu, S., Zhang, T., Cao, X., Xu, C.: Structural correlation filter for robust visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
17. Liu, T., Wang, G., Yang, Q.: Real-time part-based visual tracking via adaptive correlation filters. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
18. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: IEEE Conference on International Conference on Computer Vision (2015)
19. Ma, C., Yang, X., Zhang, C., Yang, M.H.: Long-term correlation tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
20. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York, USA (2006)
21. Sun, C., Wang, D., Lu, H., Yang, M.: Correlation tracking via joint discrimination and reliability learning. CoRR abs/1804.08965 (2018)
22. Sun, C., Wang, D., Lu, H., Yang, M.: Correlation tracking via joint discrimination and reliability learning. In: CVPR, pp. 489–497 (2018)
23. Wang, L., Ouyang, W., Wang, X., Lu, H.: Visual tracking with fully convolutional networks. In: IEEE Conference on International Conference on Computer Vision (2015)
24. Wang, L., Ouyang, W., Wang, X., Lu, H.: Stct: Sequentially training convolutional networks for visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
25. Wu, Y., Lim, J., Yang, M.H.: Object tracking benchmark. IEEE Trans. Pattern Anal. Mach. Intell. **37**(9), 1834–1848 (2015)
26. Zhang, J., Ma, S., Sclaroff, S.: Meem: robust tracking via multiple experts using entropy minimization. In: European Conference on Computer Vision (2014)

Chapter 7
Visual Tracking Based on Deep Learning

Deep neural network gains popularity in visual tracking. In this chapter, we introduce three deep trackers (FCNT [36], StructSiam [41], ACT [5]). The FCNT method complements two convolutional layers of different levels in handling drastic appearance change and distinguishing target object from its similar distractors. The StructSiam method exploits the local structure within deep networks, which simultaneously considers the local patterns of the target and their structural relationships for more accurate target tracking. Finally, the "Actor-Critic" tracking framework attempts to study the tracking problem based on deep reinforcement learning.[1,2,3]

7.1 Introduction

Traditional tracking algorithms usually focus on developing robust appearance model from the perspectives of handcrafted features, online learning algorithms, or both. Some milestones include IVT [29], MIL [2], TLD [19], APGL1 [3], SCM [42], ASLAS [18], STRUCK [14], and KCF [39]. However, the reports on large-scale benchmark evaluations (OTB-2015 [38]) demonstrate that the performance of these traditional algorithms is far from the requirement of realistic applications.

Over the last five years, deep learning [21] have achieved an impressive suite of results thanks to their success on automatic feature extraction via multi-layer nonlinear transformations, especially in computer vision, speech recognition, and natural language processing. Motivated by these breakthroughs, several deep-learning-based

[1] ©2015 IEEE, Reprinted, with permission, from Ref. [36].

[2] ©Reprinted by permission from [Springer Nature]: [Springer] [ECCV] [Structured Siamese Network for Real-Time Visual Tracking, Yunhua Zhang, Lijun Wang, Jinqing Qi, Huchuan Lu, 2018.).

[3] ©Reprinted by permission from [Springer Nature]: [Springer] [ECCV] [Real-Time Actor-Critic Tracking, Boyu Chen, Dong Wang, Peixia Li, Chong Sun, Huchuan Lu, 2018.).

H. Lu and D. Wang, *Online Visual Tracking*,
https://doi.org/10.1007/978-981-13-0469-9_7

trackers (e.g., FCNT [36], STCT [37], DNT, LSART [34]) have demonstrated the potential advantages for significantly improving the tracking performance. As a result, the performance on the OTB-100 [38] dataset is constantly refreshed by the tracking methods based on deep learning. The MDNet [27] tracker is the winner of VOT2015 [20] competition. All the top-4 trackers in the VOT2016 competition, including C-COT [11], TCNN [26], SSAT[4] and MLDF,[5] are based on deep neural networks.

Most of the existing trackers based on deep learning use the convolutional neural network (CNN). The CNN model is very suitable for developing robust appearance model in the tracking task, due to its powerful ability on feature extraction and image classification. Similar to the traditional trackers, the CNN-based tracking methods can also be either discriminative or generative. The discriminative method (e.g., DeepSRDCF [10], RPNT [43], C-COT [11]) aims to conduct a binary classification with the CNN model for effectively distinguishing the tracked object from its surrounding backgrounds. Whereas the generative one (e.g., SINT [35], SiameseFC [4], YCNN [6]) focuses on learning a robust similarity function to accurately match the object template within a given search region. In addition, few trackers (e.g.,RTT [7], SANet [13], ROLO [28]) use the recurrent neural network (RNN). The RNN model is suitable for sequence modeling since its neuron's output can be directly applied to itself in the next time.

The main content of this section is as follows.

First, we analyze CNN features learned from the large-scale image classification task, find important properties for online tracking, and then propose a new tracking method. The proposed tracker complements two convolutional layers of different levels in handling drastic appearance change and distinguishing target object from its similar distractors. This design significantly mitigates drifts.

Second, we develop a structured Siamese network by proposing a local structure learning method, which simultaneously considers the local patterns of the target and their structural relationships for more accurate target tracking. To this end, a local pattern detection module is designed to automatically identify discriminative regions of the target objects. The detection results are further refined by a message passing module, which enforces the structural context among local patterns to construct local structures. By considering various combinations of the local structures, our tracker is able to form various types of structure patterns. The tracking process is finally achieved by a matching procedure of the structure patterns between the target templates and candidates.

Finally, we propose a novel tracking algorithm based on the "Actor-Critic framework". This framework consists of two major components: "Actor" and "Critic". The "Actor model aims to infer the optimal choice in a continuous action space, which directly makes the tracker move the bounding box to the object's location in the current frame. For offline training, the "Critic model" is introduced to form an "Actor-Critic" framework with reinforcement learning and outputs a Q-value to

[4]SSAT is an extended version of MDNet [27].

[5]MLDF is designed based on FCNT [36] and STCT [37].

guide the learning process of both "Actor" and "Critic" deep networks. Then, we modify the original deep deterministic policy gradient algorithm to effectively train our "Actor-Critic" model for the tracking task. For online tracking, the "Actor" model provides a dynamic search strategy to locate the tracked object efficiently and the "Critic" model acts as a verification module to make our tracker more robust.

7.2 Visual Tracking Based on Convolutional Networks

Analysis of deep representations is important to understand the mechanism of deep learning. However, it is still very rare for the purpose of visual tracking. In this section, we present some important properties of CNN features which can better facilitate visual tracking. Our feature analysis is conducted based on the 16-layer VGG network [32] pretrained on the ImageNet image classification task [12], which consists of 13 convolutional layers followed by 3 fully connected layers. We mainly focus on the conv4-3 layer (the 10-th convolutional layer) and the conv5-3 layer (the 13-th convolutional layer), both of which generate 512 feature maps. An illustration of deep feature analysis is presented in Fig. 7.1.

Observation 1 Although the receptive field of CNN feature maps is large, the activated feature maps are sparse and localized. The activated regions are highly correlated to the regions of semantic objects.

Due to pooling and convolutional layers, the receptive fields of the conv4-3 and conv5-3 layers are very large (92×92 and 196×196 pixels, respectively). Figure 7.2 shows some feature maps with the maximum activation values in the object region. It can be seen that the feature maps have only small regions with nonzero values. These nonzero values are localized and mainly correspond to the image region of foreground objects. We also use the approach in [31] to obtain the saliency maps of CNN features. The saliency maps in Fig. 7.2 (bottom row) show that the change of input that results in the largest increase of the selected feature maps are located within the object regions. Therefore, the feature maps are capturing the visual representation related to the objects. These evidences indicate that CNN features learned from image classification are localized and correlated to the object visual cues. Thus, these CNN features can be used for target localization.

Observation 2 Many CNN feature maps are noisy or unrelated to the task of discriminating a particular target from its background.

The CNN features pretrained on ImageNet can describe a large variety of generic objects and therefore they are supposed to detect abundant visual patterns with a large number of neurons. However, when tracking a particular target object, it should focus on a much smaller subset of visual patterns which well separate the target from background. As illustrated in Fig. 7.1c, the average of all the feature maps is cluttered with background noise. And we should discard feature maps that have high response in both target region and background region so that the tracker does not drift to the background regions. Figure 7.3 shows the histograms of the activation values

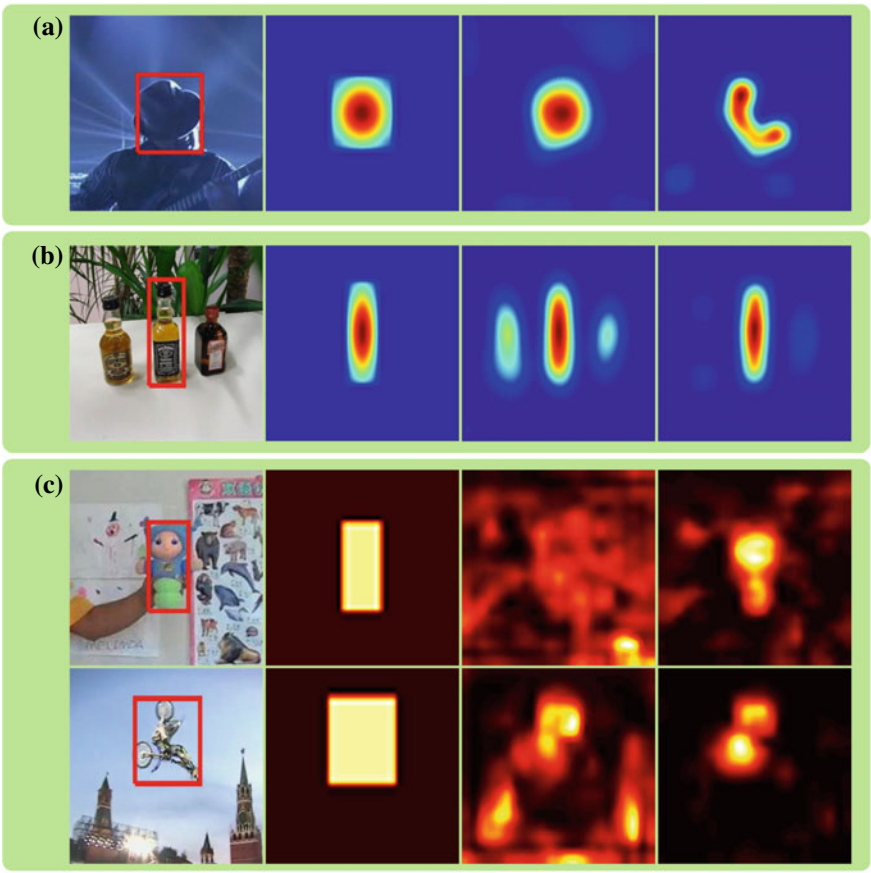


Fig. 7.1 Feature maps for target localization. **a**, **b** From left to right: input image, the ground truth target heat map, the predicted heat maps using feature maps of conv5-3 and conv4-3 layers of VGG network [32]. **c** From left to right: input image, ground truth foreground mask, average feature maps of conv5-3 (top) and conv4-3 (bottom) layers, average selected feature maps conv5-3 (top) and conv4-3 (bottom) layers

for all the feature maps within the object region. The activation value of a feature map is defined as the sum of its responses in the object region. As demonstrated in Fig. 7.3, most of the feature maps have small or zero values within the object region. Therefore, there are lots of feature maps that are not related to the target object. This property provides us the possibility of selecting only a small number of feature maps with small degradation in tracking performance.

Observation 3 Different layers encode different types of features. Higher layers capture semantic concepts on object categories, whereas lower layers encode more discriminative features to capture intra-class variations.

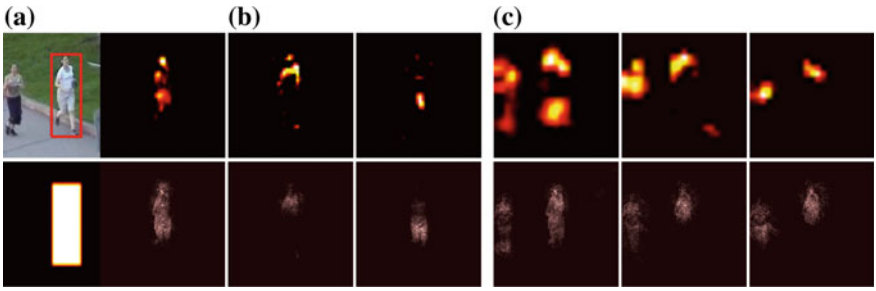


Fig. 7.2 CNNs trained on image classification task carry spatial configuration information. **a** Input image (top) and ground truth foreground mask. **b** Feature maps (top row) of conv4-3 layer which are activated within the target region and are discriminative to the background distracter. Their associated saliency maps (bottom row) are mainly focused on the target region. **c** Feature maps (top row) of conv5-3 layer which are activated within the target region and capture more semantic information of the category (both the target and background distracter). Their saliency maps (bottom row) present spatial information of the category

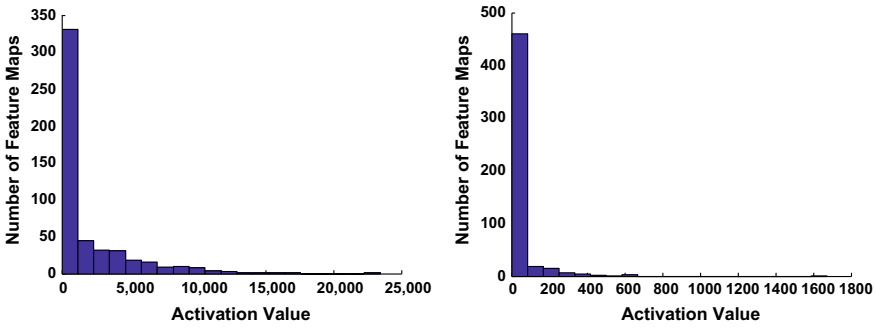


Fig. 7.3 Activation value histograms of feature maps in conv4-3 (left) and conv5-3 (right)

Because of the redundancy of feature maps, we employ a sparse representation scheme to facilitate better visualization. By feeding forward an image of an object through the VGG network, we obtain the feature maps $\mathbf{F} \in \mathbb{R}^{d \times n}$ of a convolutional layer (either the conv4-3 or conv5-3 layer), where each feature map is reshaped into a d-dimensional vector and n denotes the number of feature maps. We further associate the image with a foreground mask $\boldsymbol{\pi} \in \mathbb{R}^{d \times 1}$, where the i-th element $\pi_i = 1$ if the i-th neuron of each feature map is located within the foreground object, and $\pi_i = 0$, otherwise. We reconstruct the foreground mask using a subset of the feature maps by solving

$$\begin{aligned} &\min_{\mathbf{c}} \|\boldsymbol{\pi} - \mathbf{F}\mathbf{c}\|_2^2 + \lambda \|\mathbf{c}\|_1, \\ &\text{s.t. } \mathbf{c} \succeq 0, \end{aligned} \tag{7.1}$$

where $\mathbf{c} \in \mathbb{R}^{n \times 1}$ is the sparse coefficient vector, and λ is the parameter to balance the reconstruction error and sparsity.

Fig. 7.4 The first and the third rows are input images. The second and the fourth rows are reconstructed foreground masks using conv5-3 feature maps. The sparse coefficients are computed using the images in the first column and directly applied to the other columns without change

Figure 7.4 shows some of the reconstructed foreground masks using the feature maps of conv5-3. For the two examples (face and motorcycle) in Fig. 7.4, we only compute the sparse coefficients for images in the first column and use the coefficients to reconstruct foreground masks for the rest of the columns. The selected feature maps in Fig. 7.4a capture the semantic concepts of human faces and are robust to faces with appearance variation and even identity change. The selected feature maps in Fig. 7.4b accurately separate the target from cluttered background and are invariant to pose variation and rotation. Although trained on the image classification task, the high-level semantic representation of object categories encoded by the conv5-3 layer enables object localization. However, these features are not discriminative enough to differentiate objects of the same category, thus they can not be directly applied to visual tracking.

Compared with the conv5-3 feature maps, the features captured by conv4-3 are more sensitive to intra-class appearance variation. In Fig. 7.2, the selected feature maps of conv4-3 can well separate the target person from the other nontarget person. Besides, different feature maps focus on different object parts.

To further verify this, we conduct two quantitative experiments. 1800 human face images belonging to six identities and 2000 images containing non-face objects are collected from the benchmark sequences [39]. Each image is associated with a

Table 7.1 Face classification accuracy using different feature maps. Experiment 1 is to classify face and non-face. Experiment 2 is classify face identities

Feature map	Experiment 1 (%)	Experiment 2 (%)
conv4-3	76.42	83.83
conv5-3	89.56	57.28

foreground mask to indicate the region of the foreground object. In the first experiment, we evaluate the accuracy in classifying the images into face and non-face using the conv4-3 and conv5-3 layers separately. Three face images belonging to three identities are selected as positive training samples to compute a set of sparse coefficients $\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_2\}$ via Eq. (7.1). At the test stage, given the feature maps $\mathbf{F}$ and the foreground mask $\boldsymbol{\pi}$ of an input image, the reconstruction error e for the foreground map is computed by

$$e = \min_i \|\boldsymbol{\pi} - \mathbf{F}\mathbf{c}_i\|_2^2. \tag{7.2}$$

The image is classified as a face image if its reconstruction error e is less than a predefined threshold. Otherwise, it is classified as a non-face image.

In the second experiment, our task is to classify all the face images into different identities. For each identity, 20 images are used as the training samples to learn the sparse coefficients $\mathbf{c}_i$, $i = 1, 2, \ldots, 6$ using Eq. (7.1). At the test stage, the foreground mask reconstruction error for each identity is calculated, and the test image is classified as the identity that has the minimum error as follows:

$$id = \arg\min_i \|\boldsymbol{\pi} - \mathbf{F}\mathbf{c}_i\|_2^2. \tag{7.3}$$

The classification accuracy using the feature maps of conv4-3 and conv5-3 for the two experiments are demonstrated in Table 7.1. The feature maps of conv5-3 encode high-level semantic information and can better separate face from non-face objects. But they achieve lower accuracy than the features maps of conv4-3 in discriminating one identity from another. The feature maps of conv4-3 preserve more middle-level information and enable more accurate classification of face images belonging to different identities. But they are worse than the feature maps of conv5-3 in discriminating human face from non-face objects. These results motivate us to consider these two layers jointly for more robust tracking.

Tracking with Fully Convolutional Networks An overview (Fig. 7.5) of the proposed fully convolutional network based tracker (FCNT) is as follows:

1. For a given target, a feature map selection process is performed on the conv4-3 and conv5-3 layers of the VGG network to select the most relevant feature maps and avoid overfitting on noisy ones.
2. A general network (GNet) that captures the category information of the target is built on top of the selected feature maps of the conv5-3 layer.

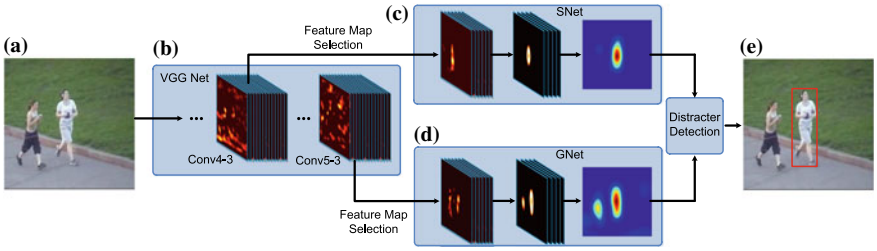


Fig. 7.5 Pipeline of our algorithm. **a** Input ROI region. **b** VGG network. **c** SNet. **d** GNet. **e** Tracking results

3. A specific network (SNet) that discriminates the target from background with similar appearance is built on top of the selected feature maps of the conv4-3 layer.
4. Both GNet and SNet are initialized in the first frame to perform foreground heat map regression for the target and adopt different online update strategies.
5. For a new frame, a region of interest (ROI) centered at the last target location containing both target and background context is cropped and propagated through the fully convolutional network.
6. Two foreground heat maps are generated by GNet and SNet, respectively. Target localization is performed independently based on the two heat maps.
7. The final target is determined by a distractor detection scheme that decides which heat map in step 6 to be used.

Feature Map Selection The proposed feature map selection method is based on a target heat map regression model, named as sel-CNN, and is conducted independently on the conv4-3 and conv5-3 layers of VGG. The sel-CNN model consists of a dropout layer followed by a convolutional layer without any nonlinear transformation. It takes the feature maps (conv4-3 or conv5-3) to be selected as input to predict the target heat map $\mathbf{M}$, which is a 2-dimensional Gaussian centered at the ground truth target location with variance proportional to the target size (See Fig. 7.1a, b for an example). The model is trained by minimizing the square loss between the predicted foreground heat map $\hat{\mathbf{M}}$ and the target heat map $\mathbf{M}$:

$$L_{sel} = \|\hat{\mathbf{M}} - \mathbf{M}\|^2. \tag{7.4}$$

After parameter learning using back-propagation converges, we fix the model parameters and select the feature maps according to their impacts on the loss function. The input feature maps $\mathbf{F}$ are vectorized into a vector denoted by $vec(\mathbf{F})$. Denote f_i as the i-th element of $vec(\mathbf{F})$. The change of the loss function caused by the perturbation of the feature map $\delta\mathbf{F}$ can be computed by a two-order Taylor expansion as follows:

$$\delta L_{sel} = \sum_i g_i \delta f_i + \frac{1}{2}\sum_i h_{ii}(\delta f_i)^2 + \frac{1}{2}\sum_{i\neq j} h_{ij}\delta f_i \delta f_j, \tag{7.5}$$

where $g_i = \frac{\partial L_{sel}}{\partial f_i}$ and $h_{ij} = \frac{\partial^2 L_{sel}}{\partial f_i \partial f_j}$ are, respectively, the first and second order derivatives of the objective function with respect to the input feature maps. The number of elements in the feature maps is very large (>270,000). The complexity for computing all the second order derivatives h_{ij} is $O(270,000^2)$, which is too time consuming. We approximate the Hessian matrix with a diagonal matrix, in which the third term of the right hand side of Eq. (7.5) is neglected. Both the first derivatives g_i and the second derivatives h_{ii} can be efficiently computed via back-propagation. We define the significance of the element f_i as the change of the objective function after setting f_i to zero, i.e., $\delta f_i = 0 - f_i$. According to (7.5), the significance of f_i can then be computed as

$$s_i = -g_i f_i + \frac{1}{2} h_{ii} f_i^2. \tag{7.6}$$

The significance of the k-th feature map are further defined as the summation of significance of all its elements $S_k = \sum_{x,y} s(x, y, k)$, where $s(x, y, k)$ is the significance of the element indexed by location (x, y) on the k-th feature map. All the feature maps are sorted in the descending order by their significance, and the top K feature maps are selected. These selected feature maps have significant impact on the objective function and thus are most relevant to the tracking task. Our feature map selection method can be conducted in an online fashion. In our experiments, we only conduct feature selection at the first frame and have achieved good performance. This should be partially attributed to the robustness of CNN features.

The idea of using quadratic approximation of the cost function to remove connections in networks can be traced back to 1989 [22]. The aim was to reduce the number of parameters and improve speed, while we target on removing noisy feature maps and improving tracking accuracy.

Target Localization Figure 7.5c, d show the design of the CNNs for target localization. After feature map selection in the first frame, we build the SNet and the GNet on top of the selected conv4-3 and conv5-3 feature maps, respectively. Both networks share the same architecture that consists of two additional convolutional layers. The first additional convolutional layer has convolutional kernels of size 9×9 and outputs 36 feature maps as the input to the next layer. The second additional convolutional layer has kernels of size 5×5 and outputs the foreground heat map of the input image. ReLU is chosen as the nonlinearity for these two layers.

SNet and GNet are initialized in the first frame by minimizing the following square loss function:

$$\begin{aligned} L &= L_S + L_G, \\ L_U &= \|\hat{\mathbf{M}}_U - \mathbf{M}\|_F^2 + \beta \|\mathbf{W}_U\|_F^2, \end{aligned} \tag{7.7}$$

where the subscript $U \in \{S, G\}$ indicates SNet and GNet, respectively; $\hat{\mathbf{M}}_U$ represents the foreground heat map predicted by the network; $\mathbf{M}$ is the target heat map, $\mathbf{W}_U$ is the weight parameter of the convolutional layers; β is a trade-off parameter for weight decay.

Note that the sel-CNN for selecting features and the SNet and GNet for localization are different in CNN structures. The sel-CNN architecture is very simple to avoid using noisy feature maps to overfit the objective function, whereas the SNet and GNet are more complex. Since the noisy feature maps have been discarded by the feature map selection, more complex models facilitate more accurate tracking.

In a new frame, we crop a rectangle ROI region centered at the last target location. By forward propagating the ROI region through the networks, the foreground heat maps are predicted by both GNet and SNet. Target localization is first performed on the heat map produced by GNet. Denote the target location as $\hat{\mathbf{X}} = (x, y, \sigma)$, where x, y and σ represent the center coordinates and scale of the target bounding box, respectively. Given the target location $\hat{\mathbf{X}}^{t-1}$ in the last frame, we assume the locations of target candidates in the current frame are subject to a Gaussian distribution

$$p(\mathbf{X}^t|\hat{\mathbf{X}}^{t-1}) = \mathcal{N}(\mathbf{X}^t; \hat{\mathbf{X}}^{t-1}, \boldsymbol{\Sigma}), \tag{7.8}$$

where $\boldsymbol{\Sigma}$ is a diagonal covariance matrix that indicates the variances of the location parameters. The confidence of the i-th candidate is computed as the summation of all the heat map values within the candidate region $conf_i = \sum_{j \in \mathbf{R}_i} \hat{\mathbf{M}}_G(j)$, where $\hat{\mathbf{M}}_G$ denotes the heat map generated by GNet; $\mathbf{R}_i$ is the region of the i-th target candidate according to its location parameter $\mathbf{X}_i^t$ (7.8); j denotes the coordinate index. The candidate with the highest confidence is predicted as the target by GNet.

According to the analysis above, GNet based on the conv5-3 layer captures semantic features and is highly invariant to intra-class variation. Hence, the foreground heat map generated by GNet highlights both the target and background distractors with similar appearances.

To prevent the tracker from drifting to background, we further promote a distractor detection scheme to determine the final target location. Denote the target location predicted by GNet as $\hat{\mathbf{X}}_G$, the corresponding target region in the heat map as $\mathbf{R}_G$. The probability of distractor occurring in background is evaluated by the proportion between the confidence values outside and inside the target region

$$P_d = \frac{\sum_{j \in \hat{\mathbf{M}}_G - \mathbf{R}_G} \hat{\mathbf{M}}_G(j)}{\sum_{k \in \mathbf{R}_G} \hat{\mathbf{M}}_G(k)}, \tag{7.9}$$

where $\hat{\mathbf{M}}_G - \mathbf{R}_G$ represents the background region on heat map $\hat{\mathbf{M}}_G$. When the proportion P_d is less than a threshold (0.2 in all the experiments), we assume no co-occurring distractor and use the target location predicted by GNet as the final result. Otherwise, the same target localization procedure described above is performed on the heat map $\hat{\mathbf{M}}_S$ predicted by SNet to determine the final target location.

Online Update To avoid the background noise introduced by online update, we fix GNet and only update SNet after the initialization in the first frame. SNet is updated following two different rules: the adaptation rule and the discrimination rule, which aim to adapt SNet to target appearance variation and improve the discriminative

power for foreground and background, respectively. According to the adaptation rule, we finetune SNet every 20 frames using the most confident tracking result within the intervening frames. Based on the discrimination rule, when distractors are detected using Eq. (7.9), SNet is further updated using the tracking results in the first frame and the current frame by minimizing

$$\min \beta \|\mathbf{W}_S\|_F^2 + \sum_{x,y} \left\{ \left[\hat{\mathbf{M}}_S^1(x, y) - \mathbf{M}^1(x, y) \right]^2 + \left[1 - \boldsymbol{\Phi}^t(x, y) \right] \left[\hat{\mathbf{M}}_S^t(x, y) - \mathbf{M}^t(x, y) \right]^2 \right\}, \tag{7.10}$$

where $\mathbf{W}_S$ denotes the convolutional weight of SNet; (x, y) are spatial coordinates; $\hat{\mathbf{M}}_S^t$ and $\mathbf{M}^t$ represent the heat map for the t-th frame predicted by SNet and the heat map generated according to the predicted target location (a 2-dimensional Gaussian centered at the target location), respectively; the foreground mask $\boldsymbol{\Phi}^t$ indicates the predicted target bounding box, i.e., $\boldsymbol{\Phi}^t(x, y) = 1$ if the location (x, y) belongs to the target region and $\boldsymbol{\Phi}^t(x, y) = 0$, otherwise.

The second term in Eq. (7.10) corresponds to loss for locating the target in the first frame. When distractors appear or the target undergoes severe occlusion in the current frame, the estimated target region is not reliable for learning target appearance. Therefore, we choose a conservative scheme by adding the first frame to supervise update so that the learned model still captures the appearance in the first frame.

Meanwhile, the third term in Eq. (7.10) removes the loss in the unreliable target region and only considers those within the background region in the current frame. It enforces the model to put more efforts into assigning the co-occurring distractors as background. The combination of the second term and the third term in Eq. (7.10) can help SNet to better separate the target from background and alleviate the model degradation caused by occlusion or distractors.

Implementation Details The implementation details of our FCNT method are as follows. Both the sel-CNN for feature map selection and the GNet and SNet for target localization are trained in the first frame using back-propagation for 50 iterations. Afterwards, SNet are fine-tuned for 3 iterations at each update step. The learning rates are set to $1e^{-9}$ for the sel-CNN and $1e^{-7}$ for the GNet and SNet. The number of feature maps selected by the proposed feature selection method is set to $K = 384$ for both the conv4-3 and conv5-3 layers. The size of the input ROI region centered at target location is 386×386 pixels. The weight decay parameter β in 7.7 and 7.10 is set to 0.005. At each frame, 600 target candidates are randomly sampled. The variance of location parameters in 7.8 are set to $\{10, 10, 0.004\}$ for x, y translation and scale, respectively. All the parameters are fixed through the experiment.

7.3 Visual Tracking Based on Structured Siamese Network

This section proposes a structured siamese (denoted as StructSiam) network, which simultaneously performs discriminative pattern detection, local structure learning and integration in an end-to-end manner. The overall pipeline of our StructSiam method is illustrated in Fig. 7.6.

The two-stream siamese network [4] is trained offline to locate a 127×127 template image z within a larger 255×255 search image x. A similarity function is learned to densely compare the template image z to each candidate region of the same size in the search image x, so as to predict a score map that highlights the target region. Specifically, a cross-correlation layer is proposed to compute the similarity for all translated subregions in x in one pass,

$$F(z, x) = \varphi(z) * \varphi(x) + b, \tag{7.11}$$

where φ is the convolutional feature embedding generated by each network stream; $b \in \mathbb{R}$ denotes the bias; and $F(\mathbf{.}, \mathbf{.})$ represents the predicted confidence score map of size 17×17.

In Fig. 7.6, the two streams of the network share the same architecture and parameters, consisting of three components: local pattern detector, context modeling module, and integration module. The final cross-correlation operation is based on the obtained maps, which we call structure patterns.

The network is offline trained to adopt the logistic loss,

$$L = log(1 + e^{-yv}), \tag{7.12}$$

where v is the real-valued score of a single template-candidate pair and $y \in \{+1, -1\}$ is its ground truth label as in [4].

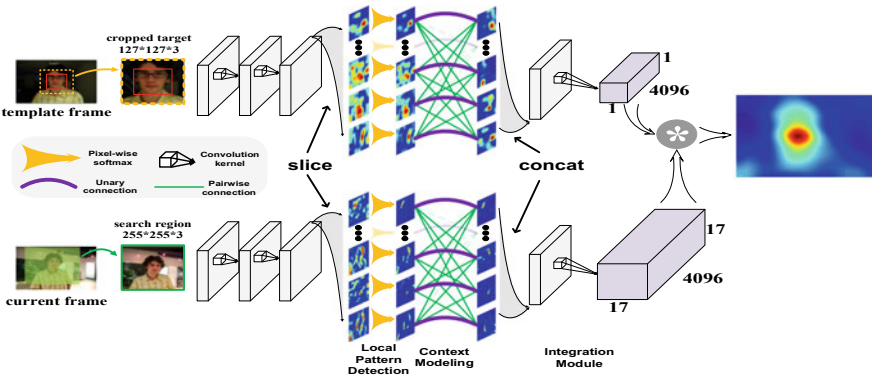


Fig. 7.6 The pipeline of our StructSiam algorithm

Informative Local Pattern Detector Informative local patterns are crucial cues to characterize target appearance. Instead of manually dividing the target region into prefixed parts, we design the local pattern detector to automatically identify discriminative patterns through end-to-end training.

The local pattern detector comprises two convolutional layers with kernel size of 11×11 and 5×5 respectively. Each of these convolutional layers is followed by a batch normalization layer [17], a ReLU layer [25], and a 3×3 max pooling layer. The module takes the image crop as input and detects local patterns of target appearance. The output feature map has 256 channels, each of which corresponds to a specific local pattern. The softmax layer is adopted to normalize the output feature map across channels.

As opposed to features in deeper layers that have low resolution and limited detailed information, the proposed local pattern detector, with only two convolutional layers and two max-pooling layers, has a relatively small receptive field. Therefore, it can better focus on local regions of the target and preserve more detailed information (see the visualized results of the local pattern detector module in Fig. 7.6 as examples). This design is also consistent with recent findings [8, 11] in visual tracking that detailed low-level features are more discriminative and suitable for target matching. However, the local pattern detector has a major drawback. Since individual patterns are independently detected by different channels of the output feature maps, the structure relationships between these local patterns are mostly ignored. As a consequence, the detection results may be inaccurate and vulnerable to background noise. Based on this observation, we introduce the context modeling module for refinement.

Context Modeling Module Generally, our local pattern detector tends to capture local patterns like heads, legs and torsos of persons, wheels of cars or bicycles, and regions with significant edges. They are common in visual tracking tasks, and their appearances can be significantly different for various targets, sequences and time. Thus, embedding prior knowledge on these generic local patterns into the network is benefit for target recognition during tracking. We regard the prior knowledge as the relationships among local patterns. When the tracked object is undergoing cluttered background or drastic appearance change, the detection result of each single local pattern is not reliable. Thus the relationships between different local patterns (i.e., context information) should be considered to facilitate the detection process. The context information incorporation is achieved by message passing, which can enforce the responses of regions that are highly structural, and suppress the noisy background responses. To implement the message passing process efficiently, we introduce conditional random field (CRF) approximation into our network. We formulate the target's local pattern detection problem by using a graph and model the joint probability relationships among local patterns generated by previous stage through CRF.

Let X_i be the random variable associated to pixel i, which represents the type of local pattern assigned to the pixel i and can take any value from a predefined set $\mathscr{P} = \{p_1, p_2, ..., p_c\}$, and c is the channel size of feature maps. We regard each

channel represents a specific local pattern. Let X be the vector formed by the random variables $X_1, X_2, ..., X_N$, where N is the number of pixels in the feature map. Given a graph $G = (V, E)$, where $V = \{X_1, X_2, ..., X_N\}$, and a global observation (image) I, the pair (I, X) can be modeled as a CRF characterized by a Gibbs distribution of the form $P(X = x|I) = \frac{1}{Z(I)} e^{-E(x|I)}$. Here $E(x)$ is called the energy of the configuration $x \in \mathscr{L}^N$ and $Z(I)$ is the partition function. From now on, we drop the conditioning on I in the notation for convenience.

The energy of a label assignment x is given by:

$$E(x) = \Sigma_i \psi_u(x_i) + \Sigma_i \Sigma_{j=max(0,i-R)}^{min(N-1,i+R)} \psi_p(x_i, x_j), \tag{7.13}$$

where the unary energy component $\psi_u(x_i)$ measures the inverse likelihood (and therefore, the cost) of the pixel i is subordinate to the local pattern x_i, and pairwise energy component $\psi_p(x_i, x_j)$ measures the cost of assigning local pattern types x_i, x_j to pixels i, j simultaneously, and R is the scope of the surrounding pixels taken into consideration for pairwise energy computation. In our model, unary energies are obtained from local pattern detector's output, which predict labels for pixels without considering the smoothness and the consistency of the local pattern type assignments. The pairwise energies serve as data-dependent smoothing terms and encourage assigning correlated types to pixels with similar features. Considering the translation invariance property of natural images, we implement the pairwise energy using convolutional operations as follows:

$$\psi_p(x_i, x_j) = \Sigma_{j=max(0,i-R)}^{min(N-1,i+R)} w_{i,j} * x_j + b_i, \tag{7.14}$$

where the kernels $w_{i,j}$ and biases b_i for $i = 1, ..., N$, are shared for all locations in the local patterns' maps and we set $w_{i,j=i} = 0$ to prevent message passing from x_i to itself.

Minimizing the above CRF energy $E(x)$ yields the most probable local structure distribution for the input image. Since the direct minimization is intractable, we adopt the mean-field variational inference to approximate the CRF distribution $P(z)$ with the product of independent marginal distributions, i.e., $Q(z) = \prod_i Q(z_i)$. To this end, we first consider individual steps of the mean-field algorithm summarized in Algorithm 1, and implement them using differentiable operations for end-to-end training. Let $U_i(p)$ denote the negative of the unary energy, i.e., $U_i(p) = -\psi_u(X_i = p)$, where p denotes the local pattern type. In our CRF, the unary energy ψ_u is obtained directly from the output of local pattern detector.

In the first step of Algorithm 1, we initialize $Q_i(p)$ with $Q_i(p) \leftarrow \frac{1}{Z_i} e^{U_i(p)}$, where $Z_i = \Sigma_p e^{U_i(p)}$. Note that this is equivalent to applying a softmax function over the unary potentials U across all the labels at each pixel. The message passing (step 4 in Algorithm 1) is then performed by applying two 3×3 convolutional kernels on Q as described in Eq. (7.14). The receptive field for each output pixel is 5×5, which is $\frac{5}{6}$ of the target object (considering the output size of the target template) and is enough to model target structure. Since there is no activation layer (e.g., ReLU)

between the two convolution layers, they can be used to implement the linear mapping in Eq. (7.14 with less parameters than one 5×5 convolution layer. The kernels are learned to encode the context structural information among local patterns. The output from the message passing stage is subtracted element-wisely from the unary input U. Finally, the normalization step of the iteration can be implemented by another softmax operation with no parameters.

Algorithm 7.1 Mean-field approximation.

1: Initialize Q^0, $Q_i^0(p) = \frac{1}{Z_i}(U_i(p))$ for all i
2: Initialize the iteration times L.
3: **for** $t = 1 : L$ **do**
4: $\quad \tilde{Q}_i(p) = \Sigma_{j=max(0,i-R)}^{j=min(N-1,i+R)} w_{i,j} \times Q_j^{t-1} + b_i$
5: $\quad \hat{Q}_i \leftarrow U_i(p) - \tilde{Q}_i(p)$
6: $\quad Q_i^t \leftarrow \frac{1}{Z_i} e^{\hat{Q}_i(p)}$
7: **end for**

Given the above implementation, one iteration of the mean-field algorithm can be formulated as a stack of common neural network layers. Multiple mean-field iterations can be achieved by recurrently passing the estimated probability Q through the network multiple times. In practice, we find that three iterative steps are enough to obtain a satisfying performance.

Every local pattern receives message from other patterns, which can be seen as contextual information. The contextual information indicates the structure information of the input image inherently. Consequently, the local pattern maps are effectively refined after message passing stage. The final score map is less noisy when the tracking target is undergoing cluttered background and drastic deformation challenges.

In general, by message passing among local patterns, the CRF probability model describes the universal structure information of generic objects despite the category of the target. Since all operations of the context modeling module are differentiable, the whole network can be trained in an end-to-end manner.

Integration Module The output maps of the context modeling module are capable of capturing precise local patterns' information. Since various local patterns correspond to different positions in the target region, directly correlating the template with the search region is vulnerable to deformation. In contrast to SiameseFC [4] using features with spatial layout for correlation, our integration module aggregates the local patterns (of both target and candidates) into a 1×1 feature map, with each channel acting as an attribute to indicate the existence of certain pattern regardless of its location. In our method, feature maps corresponding to the template and search region are fed into the 6×6 convolution layer, which leads to a $1 \times 1 \times 4096$ tensor representing the template and a $17 \times 17 \times 4096$ tensor $\mathbf{T}$ representing the search region. These two tensors are correlated to obtain the final response. Clearly, each spatial position (x, y) in the search region has a corresponding $1 \times 1 \times 4096$ tensor $\mathbf{T}(x, y, :)$ which is different from others. Each pixel in the final maps indicates the

local pattern information for a region, and the final correlation finds the pixel (as the center of the target) that incorporates the same local patterns as the targets. The feature variation caused by drastic deformation changes can be reduced to some extent in this way.

Implementation Details The implementation details of our StructSiam tracker are as follows.

Since the motivation of our network is different from SiameseFC, which aims to learn a matching function to perform metric learning, training only on ILSVRC2014 VID dataset is not suitable. Visual object tracking task is to track generic objects no matter what categories. The ILSVRC2014 VID dataset is more biased to animals contains head, limbs, and torso that will lead to ineffective learning of our structured network. And rotation objects are necessary for structure patterns learning, which aim to respond to the center of structural regions. To model generic object inner structure, we use ILSVRC2014 VID dataset [30] and ALOV dataset [33]. We discard the common sequences appear in the testing datasets for a fair comparison.

We implement the proposed method in python with the Tensorflow library [1]. The parameters of the whole network are initialized randomly guided by a Gaussian distribution [15]. It is well known that softmax leads to vanishing of gradients which makes the network training inefficient. Thus, we multiply the feature maps with a constant β, which will make the network converge much faster. β is set to equal to the channel size of feature maps, i.e., $\beta = 256$. Training is performed over 50 epochs, each consisting of 60,000 sampled pairs. The gradients for each iteration are estimated using mini-batches of size 8, and the learning rate is annealed geometrically at each epoch from 10^{-2} to 10^{-5} as in [4]. We use the SGD method to conduct optimization and train the network.

With the learned StructSiam, we summarize our tracking algorithm as follows: given the target location at I_1, i.e., a bounding box $b_1 \in \mathbb{R}$, we crop the corresponding region to serve as the target template O_1 that is slightly larger than b_1 and centered at b_1. We then extract the deep features of O_1 to get F_1. When tracking at the tth frame, we crop search regions on three scales, i.e., $1.025^{\{-1,0,1\}} \times S_0$, where S_0 is the original scale, centering at b_{t-1}. Then, we get 3 response maps via (7.11). We search the maximum value among the three response maps and get its respective location and scale, which leads to b_t.

7.4 Visual Tracking Based on Deep Reinforcement Learning

Deep-learning-based tracking algorithms have significantly improved the tracking performance in recent years [4, 8, 24, 27]. The pretrained convolutional neural networks (e.g., AlexNet, VGG-16 and VGG-M) are usually adopted to obtain rich feature representation for robust tracking. Among these methods, the MDNet tracker [27] achieves top-ranked performance in popular benchmarks (such as OTB-100 [38]

Fig. 7.7 The search strategies of of different trackers. **a** MDNet [27]: search with random sampling; **b** ADNet [40]: iterative search with a series of discrete actions; and **c** our tracker: fast search with only one continuous action

and VOT2015 [20]). This method embeds the offline trained VGG-M network into the particle filter framework, where 256 candidate samples are randomly generated and each sample is verified using a CNN-based observation model in each frame. However, it is very slow due to the random sampling search strategy. To address this issue, Yun *et al.* [40] propose a reinforcement-learning-based tracker with an action-decision network (ADNet). This method takes a series of discrete actions to iteratively search the tracked object in each frame. Experimental results show that the ADNet tracker achieves slightly worse but three times faster performance than the MDNet method. The search strategies of MDNet and ADNet methods are illustrated in Fig. 7.7a, b, respectively. We note that the learned iterative strategy in [40] is also far from the real-time requirement since it requires many iterative steps in every frame.

In this section, we attempt to conduct tracking within a novel "Actor-Critic" framework, the overall pipeline of which is illustrated in Fig. 7.8. The "Actor" model aims to give one continuous action to directly make the tracker move the bounding box to the object location in the current frame. It can be effectively offline trained with a "Critic" network based on deep reinforcement learning. During the tracking process, the "Critic" model combines the action produced by "Actor" to determine the quality of the action and facilitates improving the tracking performance. The details of our tracking framework are presented as follows.

Considering tracking as a sequential decision problem, our algorithm follows Markov Decision Process (MDP). The basic components of MDP include states $s \in S$, actions $a \in A$, the state transition function $s^{'} = f(s, a)$, and the reward $r(s, a)$. In our MDP framework, the tracker is treated as an agent to infer the accurate bounding box of the tracked object in each frame. This agent is interacted with an environment through a sequence of observations $s_1, s_2, ..., s_t$, actions $a_1, a_2, ..., a_t$ and rewards

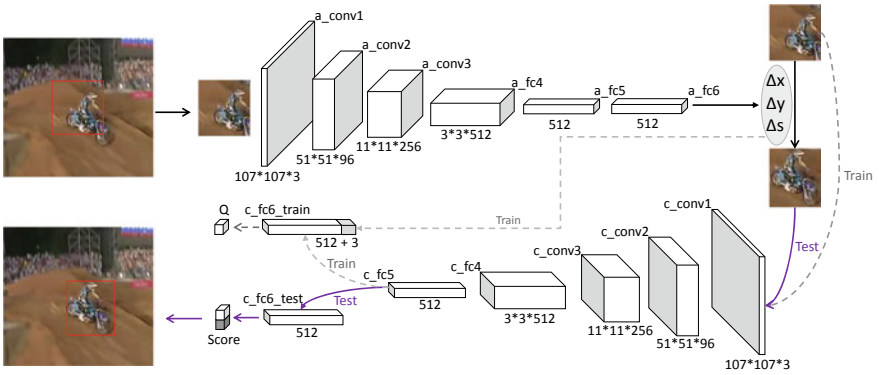


Fig. 7.8 The pipeline of the proposed tracking algorithm. "a" means "Actor" and "c" means "Critic"

$r_1, r_2, ..., r_t$. In frame t, the agent gives the continuous action a_t according to the current state s_t and obtains the tracking result as $s_t^{'}$. Here, the action a_t is defined as the relative motion of the tracked object indicating how its bounding box should move directly in frame t. Different with ADNet [40], our tracker takes only one continuous action to locate the tracked object, which makes our tracker more efficient. The detailed settings of s, a, $f(s, a)$ and r are presented as follows (we drop the frame index t for clarity in this subsection).

State. Here, we define the state s as the observation image patch within the bounding box $b = [x, y, h, w]$, where (x, y) is the center position and h and w stand for the height and width respectively. To be specific, we define a pre-processing function $s = \phi(b, F)$ to crop the image patch within the bounding box b in a given frame F and resize it to fit the input size of the deep network.

Action and State Transition. To conduct continuous control, the action space is assumed to be continuous, indicating how the bounding box should move directly. Here, we use the action $a = [\Delta x, \Delta y, \Delta s]$ to depict the relative motion of the tracked object, where Δx and Δy denote the relative horizontal and vertical translations and Δs stands for the relative scale change. Considering the temporal continuity, we introduce some constraints to restrict the range of the action a: $-1 \leq \Delta x \leq 1$, $-1 \leq \Delta y \leq 1$ and $-0.05 \leq \Delta s \leq 0.05$. By applying the action a to the original bounding box b, the new bounding box $b^{'} = \left[x^{'}, y^{'}, h^{'}, w^{'}\right]$ can be obtained as

$$\begin{cases} x^{'} = x + \Delta x \times h \\ y^{'} = y + \Delta y \times w \\ h^{'} = h + \Delta s \times h \\ w^{'} = w + \Delta s \times w \end{cases}. \tag{7.15}$$

Then, the state transition process $s^{'} = f\,(s, a)$ can be implicitly achieved by applying the pre-processing function $\phi\left(b^{'}, F\right)$.

We attempt to directly infer the optimal action a based on the state s using an "Actor" model, i.e., $a = \mu\,(s|\theta^{\mu})$. $\mu\,(.)$ denotes the deep network for our "Actor" model with the parameter θ^{μ}. The "Actor" model is trained offline with the proposed training strategy in Sect. 7.4, and then applied for online tracking. In practice, we exploit a double bounding box scheme to build the "Actor" model (i.e., two bounding boxes of the target size and twice target size with the same center location).

Reward. The reward function $r(s, a)$ describes the localization accuracy improvement when transferring state s into state $s^{'}$ with a given action a. Thus, it can be defined based on the overlap ratio of the new bounding box $b^{'}$ and the ground truth G,

$$r(s, a) = \begin{cases} 1 & \text{if IoU}(b^{'}, G) > 0.7 \\ -1 & \text{else} \end{cases}, \tag{7.16}$$

where IoU denotes the Intersection-over-Union criterion (i.e., $\text{IoU}(u, v) = u \cap v / u \cap v$ for bounding boxes u and v). With every action, the reward will be generated and then be used to update the deep networks in offline learning.

Offline Training Inspired by the recent successful trackers with lightweight deep networks, we use the pretrained VGG-M model to initialize our "Actor" and "Critic" networks. As illustrated in Fig. 7.8, there are three convolutional layers in both two networks, which are consistent with the first three convolutional layers of the VGG-M network. For the "Actor" network, the next two fully connected layers with 512 output nodes are combined with the ReLU operation, and the last fully connected layer generates a three-dimensional output. For offline training, the "Critic" model has similar structures with "Actor" expect the last fully connected layer as it requires concatenating the three-dimensional action vector to obtain a Q-value for action evaluation according to the current state.

Here, we train our "Actor-Critic" network using the DDPG approach [23], the core idea of which is to iteratively update the "Critic" and "Actor" models with training sample pairs collected based on the RL rule. Given N pairs of $(s_i, a_i, r_i, s_i^{'})$, the "Critic" model $Q(s, a)$ can be learned using the Bellman equation as in Q-learning. With the utilization of the target networks $\mu^{'}$ and $Q^{'}$, the learning process can be achieved by minimizing the following loss,

$$L = \frac{1}{N}\sum_{i}(y_i - Q(s_i, a_i|\theta^{Q}))^2, \tag{7.17}$$

where $y_i = r_i + \gamma\, Q^{'}(s_i^{'}, \mu^{'}(s_i^{'}|\theta^{\mu^{'}})|\theta^{Q^{'}})$.

Then, the "Actor" model can be updated by applying the chain rule to the expected return from the start distribution J with respect to the model parameters:

Algorithm 7.2 Offline training the "Actor" network

Input: Training sequences [F] and their corresponding ground truths [G]
Output: Trained weights for the "Actor" network

Initialize "Critic" $Q(s, a)$ and "Actor" $\mu(s|\theta^{\mu})$ with weights θ^{Q} and θ^{μ}.
Initialize the target network $Q^{'}$ and $\mu^{'}$ with weights $\theta^{Q^{'}} \leftarrow \theta^{Q}, \theta^{\mu^{'}} \leftarrow \theta^{\mu}$
Initialize the replay buffer R
repeat
Randomly select a piece of frames $\left[F_k, F_{k+1}, \ldots, F_{k+T}\right]$ with their ground truth $\left[G_k, G_{k+1}, \ldots, G_{k+T}\right]$
Receive initial observation state s_k according to F_k and G_k
Train the 'Actor' network for an iteration utilizing s_1
for each $t = 2, T + 1$ **do**
1. Obtain state s_t according to state s_{t-1} and F_{k-1+t}
2. Select action $a_t = \mu(s_t|\theta^{\mu})$ according to the current policy and exploration probability ϵ;
3. Execute action a_t according to the transition in equation (7.15), observe reward r_t as equation (7.16) and the next state $s_t^{'}$
4. Store transition $(s_t, a_t, r_t, s_t^{'})$ in R
end for
Sample a random mini-batch of N transitions $(s_i, a_i, r_i, s_i^{'})$ from R
Update 'Critic' $Q(s, a)$ by minimizing the loss following equation (7.17)
Update 'Actor' $\mu(s|\theta^{\mu})$ using the sampled policy gradient following equation (7.18)
Update the target networks:

$$\begin{aligned} \theta^{Q^{'}} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q^{'}} \\ \theta^{\mu^{'}} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu^{'}} \end{aligned} \tag{7.19}$$

until Reward become stable

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_{i} \nabla_a Q(s, a|\theta^{Q})\big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_i}. \tag{7.18}$$

During the training iteration, we randomly select a piece of training sequences $\left[F_k, F_{k+1}, \ldots, F_{k+T}\right]$ with their ground truth $\left[G_k, G_{k+1}, \ldots, G_{k+T}\right]$ from the training data (k is the start frame number and T is the frame length). After that, we apply our tracker in the selected sequence to obtain a training pair $(s_t, a_t, r_t, s_t^{'})$ at frame t.

It is not feasible to directly apply the original DDPG framework to train our model, since the action space is very huge in our tracking problem. Thus, we attempt to improve the training process from the two following aspects.

(1) Due to the huge action space, it is difficult to obtain a positive reward when an agent follows a random exploration strategy for a given video clip. This will make the DDPG method less effective in training our model. To solve this problem, we utilize the supervised information from the first frame to initialize the "Actor" model for adapting the current environment. That is, the "Actor" model is fine-tuned through the adaptive moment estimation method to minimize the following L_2 loss function,

$$\min \frac{1}{M} \sum_{m=1}^{M} \left[\mu\left(s_m|\theta^{\mu}\right) - a_m\right]^2, \tag{7.20}$$

where M is the number of training samples and $\mu\left(.|\theta^{\mu}\right)$ denotes the "Actor" network with parameter θ^{μ}. s_m is the m-th sampled state and a_m denotes its ground truth action.

(2) The initialization scheme above cannot fully solve the imbalance problem of positive and negative samples as there are many unpredicted challenges causing tracking drifts. Thus, we exploit expert decisions to guide the learning process. The original DDPG algorithm introduces an exploration policy $\mu^{'}$ by adding noise sampled from a noise process $\mathcal{N}$ to the actor policy $\mu(s_t) = \mu(s_t|\theta_t^{\mu}) + \mathcal{N}$. However, this similar exploration mechanism is not suitable for our tracking task due to the huge action space. Therefore, we adopt a probabilistic expert decision guidance to supersede exploration mechanism in reinforcement learning. In a video sequence, with a certain probability ϵ, an expert decision guidance is applied to replace the action output by the "Actor" network. The probability ϵ gradually decreases during the training process.
Our "Actor" network can be effectively trained offline by the DDPG method with two improvements above. The overall training process is summarized in Algorithm 2.

Online Tracking To make our tracker further suitable for the current sequence, we initialize both "Actor" and "Critic" models with the ground truth in the first frame.

For "Actor", we first sample M candidate bounding boxes $b_m|_{m=1}^{M}$ around the ground truth and calculate their corresponding accurate actions $a_m|_{m=1}^{M}$. Then, we extract the image observation s_m for the candidate location b_m using pre-processing function $s_m = \phi\left(b_m, F\right)$. Thus, the "Actor" network can be fine-tuned using the Adam method to minimize the L2 loss $\frac{1}{M} \sum_{m=1}^{M} \left[\mu\left(s_m|\theta^{\mu}\right) - a_m\right]^2$.

For online tracking, the "Critic" model $v\left(s|\theta^{v}\right)$ is a classification network. To initialize it, we assign a binary label l_m to the m-th candidate using the following rule,

$$l_m = \begin{cases} 1 & \text{if IoU}(b_m, G) > 0.7 \\ 0 & \text{else} \end{cases}, \tag{7.21}$$

where G denotes the ground truth bounding box. With the collected image-label pairs $\{s_m, l_m\}|_{m=1}^{M}$, the "Critic" network is initialized using the Adam method to minimize the following loss function,

$$\arg\min_{\theta^{v}} - \sum\nolimits_{s \in S_+} p_+(s|v;\theta^{v}) - \sum\nolimits_{s \in S_-} p_-(s|v;\theta^{v}), \tag{7.22}$$

where S_+ and S_- denote positive and negative training sets respectively. The "Critic" network outputs the foreground and background probabilities $p_+(s|v;\theta^{v})$ and $p_-(s|v;\theta^{v})$ for a given state (or observation patch) s.

For online tracking, we exploit both "Actor" and "Critic" networks within a tracking-and-verification scheme. In the t-th frame, we first calculate the state s_t using the preprocessing function $\phi\left(b_{t-1}^{'}, F_t\right)$ ($b_{t-1}^{'}$ denotes the optimal object location in the $t-1$ frame and F is the image frame). Second, we put the state s_t into the the "Actor" network resulting in the action a_t, i.e.,$a_t = \mu\left(s_t|\theta^{\mu}\right)$. With the action a_t and location $b_{t-1}^{'}$, we can obtain the new location $b_t^{'}$ and its corresponding state $s_t^{'}$ in the current frame. Then, we utilize the "Critic" network to verify the observation $s_t^{'}$, i.e., $v\left(s_t^{'}|\theta^{v}\right)$. If the score given by the "Critic" network is larger than 0, we treat the action a_t being reliable and adopt the location $b_t^{'}$ as the optimal location in the t-th frame. Otherwise, we exploit a re-detection technique using the "Critic" network to evaluate a series of sampled candidates $b_t^m\big|_{m=1}^{M}$ around $b_{t-1}^{'}$ (same as the sampling strategy in Network Initialization). After that, the optimal location $b_t^{'}$ is obtained as the candidate with the highest score outputed by "Critic".

An effective updating strategy helps our tracker take a good trade-off between robustness and efficiency. The "Actor" model has a stable performance during the tracking process due to our offline training. Thus, we merely update the "Critic" network when needed. If the verification score given by "Critic" is less than 0, we think it does not fit well with the appearance change in the current environment and use positive and negative samples collected in previous 10 frames to update the network based on Eq. (7.22).

Implementation Details To train the networks in both offline and online tracking stages, we sample $X_t^i = \left(x_t^i, y_t^i, z_t^i\right), i = 1, \ldots, N$ (x and y are horizontal and vertical translations; z denotes the scale) from a Gaussian distribution centered by the object location in frame $t-1$. The covariance is a diagonal matrix $diag\left(0.09d^2, 0.09d^2, 0.25\right)$, where d is the mean of the width and height of the tracked object.

To train our "Actor" network offline, we use 768 video sequences from the ImageNet Video [30] trainval set. We randomly choose continuous twenty to forty frames in a video for each iteration. For initializing the "Actor" network in the first frame, we collect 32 samples whose IoU scores with ground truth are larger than 0.7. The learning rate is set to 1e-4 in the initialization stage.

The possibility of adopting the expert decision ϵ is set to 0.5 at first and reduced by 5% after every ten thousand iterations. We update the target networks every ten thousand iterations. τ in the target networks updating is set to 0.001. The learning rates of the "Actor" and "Critic" networks are set to 1e-6 and 1e-5, respectively. In addition, we use a replay buffer size of 10^4. We finish the training of the "Actor" network after two hundred and fifty thousand iterations.

For online tracking, we collect 500 positive samples and 5000 negative samples with ground truth in the first frame. Only positive samples are used for training the "Actor" network. We initialize the "Actor" network with learning rates 1e-4 until the loss is less than 1e-4 and initialize the "Critic" network with learning rates 1e-4 for 30 iterations. The batch sizes for the "Actor" and "Critic" models are 64 and 128, respectively. When the predicted target location of the highest foreground score of all candidates are less than 0, we consider it as the tracking failure, and the re-detection

is conducted to capture the missed target. We draw 256 samples for the re-detection scheme. Simultaneously, the "Critic" model is online updated with collected samples from 10 recent successful tracking frames. We collect 50 positive samples and 150 negative samples from each successful tracking frame.

7.5 Experimental Results and Discussions

We adopt both success and precision plots to evaluate 10 trackers on OTB-2015. From Fig. 7.9, we have two fundamental observations:

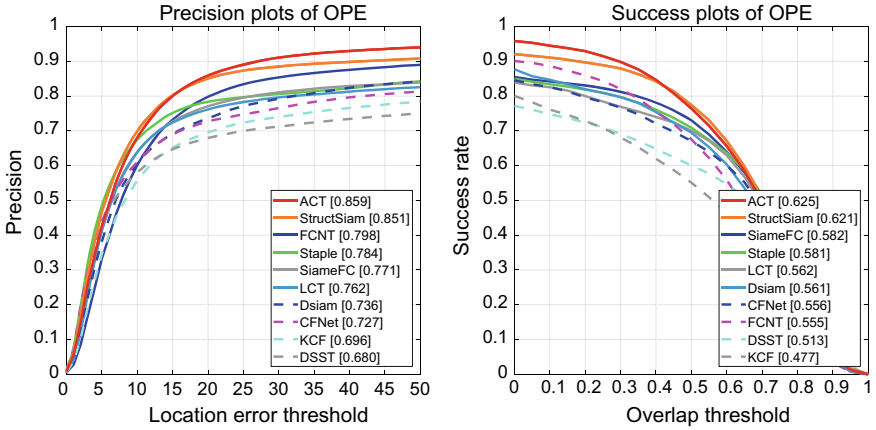


Fig. 7.9 Success and precision plots of different trackers on the OTB-2015 dataset

Table 7.2 The success comparison of mentioned trackers

Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
ACT	***0.638***	***0.578***	***0.602***	***0.614***	***0.629***	***0.633***	**0.619**	***0.585***	***0.602***	**0.536**	***0.605***
Struct Siam	**0.588**	**0.571**	**0.613**	**0.601**	**0.616**	**0.604**	***0.642***	**0.602**	**0.594**	***0.555***	***0.605***
FCNT	0.529	0.529	0.526	*0.558*	0.543	0.442	0.555	0.515	0.549	0.470	0.506
STAPLE	*0.574*	*0.554*	0.535	0.552	*0.598*	0.411	0.558	*0.548*	0.534	0.481	0.529
SiamFC	0.523	0.506	*0.569*	0.557	0.568	*0.573*	*0.568*	0.543	*0.558*	0.506	0.557
LCT	0.550	0.499	0.522	0.557	0.566	0.330	0.532	0.507	0.538	0.452	0.492
CFNet	0.559	0.470	0.514	0.526	0.554	0.533	0.484	0.504	0.530	*0.510*	*0.562*
Dsiam	0.542	0.502	0.513	0.556	0.545	0.520	0.472	0.523	0.544	0.425	0.547
KCF	0.498	0.436	0.448	0.469	0.479	0.307	0.456	0.443	0.453	0.393	0.399
DSST	0.523	0.420	0.453	0.502	0.558	0.379	0.488	0.453	0.470	0.386	0.475

Table 7.3 The precision comparison of mentioned trackers

Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
ACT	***0.881***	***0.826***	**0.788**	***0.867***	***0.860***	**0.887**	**0.804**	**0.799**	**0.837**	**0.707**	***0.843***
Struct Siam	**0.811**	**0.805**	***0.808***	**0.848**	**0.828**	***0.922***	***0.840***	***0.827***	***0.846***	***0.772***	**0.839**
FCNT	0.750	*0.760*	*0.708*	*0.830*	0.777	0.755	*0.726*	*0.73*	*0.800*	0.629	*0.763*
STAPLE	*0.766*	0.748	0.696	0.770	*0.791*	0.609	*0.726*	0.726	0.738	0.661	0.731
SiamFC	0.690	0.690	0.741	0.742	0.736	*0.815*	0.724	0.722	0.756	0.669	0.739
LCT	0.734	0.689	0.667	0.782	0.746	0.490	0.673	0.682	0.746	0.592	0.686
CFNet	0.730	0.639	0.637	0.705	0.701	0.771	0.573	0.656	0.714	*0.670*	0.714
Dsiam	0.728	0.681	0.644	0.739	0.715	0.723	0.554	0.691	0.737	0.550	0.727
KCF	0.713	0.436	0.448	0.469	0.479	0.307	0.456	0.443	0.453	0.393	0.399
DSST	0.704	0.542	0.562	0.691	0.721	0.550	0.595	0.597	0.644	0.481	0.643

(1) compared with traditional trackers (e.g., KCF [16], DSST [9]), the performance of trackers (e.g., FCNT [36], SiameFC [4]) based on deep learning is much better in both success and precision plots.
(2) we compare our algorithms (FCNT, ACT, StructSiam) with seven state-of-the-art trackers. Our algorithms have better performance than other trackers in both evaluations. In details, ACT outperforms all compared algorithms in both success and precision plots, and the performance of StructSiam is only a little worse than ACT.

Tables 7.2 and 7.3 compares the performance of above ten trackers under eleven video attributes on OTB-2015 dataset. ACT and StructSiam achieve better performance in all challenge attributes than other trackers. In addition, StructSiam handles well in several challenges such as motion blur, occlusion and out of view, while ACT is more suitable for dealing with challenges of background clutter and low resolution.

7.6 Summary

Convolutional layers at different levels have different properties, thus jointly consider these properties can capture both the semantic information of the target and discriminate the target from background distractors. A principled feature map selection method is necessary to select discriminative features and discard noisy or unrelated ones. Besides, an effective network structure or framework can greatly improve the tracking performance.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: a system for large-scale machine learning. In: USENIX Symposium on Operating Systems Design and Implementation, pp. 265–283 (2016)
2. Babenko, B., Yang, M., Belongie, S.J.: Robust object tracking with online multiple instance learning. IEEE Trans. Pattern Anal. Mach. Intell. **33**(8), 1619–1632 (2011)
3. Bao, C., Wu, Y., Ling, H., Ji, H.: Real time robust l1 tracker using accelerated proximal gradient approach. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1830–1837 (2012)
4. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: European Conference on Computer Vision, pp. 850–865 (2016)
5. Chen, B., Wang, D., Li, P., Wang, S., Lu, H.: Real-time 'actor-critic' tracking. In: ECCV, pp. 328–345 (2018)
6. Chen, K., Tao, W.: Once for all: a two-flow convolutional neural network for visual tracking. IEEE Trans. Circuits Syst. Video Technol. **28**(12), 3377–3386 (2018)
7. Cui, Z., Xiao, S., Feng, J., Yan, S.: Recurrently target-attending tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1449–1458 (2016)
8. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M., et al.: Eco: Efficient convolution operators for tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, p. 3 (2017)
9. Danelljan, M., Häger, G., Khan, F.S., Felsberg, M.: Discriminative scale space tracking. IEEE Trans. Pattern Anal. Mach. Intell. **39**(8), 1561–1575 (2017)
10. Danelljan, M., Hager, G., Shahbaz Khan, F., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: IEEE International Conference on Computer Vision, pp. 4310–4318 (2015)
11. Danelljan, M., Robinson, A., Khan, F.S., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: European Conference on Computer Vision, pp. 472–488. Springer (2016)
12. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009)
13. Fan, H., Ling, H.: Sanet: Structure-aware network for visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2217–2224 (2017)
14. Hare, S., Saffari, A., Torr, P.H.S.: Struck: Structured output tracking with kernels. In: IEEE International Conference on Computer Vision, pp. 263–270 (2011)
15. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
16. Henriques, J.F., Rui, C., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Trans. Pattern Anal. Mach. Intell. **37**(3), 583–596 (2015)
17. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
18. Jia, X., Lu, H., Yang, M.H.: Visual tracking via adaptive structural local sparse appearance model. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1822–1829 (2012)
19. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1409–1422 (2012)
20. Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Cehovin, L., Fernandez, G., Vojir, T., Hager, G., Nebehay, G., Pflugfelder, R.: The visual object tracking VOT2015 challenge results. In: IEEE International Conference on Computer Vision, pp. 1–23 (2015)
21. Lecun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
22. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Advances in Neural Information Processing Systems, pp. 598–605 (1990)

23. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
24. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: IEEE International Conference on Computer Vision, pp. 3074–3082 (2015)
25. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: International Conference on Machine Learning, pp. 807–814 (2010)
26. Nam, H., Baek, M., Han, B.: Modeling and propagating cnns in a tree structure for visual tracking. arXiv preprint arXiv:1608.07242 (2016)
27. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 4293–4302 (2016)
28. Ning, G., Zhang, Z., Huang, C., Ren, X., Wang, H., Cai, C., He, Z.: Spatially supervised recurrent convolutional neural networks for visual object tracking. In: IEEE International Symposium on Circuits and Systems, pp. 1–4 (2017)
29. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. Int. J. Comput. Vis. **77**(1–3), 125–141 (2008)
30. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
31. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
32. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
33. Smeulders, A.W., Chu, D.M., Cucchiara, R., Calderara, S., Dehghan, A., Shah, M.: Visual tracking: an experimental survey. IEEE Trans. Pattern Anal. Mach. Intell. **36**(7), 1442–1468 (2014)
34. Sun, C., Wang, D., Lu, H., Yang, M.H.: Learning spatial-aware regressions for visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 8962–8970 (2018)
35. Tao, R., Gavves, E., Smeulders, A.W.: Siamese instance search for tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1420–1429 (2016)
36. Wang, L., Ouyang, W., Wang, X., Lu, H.: Visual tracking with fully convolutional networks. In: IEEE International Conference on Computer Vision, pp. 3119–3127 (2015)
37. Wang, L., Ouyang, W., Wang, X., Lu, H.: Sequentially training convolutional networks for visual tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1373–1381 (2016)
38. Wu, Y., Lim, J., Yang, M.: Object tracking benchmark. IEEE Trans. Pattern Anal. Mach. Intell. **37**(9), 1834–1848 (2015)
39. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: a benchmark. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2411–2418 (2013)
40. Yoo, S., Yun, K., Choi, J.Y., Yun, K., Choi, J.: Action-decision networks for visual tracking with deep reinforcement learning. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
41. Zhang, Y., Wang, L., Qi, J., Wang, D., Feng, M., Lu, H.: Structured siamese network for real-time visual tracking. In: ECCV, pp. 355–370 (2018)
42. Zhong, W., Lu, H., Yang, M.: Robust object tracking via sparse collaborative appearance model. IEEE Trans. Image Process. **23**(5), 2356–2368 (2014)
43. Zhu, G., Porikli, F., Li, H.: Robust visual tracking with deep convolutional neural network based object proposals on pets. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1265–1272 (2016)

Chapter 8
Conclusions and Future Work

This chapter summarizes the work in this book and discusses potential research directions of future research for online visual tracking.

8.1 Summary

This book is a guide to designing online visual tracking algorithms. The goal is to introduce and review some tracking algorithms, and discuss their advantages and drawbacks based on the benchmark evaluation.

The book begins with the introduction of visual tracking, including the basic components, challenges, and datasets. In the second part of this work (Chaps. 2–5), we focus on visual tracking algorithms based on shallow machine learning models. Although the performance of these methods have not achieved satisfactory results in the benchmark dataset, some experiences will be useful in designing the tracking system. In the third part of this book (Chaps. 6–7), we focus on recent algorithms based on correlation filters and deep learning. The experiments show that the correlation filters with deep features and the end-to-end deep networks could achieve state-of-the-art performance.

8.2 Future Work

Visual tracking is still a hot problem in computer vision, and there are still quite a lot of future attempts to be investigated.

There exists a few standard training and test protocol in the tracking filed. Up to now, the tracking algorithms usually set the model hyper-parameters based on the test set, which makes the readers difficult to know the generalization ability of

H. Lu and D. Wang, *Online Visual Tracking*,
https://doi.org/10.1007/978-981-13-0469-9_8

different tracking algorithms. Thus, the construction of the standard training and test process will facilitate the future studies of tracking algorithms. A standard training and test protocol requires large-scale video sequences, and also includes fixed training, validation, test splits.

Most existing trackers adopt pretrained networks (e.g., VGG-16, AlexNet) as their basic models. These networks are designed and trained for image classification tasks, which are not very suitable for the tracking task. Although the structure and function of networks have developed a lot, the networks in visual tracking are still the basic ones. There is need to design more suitable networks for visual tracking. The ideal network should be small to make online tracking faster. How to design a light and powerful tracker with both high accuracy and speed is a meaningful direction.

For long-term tracking, the target may disappear for many times and experience huge appearance variations. If the tracker cannot find the target after tracking failures, it would be useless in the following frames. Similarly, it is also important for a tracker to find the target after tracking drift in real-world application. The key point to refind the target is to judge the disappearance accurately and relocate the target in the whole image. The introduction of re-detection mechanism is essential for long-term tracking and will be an interesting direction.

Visual tracking is a video sequence problem and reinforcement learning is a good solution for the sequential decision-making problem which is famous for its long-term benefits. However, there are few successful applications of reinforcement learning in short-term visual tracking, let alone long-term tracking. How to set the problem properly need to be solved. Besides, the hard training process of deep networks based on reinforcement learning should be noted. How to apply reinforcement learning on trackers to make them more intelligent is a good question.

There is a long way to apply existing tracking methods to real-world applications, because the trackers with higher precision scores usually run slowly and those with higher speed cannot localize the target accurately. To facilitate the real-world application of tracking methods, the designed tracker needs to meet some requirements. It should run at more than 30fps and have a good accuracy during the long-term tracking task. Since there are many unpredictable conditions in the real-world application, robust and fast long-term algorithms should be proposed.

Finally, more recent machine learning models will be attempted to apply on the tracking problem. The essence of visual tracking is an online machine learning problem. Any breakthrough in machine learning may facilitate the studies of the tracking problem. It is a good choice to exploit new machine learning models for visual tracking, such as attention mechanism, meta-learning, and capsules network.