Tracking and detecting objects in image sequence

Wang, Li

2016

Wang, L. (2016). Tracking and detecting objects in image sequence. Doctoral thesis,
Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/65878>

<https://doi.org/10.32657/10356/65878>

Tracking and Detecting Objects in Image Sequence

Li WANG

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2016

To *my parents*

Acknowledgments

First of all, I would like to express my sincerest gratitude to my supervisor Assistant Professor Gang WANG who has supported and helped my PhD study a lot. His guidances are definitely beneficial for my research works. I have been impressed all the time by his rich academic knowledge and high working efficiency. I could not imagine finishing my PhD study without his support and help.

I am also grateful to Associate Professor Kap Luk CHAN for his helpful advice on my research and useful comments on this thesis. I would also like to express my sincere appreciations to Dr. Nam Trung PHAM and Dr. Tian-Tsong NG for their useful suggestions on the research work carried out in Institute for Infocomm Research (I²R). My sincere thank also goes to Assistant Professor Qingxiong YANG for his useful comments on the co-authored journal paper.

I would like to thank my colleagues: Dr. Wei HUANG, Dr. Junyan WANG, Dr. Xiaoyan ZHANG, Dr. Lining ZHANG, Dr. Jinghua WANG, Mr. Bing WANG and Mr. Ting LIU, for the experience of studying and working with them. Also, I am grateful to my friends: Dr. Hao XIA, Dr. Haijian ZHANG, Dr. Pengcheng WU, Dr. Peilin ZHAO and Dr. Sixuan ZHAO, for their help during these years.

Last but not least, I would like to thank my parents for their unreserved support throughout my life.

Abstract

Tracking and detecting arbitrary objects are important in many applications such as video surveillance, video analytics and human-machine interactions. Although many promising methods have been proposed in this area, it is still very challenging to track and detect arbitrary objects due to issues such as complicated motion transformations and occlusions. In this thesis, four pieces of works are developed to address these problems in tracking and detecting objects.

The first piece of work addresses learning hierarchical features for visual object tracking by using deep learning. Previously, raw pixel values or hand-crafted features are used to represent target objects. However, these representations are not able to handle large appearance variations of arbitrary target objects. Recently, deep learning has achieved very promising results in speech recognition and image classification. Nevertheless, it is non-trivial to apply deep learning to visual object tracking. Usually, deep neural networks require a lot of training data to learn a large number of network parameters. However, training data is not sufficient for visual object tracking as annotations of a target object are only available in the first frame of a test sequence. To solve this problem, a feature learning algorithm is proposed for visual object tracking by using domain adaptation. First, hierarchical features are learned from auxiliary video sequences by using a two-layer neural network. Embedding the temporal slowness constraint into the stacked network architecture makes learned features robust to complicated motion transformations, and this manipulation is important for visual object tracking. Then, given a target image sequence, a domain adaptation module is proposed to adapt the pre-learned features according to the specific target object. The adaptation is conducted in both two layers of the neural network to include appearance information of the specific target object. As a result, learned hierarchical features can

be robust to both complicated motion transformations and appearance changes of target objects. Experimental results demonstrate that significant improvement can be achieved by using the learned hierarchical features, especially on image sequences with complicated motion transformations.

As an extension of the first work, the second piece of work focuses on learning hierarchical features for multiple object tracking. First, generic features can be pre-learned from auxiliary data. Then, it is straightforward to adapt the pre-learned features according to multiple target objects independently. In this way, the adaptation module for each target object will only use its own annotation data, which is actually still limited for feature adaptation. To solve this problem, a joint learning strategy using the annotation data of all target objects is proposed to conduct feature adaptation jointly according to multiple target objects. As a result, the feature adaptation module can achieve better performance since it makes use of more training data. The proposed joint learning strategy can simultaneously learn common features shared by all target objects and individual features for each object. Experimental results demonstrate that using learned hierarchical features can significantly improve multiple object tracking performance.

The third piece of work is about learning hierarchical features for visual object tracking by using another neural network different from the one used in the first and second works. As mentioned before, training data is not enough in visual object tracking. Consequently, the deep neural networks used in the first and second works pre-learn the network parameters from auxiliary data. However, this kind of pre-training is inconvenient for visual object tracking. In this work, a feature learning algorithm is proposed for visual object tracking to learn hierarchical features by using tree structure based Recursive Neural Networks (RNN), which have fewer parameters and do not require any network pre-training on auxiliary data. First, RNN parameters are learned to discriminate between target and background only based on the target annotation in the first frame of a test sequence. Tree structure over local patches of an exemplar region is randomly generated by using a bottom-up greedy search strategy. Given the learned RNN parameters, two dictionaries regarding target regions and corresponding local patches are created based on learned hierarchical features from both top and leaf nodes of multiple random trees. In each of the subsequent frames, sparse dictionary coding is conducted on

all candidates to select the best candidate as the new target location. In addition, two dictionaries are online updated to handle appearance changes of target objects. Experimental results demonstrate that the proposed feature learning algorithm can significantly improve tracking performance on benchmark datasets.

The last piece of work investigates background subtraction based object detection, in which foreground regions are obtained by subtracting background models from input images. Sometimes, there are more than one object in a foreground region. However, it is difficult to separate the occluded objects in a foreground region based on color images. In this work, an occlusion handling method is proposed for object detection to separate occluded objects in foreground regions based on “3-D” data comprising image plane coordinates and depth values. First, foreground regions are obtained by using background subtraction on depth data. Then, a so-called “split-merge” approach is proposed to over-segment a foreground region into subregions and then cluster them into a number of object regions by using the proposed boundary based similarity metric. As a result, occluded objects in a foreground region can be effectively detected. Experimental results demonstrate that the proposed occlusion handling method can significantly improve object detection performance.

Contents

Acknowledgments	i
Abstract	ii
List of Figures	ix
List of Tables	xvi
List of Abbreviations	xxi
1 Introduction	1
1.1 Background	1
1.2 Motivation	2
1.2.1 Object Tracking	2
1.2.2 Object Detection	7
1.3 Major Contributions of the Thesis	8
1.4 Organization of the Thesis	11
2 Literature Survey	12
2.1 Object Tracking	12

2.1.1	Single Object Tracking	12
2.1.2	Multiple Object Tracking	18
2.2	Object Detection	19
2.2.1	Color Image Based Methods	19
2.2.2	Depth Data Based Methods	20
2.3	Concluding Remarks	21
3	Visual Object Tracking Using Learned Hierarchical Features with Domain Adaptation	22
3.1	Introduction	23
3.2	Related Work	26
3.3	Proposed Algorithm	28
3.3.1	Tracking System Overview	29
3.3.2	Learning Features for Visual Object Tracking	30
3.3.3	Pre-Learning Generic Features from Auxiliary Videos	31
3.3.4	Domain Adaption Module	33
3.3.5	Optimization and Online Learning	34
3.3.6	Implementation Details	36
3.4	Experiments	37
3.4.1	Evaluation on Learned Hierarchical Features' Robustness to Complicated Motion Transformations	38
3.4.2	Evaluation on Effectivenesses of Temporal Slowness Con- straint and Feature Adaptation Module	44

3.4.3	Evaluation on Capability of Handling Typical Problems in Visual Object Tracking	45
3.4.4	Comparison with the State-of-the-art Trackers	48
3.4.5	Comparison with DLT	49
3.4.6	Failure Cases	49
3.4.7	Evaluation on Learned Features' Generalizability	50
3.4.8	Evaluation on A Larger Benchmark	51
3.5	Concluding Remarks	52
4	Multiple Object Tracking Using Learned Hierarchical Features with A Joint Learning Strategy	54
4.1	Introduction	55
4.2	Feature Pre-learning from Auxiliary Data	57
4.3	Feature Adaptation Using the Proposed Joint Learning Strategy . .	58
4.4	Optimization	60
4.5	Experiments	62
4.6	Concluding Remarks	64
5	Visual Object Tracking Using Learned Hierarchical Features with Recursive Neural Networks	66
5.1	Introduction	67
5.2	Related Work	69
5.3	Learning Hierarchical Features Using RNN	70
5.3.1	Overview of RNN	71
5.3.2	Extracting Features Using RNN	71

5.3.3	Generating Random Trees	73
5.3.4	Discriminative Learning of RNN Parameters	75
5.4	Proposed Tracking System	76
5.5	Discussions	78
5.6	Experiments	79
5.7	Concluding Remarks	83
6	Object Detection with Occlusion Handling by Over-segmentation and Clustering on Foreground Regions	84
6.1	Introduction	85
6.2	Related Work	86
6.3	Proposed Method	87
6.3.1	Boundary Based Similarity Metric	87
6.3.2	Subregion Clustering	91
6.3.3	Height Constraint of Target Objects	92
6.4	Experiments	93
6.5	Concluding Remarks	101
7	Conclusions and Future Works	102
7.1	Conclusions	102
7.2	Future Works	104
	Author's Publications	106
	Bibliography	108

List of Figures

1.1	Illustrations of the scenarios of the object tracking and detection methods developed in this thesis. (a) Given the location (red bounding box) of the target object “smiling face” in the first frame of an image sequence, the tracking method aims to find the locations (green bounding boxes) of the target object through the image sequence; (b) Foreground regions can be obtained by subtracting the background model from an input image. If there are more than one object in a foreground region, it is necessary to separate the occluded objects accurately.	2
1.2	Illustration of feature representation of a target object using raw pixel values or hand-crafted features, e.g. Histograms of Oriented Gradients (HOG)[1].	3
1.3	Illustration of the limited training data problem for applying deep learning to visual object tracking. Training data is not sufficient in visual object tracking because the target annotation is only available in the first frame of a test sequence.	4

1.4	Illustration of the motivation for the first work in this thesis. First, auxiliary video data and the temporal slowness constraint [2] are used to handle the limited training data problem and pre-learn hierarchical features robust to complicated motion transformations. To include appearance information of specific target objects, a domain adaptation module is proposed to adapt the pre-learned features according to specific target objects.	4
1.5	Illustration of the limited training data problem for applying the adaptation domain based feature learning algorithm to multiple object tracking. The adaptation module for each target object will only use its own annotation data, which is actually still limited for feature adaptation.	5
1.6	Illustration of the the motivation for the second work in this thesis. A joint learning strategy using all target annotation data together is proposed for multiple object tracking to jointly conduct the feature adaptation, which can simultaneously learn common features shared by all target objects and individual features for each object.	6
1.7	Illustration of the the motivation for the third work in this thesis. A feature learning algorithm is proposed for visual object tracking to learn hierarchical features by using tree-structure based Recursive Neural Networks (RNN) sharing the same set of parameters through the stacked architecture. As a result, RNN has fewer parameters and does not require to pre-learn features from auxiliary data.	7

1.8	Illustration of the motivation for the last work in this thesis. 3-D data could be more useful for separating occluded objects than 2-D image based features because that occluded objects occupy different depth values. To separate occluded objects in a foreground region, it is straightforward to cluster all pixels in a foreground region into a number of object regions by using “3-D” data comprising image plane coordinates and depth values. However, the number of pixels in a foreground region could be extremely large. It is difficult for the existing clustering algorithms to handle this kind of input data. To solve this problem, an occlusion handling method is proposed for object detection.	8
3.1	Overview of the proposed feature learning algorithm. First, generic features are pre-learned from the auxiliary data obtained from Hans van Hateren natural scene videos [3]. A number of learned feature filters from two layers are visualized. Then, the pre-learned features are adapted according to the specific target object. The final adapted feature filters are also visualized, from which it can be found that the adapted features are more relevant to the specific target object “face” as they contain more facial edges and corners in the first layer and more semantic elements which look like faces or face parts in the second layer.	24

3.2	Illustration of the stacked architecture of the two-layer neural networks and the feature adaptation module. In the stacked architecture, the output of the first layer is whitened using PCA and then used as the input of the second layer. For the adaptation module, given the pre-learned features from auxiliary data and a short sequence of a target object, the pre-learned features are adapted respectively in both two layers by minimizing the objective function in Equation 3.4.	31
3.3	Qualitative results on sequences with non-rigid object deformation. The purple, green, cyan, blue and red bounding boxes refer to ASLA[4]_RAW, ASLA[4]_HOG, ℓ_1 -APG[5], CT-DIF[6] and the proposed tracker respectively. This figure is better viewed in color. . .	41
3.4	Qualitative results on sequences with in-plane rotations. The purple, green, cyan, blue and red bounding boxes refer to ASLA[4]_RAW, ASLA[4]_HOG, ℓ_1 -APG[5], CT-DIF[6] and the proposed tracker respectively. This figure is better viewed in color.	42
3.5	Qualitative results on sequences with out-of-plane rotations. The purple, green, cyan, blue and red bounding boxes refer to ASLA[4]_RAW, ASLA[4]_HOG, ℓ_1 -APG[5], CT-DIF[6] and the proposed tracker respectively. This figure is better viewed in color.	43
3.6	Qualitative results on sequences with typical issues such as illumination change, occlusion and cluttered background. The green, cyan, blue and red bounding boxes refer to ASLA[4]_HOG, ℓ_1 -APG [5], CT-DIF [6] and the proposed tracker respectively. This figure is better viewed in color.	47

3.7	Failure cases due to heavy occlusions. The purple, green, cyan, blue and red bounding boxes refer to ASLA[4]_RAW, ASLA[4]_HOG, ℓ_1 -APG [5], CT-DIF [6] and the proposed tracker respectively. This figure is better viewed in color.	50
3.8	Average center error (in pixels) and average overlap rate (%). The performances between the original IVT [7] and the integrated IVT (deepIVT) using the learned features are compared.	51
3.9	Precision and success plots of the tracking results from the top 10 trackers on all sequences of the benchmark [8].	52
4.1	Overview of the proposed feature learning algorithm. First, the generic feature W_p is pre-learned from K auxiliary video sequences. Then, the individual feature $W_{i,i=1,\dots,M}$ for each target object and the common feature W_c shared by M target objects are simultaneously learned in the feature adaptation module using a joint learning strategy. Each row of a transformation matrix W can be visualized as a filter patch as shown in the figure. There are 25 filter patches which are visualized for the corresponding W	56
4.2	Illustration of the basic learning module and the two-layer architecture. The input to the first layer is raw pixel values of local patches (16×16), which are obtained by convolutions in the holistic image region (32×32). The input to the second layer is the output of the first layer whitened by PCA. The output of the first layer is the concatenated vector of mapping features from all local patches (16×16) in the holistic image (32×32).	57

4.3	Qualitative results obtained from the proposed tracker on nine videos. The rectangles with different colors denote multiple target objects. The figure is better viewed in color.	65
5.1	Illustration of extracting hierarchical features using RNN with a known tree structure over the local patches of a target object. . . .	72
5.2	Illustration of the spatial layout of local patches in a target object and the corresponding adjacency matrix. For example, patch 1 and 2 can be merged because they are spatially neighboring, so the cor- responding matrix element is 1.	74
5.3	Precision and success plots of the tracking results from the top 10 trackers on all sequences of the benchmark [8].	80
5.4	Precision and success plots of the tracking results from different variants of the proposed tracker on all sequences of the benchmark [8].	82
5.5	Qualitative results comparing the proposed tracker (red) with the other 4 state-of-the-art trackers: MEEM [9] (Green), TGPR [10] (dark blue), KCF [11] (orange), Struck [12] (light blue) on 10 se- quences (car4, soccer, ironman, singer1, jogging-1, motorRolling, walking2, freeman4, carScale, tiger1).	83
6.1	Illustration of the proposed method for object detection on fore- ground regions with occlusion handling. In the output, the bound- ing boxes with yellow dashed lines refer to the foreground regions; the bounding boxes with different colors and solid lines refer to the detected objects.	87

6.2	Illustration of the boundary based similarity metric. (a) Over-segmented subregions in a foreground region; (b) The subset of K ($K = 10$) pairs of points from the i th to the j th boundary; (c) The subset of K ($K = 10$) pairs of points from the j th to the i th boundary.	88
6.3	The plot of precision versus recall on different methods	95
6.4	Illustration of the qualitative results. The first and the third rows refer to the results from Felzenszwalb’s method [13] which only uses the color image data. The second and the last rows refer to the corresponding results from the proposed method using the “3-D” data.	97
6.5	Illustration of the tracking-by-detection results. The first and the third rows refer to the results based on the detections from Felzenszwalb’s method [13] which only uses the color image data. The second and the last rows refer to the corresponding results based on the detections from the proposed method using the “3-D” data. . .	100

List of Tables

3.1	Average center error (in pixels). The best two results are shown in red and blue fonts. The proposed tracker using the learned features is compared with 4 state-of-the-art trackers using the other feature representations: the raw pixel values (ASLA[4]_RAW), the hand-crafted HOG feature (ASLA[4]_HOG), the sparse representation (ℓ_1 -APG[5]) and the data-independent feature (CT-DIF[6]). The results are also presented for the variant of the proposed tracker (Ours_VAR) which does not use the temporal slowness constraint in the feature learning.	39
3.2	Average overlap rate (%). The best two results are shown in red and blue fonts. The proposed tracker using the learned features is compared with 4 state-of-the-art trackers using the other feature representations: the raw pixel values (ASLA[4]_RAW), the hand-crafted HOG feature (ASLA[4]_HOG), the sparse representation (ℓ_1 -APG[5]) and the data-independent feature (CT-DIF[6]). The results are also presented for the variant of the proposed tracker (Ours_VAR) which does not use the temporal slowness constraint in the feature learning.	40

3.3	Average center error (in pixels). The best two results are shown in red and blue fonts. The proposed tracker's performances with (Ours_adp) and without (Ours_noadp) the adaptation module are presented respectively. Also, the proposed tracker is compared with 4 baseline trackers using the other features such as the raw pixel values (ASLA[4]_RAW), the hand-crafted HOG feature (ASLA[4]_HOG), the sparse feature (ℓ_1 -APG [5]) and the data-independent feature (CT_DIF[6]).	45
3.4	Average overlap rates. (%) The best two results are shown in red and blue fonts. The proposed tracker's performances with (Ours_adp) and without (Ours_noadp) the adaptation module are presented respectively. Also, the proposed tracker is compared with 4 baseline trackers using the other features such as the raw pixel values (ASLA[4]_RAW), the hand-crafted HOG feature (ASLA[4]_HOG), the sparse feature (ℓ_1 -APG [5]) and the data-independent feature (CT_DIF[6]).	45
3.5	Average center error (in pixels). The best two results are shown in red and blue fonts. The proposed tracker is compared with FragT [14], IVT [7], ℓ_1 T [15], MIL [16], TLD [17], VTD [17], LSK [18], CT [6], ASLA [4] ℓ_1 APG [5], MTT [19], SCM [20], OSPT [21] and LSST [22]. The proposed tracker outperforms the state-of-the-art tracking algorithms.	48

3.6	Average overlap rate (%). The best two results are shown in red and blue fonts. The proposed tracker is compared with FragT [14], IVT [7], ℓ_1 T [15], MIL [16], TLD [17], VTD [17], LSK [18], CT [6], ASLA [4] ℓ_1 APG [5], MTT [19], SCM [20], OSPT [21] and LSST [22]. The proposed tracker outperforms the state-of-the-art tracking algorithms.	49
3.7	Comparison between DLT [23] and the proposed tracker on 8 sequences. The better results are shown in red fonts. “BetterCount” means the number of sequences on which the performance of the current tracker is better than the other one.	49
4.1	Performances of different trackers on 9 videos. It is reported in terms of two measurements: average center errors (ACE, lower is better) and average success rates (ASR, higher is better). ACE is the average distance in pixels between centers of the tracked and the ground truth bounding boxes. ASR is the average percentage of frames which are labeled as success if the overlap between the tracked and the ground truth bounding boxes is larger than 50%. ACE is averaged over all target objects and over all frames. ASR is averaged over all target objects. The best two results are shown in red and blue fonts.	63

5.1	Average precision scores on different attributes: background clutter (BC), deformation (DEF), fast motion (FM), in-plane rotation (IPR), illumination variation (IV), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out-of-view (OV) and scale variation (SV). The best and the second best results are in red and green respectively.	81
5.2	Average success rates on different attributes: background clutter (BC), deformation (DEF), fast motion (FM), in-plane rotation (IPR), illumination variation (IV), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out-of-view (OV) and scale variation (SV). The best and the second best results are in red and green respectively.	81
6.1	The average precisions on different methods	96
6.2	The run-time performances on different methods	98

6.3 Tracking performances based on the detection results from Felzenswalb’s method [13] and the proposed one. Note that only the trajectories with more than 10 frames are evaluated. “Traj_ID” denotes the index of the trajectory of the specific target object. “FN_GT” means the number of the frames that the trajectory occupies in the ground truth. Similarly, “FN_Fels[13]” and “FN_Ours” indicate the number of the frames which the trajectory occupies in the results respectively from Felzenswalb’s method [13] and the proposed one. Correspondingly, “SR_Fels[13]” and “SR_Ours” denote the success rates which are respectively defined as $\frac{FN_Fels[13]}{FN_GT}$ and $\frac{FN_Ours}{FN_GT}$. For the left four columns, higher values indicate better performances, which are in bold fonts. 99

List of Abbreviations

AAM	Adaptive Appearance Model
ACE	Average Center Error
APG	Accelerated Proximal Gradient
ASLA	Adaptive Structural Local Appearance
ASR	Average Success Rate
AUC	Area Under Curve
BC	Background Clutter
BFGS	Broyden-Fletcher-Goldfarb-Shanno algorithm
CKN	Convolutional Kernel Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CT	Compressive Tracker
CXT	Context Tracker
DEF	Deformation
DIF	Data Independent Feature
DLT	Deep Learning Tracker
FM	Fast Motion
FN	Frame Number
FragT	Fragment Tracker

GPU	Graphics Processing Unit
GT	Ground Truth
HOG	Histogram of Oriented Gradients
IPR	In-Plane Rotation
IV	Illumination Variation
IVT	Incremental Visual Tracker
KCF	Kernelized Correlation Filters
L-BFGS	Limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm
LR	Low Resolution
LSK	Local Sparse K-selection
LBP	Linear Binary Pattern
LSST	Least Soft-threshold Squares Tracking
$\ell_1 T$	ℓ_1 Tracker
MB	Motion Blur
MEEM	Multiple Experts Entropy Minimization
MIL	Multiple Instance Learning
MTT	Multiple Task Tracker
OCC	Occlusion
OPE	One Pass Evaluation
OPR	Out-of-Plane Rotation
OSPT	Online Sparse Prototypes Tracker

OV	Out-of-View
PASCAL	Pattern Analysis, Statistical modelling and Computational Learning
PCA	Principal Component Analysis
P-N	Positive-Negative
RAM	Random-Access Memory
RGB-D	Red Green Blue-Depth
RNN	Recursive Neural Network
R-CNN	Region-based Convolutional Neural Network
SCM	Sparsity-based Collaborative Model
SPOT	Structure Preserving Object Tracking
SR	Success Rates
SSD	Sum of Squared Difference
Struck	Structured output tracking with kernels
SV	Scale Variation
SVM	Support Vector Machine
TLD	Tracking-Learning-Detection
TGPR	Transfer Gaussian Process Regression
VGG	Visual Geometry Group
VOC	Visual Object Classes
VTD	Visual Tracking Decomposition
VTs	Visual Tracking Sampling

Chapter 1

Introduction

1.1 Background

Tracking and detecting objects in image sequence play an important role in a wide range of computer vision applications such as video surveillance, video analytics and human-machine interactions. Although many promising methods have been proposed in object tracking and detection, it is still challenging due to some unsolved problems. For instance, object tracking algorithms usually suffer from complicated motion transformations and appearance changes. Also, object detection methods often fail when encountering severe occlusions. In this thesis, these issues are addressed in four pieces of works.

The scenario of the object tracking methods studied in this thesis is illustrated in Figure 1.1(a). Given the location of the target object in the first frame of an image sequence, the object tracking methods aim to localize single or multiple target objects through the image sequence. Feature representation is an important component in visual object tracking. In this thesis, three feature learning algorithms have been developed for visual object tracking.

The scenario of the object detection method investigated in this thesis is illus-

(a) Scenario of the object tracking methods studied in this thesis

(b) Scenario of the object detection method investigated in this thesis

Figure 1.1: Illustrations of the scenarios of the object tracking and detection methods developed in this thesis. (a) Given the location (red bounding box) of the target object “smiling face” in the first frame of an image sequence, the tracking method aims to find the locations (green bounding boxes) of the target object through the image sequence; (b) Foreground regions can be obtained by subtracting the background model from an input image. If there are more than one object in a foreground region, it is necessary to separate the occluded objects accurately.

trated in Figure 1.1(b). Foreground regions can be obtained by subtracting the background model from an input image. If there are more than one object in a foreground region, it is necessary to separate the occluded objects accurately. In this thesis, an occlusion handling method has been developed for background subtraction based object detection.

1.2 Motivation

1.2.1 Object Tracking

Feature representation is an important component of visual object tracking. Traditionally, raw pixel values or hand-crafted features are used to represent target

Figure 1.2: Illustration of feature representation of a target object using raw pixel values or hand-crafted features, e.g. Histograms of Oriented Gradients (HOG)[1].

objects as shown in Figure 1.2. For example, hand-crafted feature, e.g. Histograms of Oriented Gradients (HOG) [1], can be very useful in some tasks such as human detection. However, they cannot handle large appearance variations of arbitrary targets in visual object tracking. Recently, deep learning [24] has achieved very promising performance in speech recognition [25] and image classification [26]. Inspired from this, three works in this thesis will be developed to learn hierarchical features for visual object tracking by using deep learning.

Deep neural networks usually requires a lot of training data to learn a large number of parameters. However, training data in visual object tracking is not sufficient because the target annotation is only available in the first frame of a test sequence as illustrated in Figure 1.3. To address this problem, in the first work of this thesis, a feature learning algorithm is developed for visual object tracking based on domain adaptation. First, auxiliary video data and the temporal slowness constraint [2] are used to pre-learn hierarchical features robust to complicated motion transformations. To include appearance information of specific target objects, a domain adaptation module is proposed to adapt the pre-learned features according to specific target objects. As a result, learned hierarchical features are

Figure 1.3: Illustration of the limited training data problem for applying deep learning to visual object tracking. Training data is not sufficient in visual object tracking because the target annotation is only available in the first frame of a test sequence.

Figure 1.4: Illustration of the motivation for the first work in this thesis. First, auxiliary video data and the temporal slowness constraint [2] are used to handle the limited training data problem and pre-learn hierarchical features robust to complicated motion transformations. To include appearance information of specific target objects, a domain adaptation module is proposed to adapt the pre-learned features according to specific target objects.

Figure 1.5: Illustration of the limited training data problem for applying the adaptation domain based feature learning algorithm to multiple object tracking. The adaptation module for each target object will only use its own annotation data, which is actually still limited for feature adaptation.

robust to both complicated motion transformations and appearance changes of specific target objects. The motivation of the first work is illustrated in Figure 1.4.

Given multiple target objects in a test sequence, the feature learning algorithm developed in the first work can be straightforwardly used to learn hierarchical features for multiple object tracking as shown in Figure 1.5. First, hierarchical features are learned from auxiliary video data. Then, the pre-learned features are independently adapted according to multiple target objects. In this way, the adaptation module for each target object will only use its own annotation data, which is actually still limited for feature adaptation. To solve this problem, a joint learning strategy using all target annotation data together is proposed for multiple object tracking to jointly conduct the feature adaptation, which can simultaneously learn common features shared by all target objects and individual features for each

Figure 1.6: Illustration of the the motivation for the second work in this thesis. A joint learning strategy using all target annotation data together is proposed for multiple object tracking to jointly conduct the feature adaptation, which can simultaneously learn common features shared by all target objects and individual features for each object.

object. As a result, the feature adaptation module can achieve better performance since it makes use of more training data. The motivation of the second work is illustrated in Figure 1.6.

As mentioned before, training data is not enough in visual object tracking. Hence, the feature learning algorithms developed in the first and second works require pre-learning features from auxiliary data. However, this kind of pre-training is inconvenient for visual object tracking. In the third work of this thesis, a feature learning algorithm is proposed for visual object tracking to learn hierarchical features by using tree-structure based Recursive Neural Networks (RNN). The stacked architecture of RNN is established by applying the same set of parameters over a tree structure merging local patches into a target region. Therefore, RNN has fewer parameters than other deep neural networks and does not require to pre-learn features from auxiliary data. The motivation of the third work is illustrated

Figure 1.7: Illustration of the the motivation for the third work in this thesis. A feature learning algorithm is proposed for visual object tracking to learn hierarchical features by using tree-structure based Recursive Neural Networks (RNN) sharing the same set of parameters through the stacked architecture. As a result, RNN has fewer parameters and does not require to pre-learn features from auxiliary data.

in Figure 1.7.

1.2.2 Object Detection

Although many promising object detection methods using color images have been proposed, handling object occlusion remains a challenging issue. Recently, new sensors, such as the Microsoft Kinect, are able to obtain not only color images but also depth data. 3-D data could be more useful for separating occluded objects than 2-D image based features because that occluded objects occupy different depth values as illustrated in Figure 1.8. Although image plane coordinates and depth values can be considered constituting a 3-dimensional space, it is not a true Euclidean space. Hence it is written as “3-D” in this thesis. To separate occluded objects in a foreground region, it is straightforward to cluster all pixels in a foreground region into a number of object regions by using “3-D” data comprising image plane coordinates and depth values. However, the number of pixels in a foreground region could be extremely large. Due to potential computational requirements and ambiguities of determining number of clusters, the

Figure 1.8: Illustration of the motivation for the last work in this thesis. 3-D data could be more useful for separating occluded objects than 2-D image based features because that occluded objects occupy different depth values. To separate occluded objects in a foreground region, it is straightforward to cluster all pixels in a foreground region into a number of object regions by using “3-D” data comprising image plane coordinates and depth values. However, the number of pixels in a foreground region could be extremely large. It is difficult for the existing clustering algorithms to handle this kind of input data. To solve this problem, an occlusion handling method is proposed for object detection.

existing clustering algorithms cannot easily handle this kind of input data. To solve this problem, an occlusion handling method is proposed for object detection by using “3-D” data. The motivation for the last work in this thesis is illustrated in Figure 1.8.

1.3 Major Contributions of the Thesis

The major contributions of this thesis are summarized as follows.

- A feature learning algorithm is developed for visual object tracking based on

domain adaptation. First, generic features are learned from auxiliary video data under the temporal slowness constraint. Then, a domain adaptation module is proposed to adapt pre-learned features according to specific target objects. Finally, learned hierarchical features are robust to both complicated motion transformations and appearance changes of specific target objects. Moreover, this framework using feature pre-learning and adaptation is proposed to solve the limited training data problem for applying deep learning to visual object tracking. Experimental results demonstrate that the proposed feature learning algorithm can significantly improve visual object tracking performance.

- A feature learning algorithm is developed for multiple object tracking by using a joint learning strategy. First, generic features are learned from auxiliary video data. Then, a domain adaptation module is proposed to jointly conduct the feature adaptation based on all annotation data of multiple target objects by using a joint learning strategy, which can simultaneously learn common features shared by all target objects and individual features for each object. Finally, learned hierarchical features are discriminative and compact for multiple object tracking. Moreover, the proposed joint learning strategy can achieve better performance of feature adaptation because it makes use of more training data by putting annotation data from all target objects together. Experimental results demonstrate that the proposed feature learning algorithm with the joint learning strategy can significantly improve multiple object tracking performance.
- A feature learning algorithm is developed for visual object tracking by using Recursive Neural Networks (RNN). First, RNN parameters are learned to discriminate between target and background by using the target annotation

in the first frame of a test sequence. Tree structure over local patches of an exemplar region is randomly generated by using a bottom-up greedy search strategy. Given the learned RNN parameters, two dictionaries regarding target regions and corresponding local patches are created based on learned hierarchical features from both top and leaf nodes of multiple random trees. In each of the subsequent frames, sparse dictionary coding is conducted on all candidates to select the best candidate as the new target location. In addition, two dictionaries are online updated to handle appearance changes of target objects. Learned hierarchical features can successfully encode spatial information over local patches of target regions. Moreover, the proposed feature learning algorithm does not pre-learn features from auxiliary data as RNN has fewer parameters and requires less training data. Experimental results demonstrate that the proposed feature learning algorithm can significantly improve visual object tracking performance.

- An occlusion handling method is developed for object detection by using “3-D” data. First, foreground regions are obtained by using background subtraction based on depth data. Then, each foreground region is over-segmented into subregions. Next, a number of segments are obtained by clustering subregions based on a boundary based similarity metric using “3-D” data comprising image plane coordinates and depth values. Finally, occluded objects in a foreground region can be separately detected. The proposed method handles object occlusion effectively by using depth data. Moreover, the proposed boundary based similarity metric can be robust to noisy depth values. Experimental results demonstrate that the proposed occlusion handling method using “3-D” data can significantly improve object detection performance.

1.4 Organization of the Thesis

This thesis is organized as follows. First, a literature survey on object tracking and detection is presented in Chapter 2. Then, three feature learning algorithms for visual object tracking are described in Chapter 3, 4 and 5 respectively. Next, an occlusion handling method using “3-D” data for object detection is detailed in Chapter 6. Finally, conclusions and future works are given in Chapter 7.

Chapter 2

Literature Survey

In this chapter, a literature survey on object tracking and detection is presented. First, object tracking methods are reviewed in two categories: single and multiple object tracking. Then, object detection methods are also reviewed in two categories: color image and depth data based methods.

2.1 Object Tracking

Given the initial locations of single or multiple target objects in the first frame of a test sequence, object tracking aims to find the locations of the target objects through the test sequence. The related works are reviewed in terms of single and multiple object tracking as below.

2.1.1 Single Object Tracking

A huge number of methods have been reported in literature for single object tracking. First, the related works can be divided into three categories, the generative model based methods, the discriminative model based methods and the hybrid generative-discriminative based methods.

Generative Model Based Methods. Generative model based methods aim to model appearance distributions of target objects as much as possible. Some of the generative model based methods are as follows. Black and Jepson [27] address the appearance change problem by using the subspace techniques. McKenna et al. [28] utilize the adaptive Gaussian mixtures to model the color distribution of target objects. Hou et al. [29] develop a so-called direct appearance model to utilize texture information directly for estimating position and appearance of target objects. Jepson et al. [30] propose to learn the adaptive appearance models for motion-based object tracking. Comaniciu et al. [31] present how to regularize the histogram-based representations by using spatial masking with an isotropic kernel and formulate the target localization problem as a mean-shift based optimization problem. Yang et al. [32] show how to track more general motion models by using a new similarity measure in spatial-feature spaces. Lee and Kriegman [33] elaborate how to learn offline a generic representation of appearance for the class-specific object and incrementally learn the appearance model online from the video of an instance of the specific class. Wang et al. [34] develop a spatial-color mixture of Gaussians appearance model based similarity measure to improve particle filters. Yilmaz [35] propose an asymmetric kernel mean-shift based object tracking method to consider the scale and orientation of the kernel adaptively change. Ross et al. [7] demonstrate how to incrementally learn a low-dimensional subspace representation for handling the appearance changes of the target objects. Mei and Ling [15] formulate object tracking into a sparse approximation problem under a particle filter framework. Kwon and Lee [17] present a visual tracking decomposition strategy for the observation and the motion model. Li et al. [36] propose to utilize the signal recovery power of compressive sensing for visual object tracking and adopt dimensionality reduction and orthogonal matching pursuit for accelerating their tracker. Zhang et al. [19] formulate object tracking into a

multi-task sparse learning based particle filter framework and learn the particle representations jointly by using the sparsity-inducing $l_{p,q}$ mixed norms. Li et al. [37] develop an incremental 3D discrete cosine transform (3D-DCT) algorithm to reduce the computational complexity of visual tracking by decomposing the 3D-DCT into the 2D discrete cosine transform and 1D discrete cosine transform on the input video data. Besides the methods mentioned above, there are many other generative model based methods. Interested readers can refer to the survey paper [38] for more details.

Discriminative Model Based Methods. Discriminative model based methods formulate visual object tracking into a binary classification problem, in which the appearance model aims to separate between the object and non-object regions. Some of the discriminative model based methods are as follows. Avidan [39] proposes to utilize the support vector machine (SVM) to facilitate an optic-flow-based tracker by maximizing the SVM classification score. Lin et al. [40] show how to apply incremental fisher linear discriminant algorithm into visual object tracking. Williams et al. [41] elaborate how to integrate the probabilistic relevance vector machine into the real-time tracking systems. Wang et al. [42] demonstrate how to select discriminative features in an online learning by using Fisher discriminant to rank the classification capacity of features. Grabner and Bischof [43] present how to select AdaBoost features through online learning for visual object tracking. Nguyen and Smeulders [44] describe how to utilize linear discriminant analysis to train the foreground/background texture discrimination dynamically for visual object tracking. Tang et al. [45] propose to classify newly arrived data of the target object and update the classifier simultaneously in a co-training framework. Zhang et al. [46] show how to utilize graph embedding based discriminative learning to simultaneously learn the subspace of the target object and its local discriminative structure against the background. Grabner et al. [47] develop an on-line

semi-supervised boosting method to alleviate the drifting problem in visual object tracking. Li et al. [48] elaborate how to utilize incremental 2D-LDA learning and Bayes inference for handling appearance changes of the target object. Babenko et al. [16] demonstrate how to utilize multiple instance learning to handle the incorrectly labeled training examples caused by the inaccuracies in the tracker. Liu et al. [49] present a boosting based tracking method in co-training framework to online update and boost the tracker with multi-view weak hypothesis. Zeisl et al. [50] describe how to combine semi-supervised learning and multiple instance learning into an integrated framework. Zha et al. [51] propose to formulate the object tracking problem as a transductive learning problem. Luo et al. [52] show how to conduct transfer learning between the source data in the previous frames and the target data in the current frame. Hare et al. [12] elaborate how to utilize a kernelized structured output support vector machine to adaptively predict the tracking output. Bai and Tang [53] develop a so-called online Laplacian ranking support vector tracker to handle occlusions and fluctuations of object appearance. Jiang et al. [54] demonstrate how to regularize metric learning based on sparsity to obtain the lowest possible dimensionality of the feature space projection. Besides these methods, there exist many other discriminative model based methods. Interested readers may refer to the survey paper [38] for more details.

Hybrid Generative-Discriminative Model Based Methods. The hybrid generative-discriminative model based methods aim to find a good trade-off between the generative and the discriminative appearance models. Some of the hybrid generative-discriminative model based methods are as follows. Lin et al. [55] develop an adaptive discriminative generative model to separate the target from the background. Grabner et al. [56] propose to utilize the generative and discriminative methods to select the features during tracking. Lei et al. [57] present a three-level hierarchy to combine the discriminative and generative models for track-

ing. Yu et al. [58] elaborate how to label incoming data by using a co-training based approach and online update a hybrid discriminative generative model to learn appearance changes. Yang et al. [59] demonstrate how to use positive and negative data for discriminative subspace tracking. Shen et al. [60] show how to use the support vector machine to enhance the kernel-based mean shift tracker. For more details, interested readers may refer to the survey paper [38].

Moreover, these related works can be reviewed respectively in terms of three components of single object tracking: feature representation, motion model and appearance model update.

Feature Representation. Feature representation is very important for visual object tracking. Some related works are as follows. Lucas and Kanade [61] propose to use the spatial intensity gradient of the holistic images for image registration. Black and Jepson [27] show how to use a view-based representations to track rigid and articulated objects. Comaniciu et al. [31] demonstrate how to represent the target object by the color histogram based features. Collins et al. [62] elaborate how to online select the features and adjust the set of features to improve tracking performance. Adam et al. [14] describe how to represent the target object by using multiple image fragments or patches. Grabner et al. [63] present how to utilize AdaBoost learning to online select the most discriminative features for stable tracking. Porikli et al. [64] explain how to represent the target object by using the covariance matrix of features to efficiently fuse different types of features. Avidan [65] propose to combine the weak features into the strong features by using AdaBoost to distinguish between the object and the background. Tang et al. [45] show how to combine different features by using a classifier weighting method. Ross et al. [7] demonstrate how to incrementally learn a low-dimensional subspace representation to adapt to the appearance change of the target object. Li et al. [66] elaborate how to utilize a cascade particle filter to fuse different sub-

sets of features. Mei and Ling [15] describe how to represent the target object in the space spanned by target templates and trivial templates. Kwon and Lee [67] present how to represent the target object by local patches to handle geometric appearance changes. Zhang et al [19] explain how to use multi-task learning to jointly learn the particle representations in a particle filter framework. Wu et al. [68] propose to incrementally learn a low-dimensional covariance tensor representation to efficiently adapt to the on-line appearance changes. Wang et al. [69] show how to represent the target object by using the sparse prototypes learned online with update. Wang et al. [23] elaborate how to learn deep compact features for visual tracking. Interested readers may refer to the survey paper [8] for more details.

Motion Model. Motion models in visual tracking aims to generate the candidate locations of the target object. Some related works are as follows. Isard and Blake [70] propose to utilize a randomly generated set to interpret the locations of the target object. Fan et al. [71] develop a gradient-based searching strategy to locate the set of local attentional regions. Hare et al. [12] present how to utilize a kernelized structured output support vector machine to jointly estimate object positions and predict classifier labels. Besides these methods, there are many other trackers using different motion models. For more details, interested readers may refer to the survey paper [8].

Appearance Model Update. Updating the appearance model is very important for visual tracking to handle the appearance changes. Some related works are as follows. Lucas and Kanade [61] propose to update the templates by using the fixed template from the first frame and the one from the most recent frame. Jepson et al. [72] present how to utilize an on-line EM-algorithm to update the appearance model over time. Matthews et al. [73] show how to update the templates to solve the drifting problem in visual tracking. Grabner et al. [63] develop an on-

line AdaBoost feature selection algorithm to update the appearance model during tracking. Ross et al. [7] demonstrate how to update the appearance model by using an incremental subspace learning algorithm. Grabner et al. [47] describe how to utilize an on-line semi-supervised boosting algorithm to conduct the appearance model update process. Babenko et al. [16] elaborate how to use the multiple instance learning to handle the ambiguities of training samples during updating the appearance model. Kalal et al. [74] explain how to utilize the positive and negative constraints to online update the appearance model. Hare [12] propose to update the object model by using a kernelized structured output support vector machine. For more details, interested readers can refer to the survey paper [8].

2.1.2 Multiple Object Tracking

Recently, some works have been proposed to address multiple object tracking. The number of the related works is relatively smaller and these methods are as follows. Hu et al. [75] propose to use incremental log-euclidean Riemannian subspace learning to track multiple objects in the particle filtering based Bayesian state inference. Zhao et al. [76] present how to manually initialize an object in the first frame and track it in the subsequent few frames to obtain training samples which are used to train an object detector of similar kind of objects. Then, the top N similar objects are detected in the first frame and tracked independently through the video sequence. Luo and Kim [77] show how to utilize tracking-by-detection strategy and multi-task learning to detect and track multiple generic objects. Zhang and van der Maaten [78] demonstrate how to track multiple arbitrary objects jointly by using the spatial constraints between the target objects. Luo et al. [79] describe how to propagate bi-labels of multiple target objects by using clustered multi-task learning. Wang et al. [80] elaborate how to learn hierarchical features for multiple object tracking by using a multi-task learning strategy. Interested reader can also

refer to the survey paper [81].

2.2 Object Detection

Object detection in this thesis focuses on background subtraction based object detection which aims to extract foreground regions by subtracting background models from input images. The related works are reviewed in terms of color image and depth data based methods as below.

2.2.1 Color Image Based Methods

Some related works based on color image are as follows. Haritaoglu et al. [82] propose to represent each pixel of the background model by three values: the minimum and maximum intensity values and the maximum intensity difference between consecutive frames. Stauffer and Grimson [83] present how to model each pixel as a mixture of Gaussians and update the model using an on-line approximation. McKenna et al. [84] show how to combine color and gradient information for background subtraction to handle shadows and unreliable color values. Oliver et al. [85] demonstrate how to create an eigenspace to model the background and then detect the moving objects. Cavallaro and Ebrahimi [86] describe how to extract objects from video sequences by modeling the color difference between frames. Li et al. [87] develop a Bayes decision rule for detection and segmentation of foreground objects. Kim et al. [88] explain how to construct a codebook for background subtraction to handle moving backgrounds and illumination changes. Kim et al. [89] propose to quantize the background values at each pixel into codebooks for foreground-background segmentation. Heikkila and Pietikainen [90] present how to model the background by calculating local binary pattern histograms over a circular region around each pixel. Zivkovic and van der Heijden [91] develop a para-

metric Gaussian mixture model based algorithm and a non-parametric adaptive density estimation method for background subtraction. Maddalena and Petrosino [92] show self-organization based artificial neural networks to handle moving backgrounds, illumination variations and shadows caused by moving objects. Ilyas et al. [93] demonstrate how to modify the codebook model by accessing, deleting, matching and adding codewords in codebook. Barnich and Van Droogenbroeck [94] describe how to build a samples-based estimation of the background by using a random policy for selecting pixel values. Pham et al. [95] explain how to infer the latent distribution and its consistent region by using the given reference distribution for foreground-background segmentation. Farcas et al. [96] propose to utilize a discriminative subspace learning model to classify pixels as background or foreground. Interested readers can refer to the survey paper [97] for more details.

2.2.2 Depth Data Based Methods

Depth data is utilized to facilitate background subtraction based object detection. Some related works are as follows. Eveland et al. [98] propose to statistically model background by using stereo sequences to segment foreground objects. Gordon et al. [99] present how to estimate background by using a range and color clustering at each image pixel. Ivanov et al. [100] show how to construct disparity fields for background subtraction to handle rapid illumination changes. Darrell et al. [101] demonstrate how to detect foreground points quickly by using pruned disparity search. Harville et al. [102] describe how to model the background by using per-pixel, time-adaptive, Gaussian mixtures based on both depth and color space. Kolmogorov et al. [103] elaborate how to fuse color and stereo matching information to accurately and efficiently segment foreground from background layers. Lim et al. [104] explain how to utilize the depth information obtained from stereo matching to eliminate detection errors in background subtraction. Bahadori et al.

[105] propose to dynamically update the background model and extract foreground objects by analyzing the stereo data. Crabb et al. [106] present how to segment foreground from background by using depth information of a TOF-sensor. Zhu et al. [107] show how to combine passive stereo information, active range sensor and matte for foreground-background segmentation. Schiller and Koch [108] demonstrate how to fuse the foreground-background segmentation results from both color images and depth data by weighting these two measures. Greff et al. [109] describe how to adapt the depth images into the existing background subtraction algorithms and evaluate their performances. Fernandez-Sanchez et al. [110] elaborate how to extend a range and color data based background subtraction algorithm by using a post-processing mask fusion stage. Murgia et al. [111] explain how to incorporate depth information into the traditional color image based codebook model for background subtraction.

2.3 Concluding Remarks

In this chapter, a literature survey on object tracking and detection is presented. First, the related works in object tracking are reviewed in two categories: single and multiple object tracking. Then, the related works in object detection are reviewed in terms of color image and depth data based methods. It can be said there was never short of methods in this research area, yet, there are still some challenging problems in tracking and detecting objects that remain unsolved or methods for better performance. The remaining chapters of this thesis account some of these challenges.

Chapter 3

Visual Object Tracking Using Learned Hierarchical Features with Domain Adaptation

In this chapter, a feature learning algorithm is developed for visual object tracking. First, hierarchical features are learned from auxiliary video data by using a two-layer neural network. By embedding the temporal slowness constraint into the stacked network architecture makes learned hierarchical features robust to complicated motion transformations. This is important for visual object tracking. Then, a domain adaptation module is proposed to adapt the pre-learned features according to specific target objects. The adaptation is conducted in both two layers of the stacked neural network so as to include appearance information of specific target objects. As a result, learned hierarchical features can be robust to both complicated motion transformations and appearance changes of specific target objects. The proposed feature learning algorithm is integrated into a baseline object tracking method. Experimental results demonstrate that the proposed feature learning algorithm can significantly improve object tracking performance,

especially for image sequences with complicated motion transformations.

3.1 Introduction

Visual object tracking aims to find the locations of an arbitrary-class target object through the image sequence given the bounding box of the target object in the first few frames. Although many promising object tracking methods have been proposed, it is still very challenging to handle complicated motion transformations in visual object tracking. In this chapter, a feature learning algorithm is proposed to enhance the performance of single object tracking.

Learning hierarchical feature representation (also called deep learning) has emerged recently as a promising research direction in computer vision and machine learning. Rather than using hand-crafted features, deep learning aims to learn data-adaptive, hierarchical, and distributed representation from raw data. The learning process is expected to extract and organize discriminative information from data. Deep learning has achieved impressive performance on image classification [26], action recognition [112], speech recognition [25] etc.

Feature representation is an important component for visual object tracking. Deep learning usually requires a lot of training data to learn a large number of parameters. However, in visual object tracking, only the annotations of the target object in the first few frames of the test video sequence are available. Recently, a so-called deep learning tracker (DLT) [23] has been proposed to learn offline generic features from auxiliary natural images. However, using unrelated images for training, DLT cannot obtain hierarchical features with temporal invariance, which is actually very important for visual object tracking. Moreover, DLT does not have an integrated objective function to bridge offline training and online tracking. It conducts the knowledge transfer from offline training to online tracking

Figure 3.1: Overview of the proposed feature learning algorithm. First, generic features are pre-learned from the auxiliary data obtained from Hans van Hateren natural scene videos [3]. A number of learned feature filters from two layers are visualized. Then, the pre-learned features are adapted according to the specific target object. The final adapted feature filters are also visualized, from which it can be found that the adapted features are more relevant to the specific target object “face” as they contain more facial edges and corners in the first layer and more semantic elements which look like faces or face parts in the second layer.

by simply feeding the hierarchical features extracted from the pre-trained encoder to the target object classifier and tuning the parameters of the pre-trained encoder when significant change of the object appearance is detected.

To address these two issues in DLT [23], a domain adaptation based feature learning algorithm is proposed. Figure 3.1 presents an overview of the proposed feature learning algorithm. First, the algorithm aims to learn hierarchical features robust to complicated motion transformations of the target object, which are not considered by DLT [23]. Also, the algorithm intends to learn features which can handle a wide range of motion patterns in the test video sequences. Therefore, the feature learning algorithm proposed by Zou et al. [2] is employed as a basic model to pre-learn features robust to diverse motion patterns from auxiliary video sequences (offline learning part shown in Figure 3.1). Given the corresponding

patches in the training video sequences, the basic model learns patch features invariant between two consecutive frames. As a result, high-level features which are robust to non-linear motion patterns can be discovered. Zou et al. [2] utilize the learned features for generic object recognition. In this chapter, it is argued that learning hierarchical features is also beneficial to visual object tracking, as the temporal robustness can help a tracker to find the corresponding target patch reliably.

As stated above, DLT [23] does not have an extra united objective function connecting offline learning and online tracking. As a result, the learned features from DLT do not include the appearance information of the specific target object. To solve this issue, a domain adaptation module is proposed to effectively adapt the pre-learned features according to the specific target object (online learning part shown in Figure 3.1). The adaptation module is seamlessly incorporated into both the layers of the stacked architecture of the deep learning model. As a result, the adapted features can be robust to both complicated motion transformations and appearance changes of the target object. As shown in Figure 3.1, it can be observed that the adapted features are more relevant to the specific target object “face” as they contain more facial edges and corners in the first layer and more semantic elements which look like faces or face parts in the second layer.

In order to capture the appearance changes of the target object, the learned features are online adapted according to the new coming data of the test video sequence. Due to the high dimensionality of the parameter space of the deep learning model, the limited memory BFGS (L-BFGS) algorithm [113] is employed to solve the optimization problem in the adaptation module. As a result, convergence can be quickly reached in each adaptation.

The proposed feature learning algorithm is validated on benchmark test image sequences. Experimental results demonstrate that significant improvement over

baseline trackers can be obtained by using the learned hierarchical features.

3.2 Related Work

Object Tracking. For decades, many promising algorithms have been proposed for object tracking. Eigentracker [27] has had a deep impact on subspace based trackers [114] [7]. The method known as “Condensation” [70] is well-known because it is the first one to apply particle filter [115] to object tracking. In [31], mean-shift [116] is used to optimize the target localization problem in visual tracking. The “Lucas-Kanade” algorithm [117] is famous for defining the cost function by using the sum of squared difference (SSD). Another pioneering method [30] paves the way for the subsequent trackers based on the adaptive appearance model (AAM).

Recently, the tracking problem has also been considered as a binary classification problem due to the significant improvement on object recognition. In [39], the Support Vector Machine (SVM) is integrated into an optical-flow based tracker. Subsequently, the ensemble tracker [118] trains an ensemble of weak classifiers online to label pixels as the object or the background. In [47], an online semi-supervised boosting method is proposed to handle the drifting problem caused by inaccuracies in updating of the tracker. In [16], on-line multiple instance learning is also proposed to solve the drifting problem. P-N learning [74] is proposed to train a binary classifier from labeled and unlabeled examples which are iteratively corrected by positive (P) and negative (N) constraints.

Many advanced trackers are also developed based on sparse representation [119]. ℓ_1 tracker [15] solves an ℓ_1 -regularized least squares problem to achieve the sparsity for target candidates, in which the one with the smallest reconstruction error is selected as the target in the next frame. Two pieces of works [120] [36] focus on accelerating the ℓ_1 tracker [15] because the ℓ_1 minimization requires high compu-

tational costs. There are some other promising sparse trackers [121] [18] [4]. The tracker [4] that employs the adaptive structural local appearance (ASLA) model achieves especially good performance, and this is used as the baseline tracking system into which the proposed feature learning algorithm is integrated.

Feature Representation. Some tracking methods focus on feature representation. In [62], an online feature ranking mechanism is proposed to select features which are capable of discriminating between object and background for visual tracking. Similarly, an online AdaBoost feature selection algorithm is proposed in [63] to handle appearance changes in object tracking. In [122], keypoint descriptors in the region of the interested object are learned online together with background information. The compressive tracker (CT) [6] employs random projections to extract data independent features for the appearance model and separates the object from the background using a naive Bayes classifier. Recently, Wang and Yeung [23] has proposed to learn deep compact features for visual object tracking.

Deep Learning. Deep learning [123] [24] has recently attracted much attention in machine learning. It has been successfully applied in computer vision applications, such as shape modeling [124], action recognition [112] and image classification [26]. Deep learning aims to replace the hand-crafted features with high-level and robust features learned from raw pixel values. This process is also known as unsupervised feature learning [125] [126] [127] [128]. In [2], the temporal slowness constraint [129] is combined with deep neural networks in order to learn hierarchical features. Inspired by this work, the proposed algorithm aims to learn hierarchical features to handle complicated motion transformations in visual object tracking.

Domain Adaptation. Recently, there has been increasing interest in the visual domain adaptation problem. Saenko et al. [130] use domain adaptation to learn object category models. In [131], domain adaptation techniques are devel-

oped to detect video concepts. In [132], Duan et al. adapt the learned models from web data to recognize visual events. Recently, Glorot et al. [133] develop a meaningful representation for large-scale sentiment classification by combining deep learning and domain adaptation. Domain adaptation has also been applied in object tracking. Wang et al. [134] pre-learn an over-complete dictionary and transfer the learned visual priors for tracking the specific target objects.

3.3 Proposed Algorithm

The principles behind the proposed algorithm are deep learning and domain adaptation learning. The temporal slowness constraint is first utilized to offline learn the generic features which are robust to complicated motion transformations. Then, a domain adaptation module is proposed to adapt the pre-learned features according to the specific target object. The difference between DLT [23] and the proposed algorithm is as follows. First, DLT pre-learns the hierarchical features from untracked images. In contrast, the proposed algorithm uses tracked video sequences and focuses on learning the features robust to complicated motion transformations. Second, DLT does not have a united objective function with the regularization term for domain adaptation, whereas the proposed algorithm has an adaptation module to integrate the specific target object's appearance information into the pre-learned generic features. Also, the proposed algorithm is different from [134] in which the dictionary is pre-defined and then the tracking object is reconstructed by the patterns in the pre-defined dictionary. The algorithm in [134] may fail if the pre-defined dictionary does not include the visual patterns of the target object. Last, it is necessary to mention that Zou et al. [2] learn hierarchical features of tracked objects from image sequences for image classification whereas the proposed algorithm focuses on visual object tracking.

3.3.1 Tracking System Overview

The proposed feature learning algorithm aims to learn hierarchical features to enhance the state-of-the-art tracking algorithms. The tracking system with the adaptive structural local sparse appearance model (ASLA) [4] achieves very good performance. Hence, the proposed feature learning algorithm is integrated into this tracking system. But note that the proposed feature learning algorithm is general for visual object tracking, and it can be used with other tracking systems as well by replacing the original feature representation. In this section, the tracking system is briefly described. The interested readers may refer to [4] for more details.

Suppose there are an observation set of target $x_{1:t} = \{x_1, \dots, x_t\}$ up to the t^{th} frame and a corresponding feature representation set $z_{1:t} = \{z_1, \dots, z_t\}$. The target state y_t can be calculated as follows

$$y_t = \arg \max_{y_t^i} p(y_t^i | z_{1:t}), \quad (3.1)$$

where y_t^i denotes the state of the i^{th} sample in the t^{th} frame. The posterior probability $p(y_t | z_{1:t})$ can be inferred by the Bayes' theorem as follows

$$p(y_t | z_{1:t}) \propto p(z_t | y_t) \int p(y_t | y_{t-1}) p(y_{t-1} | z_{1:t-1}) dy_{t-1}, \quad (3.2)$$

where $z_{1:t}$ denotes the feature representation, $p(y_t | y_{t-1})$ denotes the motion model and $p(z_t | y_t)$ denotes the appearance model. The motion model $p(y_t | y_{t-1})$ encodes the temporal continuity of the target states between consecutive frames by using the affine transformation. The appearance model $p(z_t | y_t)$ is defined as the similarity between the candidate and the exemplars of the target object. The similarity is calculated based on the adaptive structural local sparse appearance model (ASLA), in which overlapping local patches are extracted in a fixed spatial pattern from object regions and then sparse coding is used as the metric measurement. In [4],

the representations $z_{1:t}$ simply use raw pixel values. In next subsection, learning hierarchical features from raw pixels for visual object tracking will be discussed.

3.3.2 Learning Features for Visual Object Tracking

Previous tracking methods usually use raw pixel values or hand-crafted features to represent target objects. However, they cannot capture essential information which is invariant to non-rigid object deformations, in-plane and out-of-plane rotations in object tracking. The proposed algorithm aims to enhance the performance of object tracking by learning hierarchical features which have the capability of handling complicated motion transformations. Moreover, a domain adaptation module is proposed to make feature learning feasible for visual object tracking. First, the approach proposed in [2] is employed to offline learn the generic features from auxiliary video sequences. These features are robust to complicated motion transformations. However, they do not include the appearance information of the specific target object. Hence, a domain adaptation module is used to adapt the pre-learned features according to the specific target object.

The proposed feature learning algorithm is integrated into the baseline tracker ASLA [4] and its details are given in Algorithm 4.

Algorithm 1 The proposed tracker with the feature learning algorithm

- 1: **Input:** the previous tracking state y_{t-1} , the existing feature learning parameter $\hat{\Theta}$ and the exemplar library.
 - 2: Apply the affine transformation on y_{t-1} to obtain a number of tracking states y_t^i and the corresponding candidate image patches x_t^i .
 - 3: Extract feature representations z_t^i from the candidate image patches x_t^i under the existing feature learning parameter $\hat{\Theta}$.
 - 4: Calculate the posterior probability $p(y_t^i | z_{1:t})$ according to Equation 5.11.
 - 5: Predict the tracking state by $y_t = \arg \max_{y_t^i} p(y_t^i | z_{1:t})$.
 - 6: Update the feature learning parameter and the exemplar library every M frames.
 - 7: **Output:** the predicted tracking state y_t , the up-to-date feature learning parameter Θ and the up-to-date exemplar library.
-

Figure 3.2: Illustration of the stacked architecture of the two-layer neural networks and the feature adaptation module. In the stacked architecture, the output of the first layer is whitened using PCA and then used as the input of the second layer. For the adaptation module, given the pre-learned features from auxiliary data and a short sequence of a target object, the pre-learned features are adapted respectively in both two layers by minimizing the objective function in Equation 3.4.

3.3.3 Pre-Learning Generic Features from Auxiliary Videos

Since the appearance of a target object could change significantly due to its motion, a good tracker desires features robust to complicated motion transformations. Zou et al. [2] have claimed that generic features robust to diverse motion patterns can be learned from auxiliary video data. Therefore, the deep learning model in [2] is employed to offline learn hierarchical features from auxiliary video data [3] to handle complicated motion transformations in visual object tracking.

The deep learning model has two layers as illustrated in Figure 3.2. The first layer works on the smaller patches (16×16). The second layer works on the larger patches (32×32). The network architecture used in the proposed feature learning algorithm is same as the one used in the work [2] but different from the one used in DLT tracker [23] which does not consider different input sizes regarding local patches and holistic target regions.

Given the offline training patch $\mathbf{x}_i$ from the i^{th} frame, the corresponding learned

feature is denoted as $\mathbf{z}_i = \sqrt{H(W\mathbf{x}_i)^2}$, where H is the pooling matrix and $(W\mathbf{x}_i)^2$ is the element-wise square on the output of the linear network layer. To better explain the basic learning module in each layer, Figure 3.2 may be used for the reference. First, it is necessary to mention that the blue, yellow and purple circles denote the input vector $\mathbf{x}_i$, the intermediate vector $(W\mathbf{x}_i)^2$ and the output vector $\sqrt{H(W\mathbf{x}_i)^2}$ respectively for the basic learning module. The numbers of the blue, yellow and purple circles are respectively 256, 256 and 128. Then, H can be illustrated as the transformations between the intermediate vector (yellow circles) and the output vector (purple circles). The pooling mechanism of H calculates the summation of two adjacent feature dimensions of the intermediate vector (yellow circles) in a non-overlapping fashion. Also, W can be illustrated as the transformations between the input vector (blue circles) and the intermediate one (yellow circles). The feature transformation matrix W is learned by solving the following unconstrained minimization problem

$$\min_W \lambda \sum_{i=1}^{N-1} \|\mathbf{z}_i - \mathbf{z}_{i+1}\|_1 + \sum_{i=1}^N \|\mathbf{x}_i - W^T W \mathbf{x}_i\|_2^2, \quad (3.3)$$

where $\mathbf{z}_{i+1}$ denotes the learned feature from the $(i+1)^{th}$ frame. The first term forces learned features to be temporally continuous and the second term is an auto-encoder reconstruction cost [127]. As a result, the feature $\mathbf{z}$ is learned to be robust to complicated motion transformations.

The input of the first layer is the raw pixel values of smaller patches (16×16). The feature transformation matrix W^{L1} for the first layer can be learned by Equation 3.3. Then, W^{L1} is convolved with the larger patches (32×32). The larger patch is divided into a number of sub-patches (16×16). W^{L1} is used to conduct feature mapping for each sub-patch and then the features of all the sub-patches are concatenated to represent the larger patch. Next, PCA whitening is applied to the concatenated feature vector. Finally, the whitened feature vector of the larger

patch is used as the input to the second layer and the feature transformation matrix W^{L2} for the second layer is then learned.

The first layer can extract features robust to local motion patterns. From the second layer, the features robust to more complicated motion transformations (See Figure 3.1) can be extracted. The features from two layers are concatenated as generic features. Moreover, the generic features are learned from a large number of auxiliary video data. As a result, the pre-learned features can provide the tracker with the robustness to complicated motion transformations.

3.3.4 Domain Adaption Module

Although the generic features are robust to non-linear motion patterns in visual tracking, they do not include the appearance information of the specific target object, e.g. shape and texture. Hence, a domain adaptation module is proposed to adapt the generic features according to the specific target object. The domain adaptation module is illustrated in Figure 3.2.

Given a target video sequence, ASLA [4] is employed to track the target object in the first N frames and these tracking results are used as the training data for the adaptation module. The adapted feature is denoted as $\mathbf{z}_i^{adp} = \sqrt{H(W\mathbf{x}_i^{obj})^2}$, where $\mathbf{x}_i^{obj}$ indicates the object image patch in the i^{th} frame of the training data for adaptation and W is the feature transformation matrix to be learned. The adaptation module is formulated by adding a regularization term as follows,

$$\begin{aligned}
 W_{adp} = & \arg \min_W \lambda \sum_{i=1}^{N-1} \|\mathbf{z}_i^{adp} - \mathbf{z}_{i+1}^{adp}\|_1 \\
 & + \gamma \sum_{i=1}^N \|W \mathbf{x}_i^{obj} - W_{old} \mathbf{x}_i^{obj}\|_2^2 \\
 & + \sum_{i=1}^N \|\mathbf{x}_i^{obj} - W^T W \mathbf{x}_i^{obj}\|_2^2,
 \end{aligned} \tag{3.4}$$

where the second term refers to the adaptation module and aims to make the adapted feature close to the old one for the sake of preserving the pre-learned features' robustness to complicated motion transformations. Meanwhile, using the training data $\mathbf{x}_i^{obj}$ is intended to include the appearance information of the specific target object, e.g. shape and texture. γ is the trade-off parameter which controls the adaptation level.

The generic features are adapted in a two-layer manner. It means that the minimization in Equation 3.4 with respect to W is conducted in both layers respectively.

3.3.5 Optimization and Online Learning

Succinctly, the objective function of the adaptation module can be denoted as $f(\mathbf{X}; \Theta, \hat{\Theta})$, where $\mathbf{X}$ denotes a number of training images of object regions for the adaptation, $\Theta = \{w_{ij} | i, j = 1, \dots, N\}$ indicates the parameter set representing all entries in the transformation matrix W and $\hat{\Theta}$ refers to the known parameter set w.r.t. W_{old} . The limited-memory BFGS (L-BFGS) algorithm [113] is employed to optimize the objective function $f(\mathbf{X}; \Theta, \hat{\Theta})$ w.r.t. the parameter set Θ .

The Quasi-Newton methods, such as BFGS algorithm [135], need to update the approximate Hessian matrix B_k at the i^{th} iteration to calculate the search direction $p_k = -B_k^{-1} \nabla f_k$, where ∇f_k is the derivative of the objective function f

w.r.t. Θ_k at the k^{th} iteration. The cost of storing the approximate Hessian matrix B_k ($N^2 \times N^2$) is prohibitive here because the dimension N^2 of the parameter set Θ is high ($\approx 10^4$). Therefore, the optimization uses L-BFGS in which the search direction p_k is calculated based on the current gradient ∇f_k and the curvature information from m most recent iterations $\{s_i = \Theta_{i+1} - \Theta_i, y_i = \nabla f_{i+1} - \nabla f_i | i = k-m, \dots, k-1\}$. Algorithm 2 presents the calculation on L-BFGS search direction p_k . In the implementation, m is set to 5.

Algorithm 2 Calculation on L-BFGS search direction p_k

- 1: **Input:** the derivative ∇f_k of the objective function f w.r.t. Θ_k , the curvature information from m most recent iterations $\{s_i, y_i | i = k-m, \dots, k-1\}$.
 - 2: $p_k = -\nabla f_k$;
 - 3: **for** $i = k-1, k-2, \dots, k-m$ **do**
 - 4: $\alpha_i = \frac{s_i^T p_k}{y_i^T s_k}$;
 - 5: $p_k = p_k - \alpha_i y_i$;
 - 6: **end for**
 - 7: $p_k = B_0^{-1} p_k$
 - 8: **for** $i = k-m, k-m+1, \dots, k-1$ **do**
 - 9: $\beta = \frac{y_i^T p_k}{y_i^T s_i}$;
 - 10: $p_k = p_k + s_i(\alpha_i - \beta)$;
 - 11: **end for**
 - 12: **Output:** L-BFGS search direction p_k
-

Given the search direction p_k obtained from Algorithm 2, there is $\Theta_{k+1} = \Theta_k + \alpha_k p_k$, where α_k is chosen to satisfy the Wolfe conditions [135]. When $k > m$, the curvature information $\{s_{k-m}, y_{k-m}\}$ is discarded and the new one $\{s_k = \Theta_{k+1} - \Theta_k, y_k = \nabla f_{k+1} - \nabla f_k\}$ is computed and saved. Using L-BFGS to optimize the adaptation formulation, the convergence can be reached after several iterations. Similarly, the objective function in Equation 3.3 for the pre-learning part can be solved by using the same optimization method as it is a short version of Equation 3.4.

To capture the appearance changes of the object, the parameter set Θ of the adaptation module is learned online every M frames. The L-BFGS algorithm is

also utilized to solve the minimization problem $\arg \min_{\Theta} f(\Theta; \mathbf{X}, \tilde{\Theta})$, where $\mathbf{X} = \{\mathbf{x}_1 : \mathbf{x}_M\}$ denotes training data within object regions from M most recent frames and $\tilde{\Theta}$ indicates the old parameter set. The learned parameter set Θ converges quickly in the current group of M frames and it will be used as the old parameter set $\tilde{\Theta}$ in the next group of M frames. In our implementation, M is set to 20 in all test video sequences.

3.3.6 Implementation Details

Auxiliary Data. The generic features are pre-learned by using the auxiliary data from Hans van Hateren natural scene videos [3]. As mentioned in [2], the generic features learned from sequences containing tracked objects can encode more useful information such as non-linear warping. Hence, the video sequences containing tracked objects are employed for pre-learning features (see Figure 3.1).

Initialization. The tracking results in the first 20 frames from ASLA [4] are used as the initial training data for the adaptation module. It is fair to compare with other methods under this setting. Many tracking algorithms have this kind of initializations. For example, Jia et al. [4] utilize the k-d tree matching scheme to track the object in the first 10 frames and then build the exemplar library and the patch dictionary based on these tracking results.

Computational Cost Learning generic features consumes much time (about 20 minutes) due to the large data set. However, it is conducted offline, hence the time for this part is not important. For the online adaptation, the transformation matrix W to be learned is initialized with the pre-learned W_{old} . Based on the training data collected online, each update of the adaptation module takes only several iterations to achieve the convergence. Another part is feature mapping, which requires that features are extracted from candidate image patches. ASLA [4] requires to sample 600 candidate patches in each frame. It is very expensive

if the feature mapping is carried out for all candidate patches. Therefore, the proposed algorithm uses a coarse-to-fine searching strategy, in which a number of (e.g. 20) promising candidates in each frame are first selected according to tracking results from ASLA [4] using raw pixel values and these candidates is then ranked by using learned hierarchical features. The experiments including both offline training and online tracking are run on a PC with a Quad-Core 3.30 GHz CPU and 8 GB RAM. However, the multi-core setting of the PC is not used. The speed of the proposed tracker (about 0.8 fps) is roughly twice slower than the one of ASLA [4] (about 1.6 fps) due to the additional feature extraction step. The time (about 625 ms) spent on the feature extraction is about same as on the other parts of the proposed tracker. Note that the main objective here is to show that using the proposed feature learning algorithm can improve the tracking accuracy. The efficiency of the proposed tracker could be improved further because the feature mappings for different patches could be conducted in parallel by more advanced techniques, e.g. GPU. Finally, the trade-off parameters of λ and γ in Equation 3.4 are empirically tuned for different sequences. However, the parameters change in a small range. λ and γ are tuned in $[1, 10]$ and $[90, 110]$ respectively.

3.4 Experiments

First, learned hierarchical features are evaluated to demonstrate their robustness to complicated motion transformations. Second, the temporal slowness constraint and the adaptation module of the proposed feature learning algorithm are evaluated respectively. Third, the proposed tracker’s capability of handling typical problems in visual tracking is evaluated. Then, the proposed tracker is compared with 14 state-of-the-art trackers. Moreover, the comparison results between DLT [23] and the proposed tracker are presented. Next, two failure cases of the proposed

tracker due to heavy occlusions are shown. Furthermore, the generalizability of the proposed feature learning algorithm is demonstrated on the other two tracking algorithms. Finally, the evaluation on the benchmark dataset [8] is presented.

For the evaluations from Section 3.4.1 to Section 3.4.7, two measurements are used to quantitatively evaluate tracking performances. The first one is called center location error which measures the distances of centers between tracking results and ground truths in pixels. The second one is called overlap rate which is calculated according to $\frac{area(R_T \cap R_G)}{area(R_T \cup R_G)}$ and indicates extent of region overlapping between tracking results R_T and ground truths R_G . It is necessary to mention that there is often subjective bias in evaluating tracking algorithms as indicated in [136].

For the evaluation on the benchmark [8] in Section 3.4.8, two measurements are used: i) Precision vs. Location error threshold, the percentage of the frames in which distances between tracking results and ground truth are below certain thresholds. ii) Success rate vs. Overlap threshold, the percentage of the frames in which overlapping percentages of tracking results against ground truth are higher than certain thresholds. Different trackers are ranked according to location error thresholding at 20 pixels for precision and Area Under Curve (AUC) for success rate. In addition, the one-pass evaluation (OPE) setting in the benchmark is used to generate experimental results.

3.4.1 Evaluation on Learned Hierarchical Features' Robustness to Complicated Motion Transformations

Both quantitative and qualitative results on 15 challenging sequences are presented, in which target objects have complicated motion transformations. e.g. in-plane rotation, out-of-plane rotation and non-rigid object deformation. To demonstrate the learned features' robustness to complex motion patterns, the proposed tracker

Table 3.1: Average center error (in pixels). The best two results are shown in red and blue fonts. The proposed tracker using the learned features is compared with 4 state-of-the-art trackers using the other feature representations: the raw pixel values (ASLA[4]_RAW), the hand-crafted HOG feature (ASLA[4]_HOG), the sparse representation (ℓ_1 -APG[5]) and the data-independent feature (CT_DIF[6]). The results are also presented for the variant of the proposed tracker (Ours_VAR) which does not use the temporal slowness constraint in the feature learning.

Sequence	ASLA[4]_RAW	ASLA[4]_HOG	ℓ_1 -APG[5]	CT_DIF[6]	Ours_VAR	Ours
Basketball	70.2	245.6	107.2	123.6	14.0	11.2
Biker	88.0	68.2	79.6	80.7	19.1	11.7
David2	24.0	25.1	44.1	59.8	12.4	1.5
FleetFace	129.2	180.0	123.5	145.5	30.2	24.8
Freeman1	79.2	69.8	21.7	17.8	9.2	8.7
Freeman3	21.4	43.4	15.6	65.6	6.5	4.4
Kitesurf	40.3	38.5	71.7	62.1	22.6	13.2
Lemming	165.5	155.4	138.2	149.3	8.6	6.7
MountainBike	185.9	155.2	210.1	212.9	10.3	9.2
Shaking	86.5	86.6	113.0	30.9	37.9	15.2
Skating1	14.6	15.6	72.3	87.9	6.6	6.5
Sylvester	27.2	27.7	31.6	17.6	52.1	6.4
Tiger1	71.2	112.9	61.7	85.6	70.3	40.9
Tiger2	61.8	96.2	58.4	83.3	27.1	19.2
Trellis	31.9	18.8	62.5	47.4	13.2	3.0
Average	73.1	89.3	80.7	84.7	22.7	12.2

is compared with the other 4 state-of-the-art trackers using different feature representations such as the raw pixel value (ASLA[4]_RAW), the hand-crafted feature of Histogram of Oriented Gradients (HOG) [1] (ASLA[4]_HOG), the sparse representation (ℓ_1 -APG[5]) and the data-independent feature (CT_DIF[6]). It is necessary to mention that ASLA_HOG and the proposed tracker use the same tracking framework as in ASLA_RAW. The difference is that ASLA_HOG and the proposed tracker integrate the HOG feature and the proposed learned hierarchical features into the tracking framework respectively. However, the other 2 trackers, ℓ_1 -APG and CT_DIF, use their own tracking frameworks which are different from ASLA_RAW. The hand-crafted HOG feature and the sparse feature are employed here because of their superior performances in object detection and recognition. Additionally, the data-independent feature is used here because it also aims to solve the problem of insufficient training data in object tracking.

In this evaluation, 13 sequences used in [137] are used for test. Also, there

Table 3.2: Average overlap rate (%). The best two results are shown in red and blue fonts. The proposed tracker using the learned features is compared with 4 state-of-the-art trackers using the other feature representations: the raw pixel values (ASLA[4]_RAW), the hand-crafted HOG feature (ASLA[4]_HOG), the sparse representation (ℓ_1 _APG[5]) and the data-independent feature (CT_DIF[6]). The results are also presented for the variant of the proposed tracker (Ours_VAR) which does not use the temporal slowness constraint in the feature learning.

Sequence	ASLA[4]_RAW	ASLA[4]_HOG	ℓ_1 _APG[5]	CT_DIF[6]	Ours_VAR	Ours
Basketball	0.46	0.27	0.29	0.36	0.54	0.59
Biker	0.52	0.41	0.29	0.40	0.51	0.67
David2	0.59	0.26	0.29	0.05	0.61	0.85
FleetFace	0.13	0.13	0.41	0.24	0.58	0.60
Freeman1	0.32	0.28	0.34	0.27	0.45	0.51
Freeman3	0.47	0.52	0.45	0.13	0.49	0.59
Kitesurf	0.35	0.36	0.34	0.24	0.50	0.63
Lemming	0.42	0.41	0.40	0.41	0.69	0.76
MountainBike	0.27	0.37	0.40	0.25	0.61	0.68
Shaking	0.10	0.06	0.06	0.33	0.50	0.58
Skating1	0.38	0.35	0.31	0.26	0.35	0.42
Sylvester	0.35	0.47	0.37	0.55	0.39	0.71
Tiger1	0.25	0.24	0.42	0.15	0.27	0.50
Tiger2	0.32	0.21	0.38	0.10	0.44	0.53
Trellis	0.49	0.49	0.28	0.28	0.69	0.79
Average	0.36	0.32	0.34	0.27	0.51	0.63

are additional 2 special sequences of “biker” and “kitesurf”, in which the original video sequences are used, but new target objects are defined for tracking. These 2 sequences are challenging because the newly defined objects contain complex motion patterns. For example, in the “biker” sequence (see Figure 3.3), the biker’s whole body with non-rigid object deformation needs to be tracked. Tables 3.1 and 3.2 present quantitative results which demonstrate that the learned features outperform the other state-of-the-art feature representations in terms of handling complicated motion transformations effectively. Figures 3.3, 3.4 and 3.5 show the qualitative results on sequences with non-rigid object deformation, in-plane rotations and out-of-plane rotations respectively. The qualitative results are explained as follows.

Non-rigid object deformation The sequences (Basketball, Biker, FleetFace, Kitesurf and Skating1) shown in Figure 3.3 are challenging because the target objects have non-rigid object deformations. For example, the basketball player in Figure 3.3

Figure 3.3: Qualitative results on sequences with non-rigid object deformation. The purple, green, cyan, blue and red bounding boxes refer to ASLA[4]_RAW, ASLA[4]_HOG, ℓ_1 -APG[5], CT.DIF[6] and the proposed tracker respectively. This figure is better viewed in color.

(a) has deformable changes due to his running and defending actions. The biker in Figure 3.3 (b) has dramatic body deformations during his acrobatic actions. The man in Figure 3.3 (c) has significant facial changes due to his laughing expression. The person in Figure 3.3 (d) has deformable pose changes because of his surfing actions. The girl in Figure 3.3 (e) has articulated deformations caused by her arm waving and body spinning. It can be observed that the 4 baseline trackers (ASLA[4]_RAW, ASLA[4]_HOG, ℓ_1 -APG [5] and CT.DIF [6]) fail to track the target objects in these challenging sequences. In contrast, the proposed tracker succeeds in capturing the target objects because the learned features are invariant to non-rigid object deformation.

In-plane rotations The target objects in the sequences (David2, MountainBike,

Figure 3.4: Qualitative results on sequences with in-plane rotations. The purple, green, cyan, blue and red bounding boxes refer to ASLA[4]_RAW, ASLA[4]_HOG, ℓ_1 -APG[5], CT-DIF[6] and the proposed tracker respectively. This figure is better viewed in color.

Sylvester, Tiger1 and Tiger2) have significant in-plane rotations which are difficult for trackers to capture. In Figure 3.4 (a), the man’s face not only has translations but also in-plane rotations which occur when the face is slanted. In Figure 3.4 (b), the mountain bike has the in-plane rotations due to its acrobatic actions in the sky. In Figure 3.4 (c), the front of the toy rotates a lot in the plane. In Figure 3.4 (d) and (e), the toy tiger also has a lot of in-plane rotations. It can be observed that all the baseline trackers have drifted away from the target objects in these sequences because of this in-plane rotation, whereas the proposed tracker can handle this kind of motion transformation very well by using the learned features.

Out-of-plane rotations The sequences (Freeman1, Freeman3, Lemming, Shak-

Figure 3.5: Qualitative results on sequences with out-of-plane rotations. The purple, green, cyan, blue and red bounding boxes refer to ASLA[4]_RAW, ASLA[4]_HOG, ℓ_1 -APG[5], CT-DIF[6] and the proposed tracker respectively. This figure is better viewed in color.

ing and Trellis) are difficult because the target objects have out-of-plane rotations which change the object appearance significantly and hence yield a tracking failure. For instance, in Figure 3.5 (a), (b) and (e), the men’s faces have significant out-of-plane rotations because the poses of their heads change a lot during walking. The toy in Figure 3.5 (c) has the out-of-plane rotations because its front does not always face to the camera. The singer’s head shown in Figure 3.5 (d) has the out-of-plane rotations because it shakes up and down. It can be observed that the proposed tracker can successfully capture the target objects through these sequences. This success can be ascribed to the learned features’ robustness to the out-of-plane rotations. In contrast, the baseline trackers cannot handle this complicated motion transformations because their feature representations are not designed to capture the motion invariance.

3.4.2 Evaluation on Effectivenesses of Temporal Slowness Constraint and Feature Adaptation Module

First, the results are presented in Table 3.1 and Table 3.2 for the variant of the proposed tracker (Ours_VAR) which does not use the temporal slowness constraint for learning the generic features. It can be observed that the proposed tracker using the constraint has a better performance on 15 challenging video sequences. It demonstrates that the temporal slowness constraint is beneficial for learning the generic features robust to complicated motion transformations. Then, the adaptation module in the feature learning algorithm is evaluated on 8 video sequences reported in ASLA [4]. Table 3.3 and Table 3.4 respectively present the average center location errors and the average overlap rates of the proposed tracker with (Ours_adp) and without (Ours_noadp) the adaptation module. From the quantitative comparison, it can be found that the adaptation module enhances the performance of the proposed tracker. It is due to the fact that the adaptation

Table 3.3: Average center error (in pixels). The best two results are shown in red and blue fonts. The proposed tracker’s performances with (Ours_adp) and without (Ours_noadp) the adaptation module are presented respectively. Also, the proposed tracker is compared with 4 baseline trackers using the other features such as the raw pixel values (ASLA[4]_RAW), the hand-crafted HOG feature (ASLA[4]_HOG), the sparse feature (ℓ_1 -APG [5]) and the data-independent feature (CT-DIF[6]).

Sequence	ASLA[4]_RAW	ASLA[4]_HOG	ℓ_1 -APG[5]	CT-DIF[6]	Ours_noadp	Ours_adp
Board	7.3	15.3	259.4	80.3	7.1	8.1
Car11	2.0	2.7	22.2	78.0	1.9	1.4
Caviar	2.3	66.8	95.6	65.5	2.2	2.2
David	3.6	45.8	138.2	12.8	3.5	3.2
Faceocc2	3.8	32.4	17.7	12.8	3.7	3.1
Singer1	4.8	5.0	167.9	13.7	4.5	4.0
Stone	1.8	2.8	136.8	32.4	2.1	1.5
Woman	2.8	140.6	176.1	110.2	2.6	2.2

Table 3.4: Average overlap rates. (%) The best two results are shown in red and blue fonts. The proposed tracker’s performances with (Ours_adp) and without (Ours_noadp) the adaptation module are presented respectively. Also, the proposed tracker is compared with 4 baseline trackers using the other features such as the raw pixel values (ASLA[4]_RAW), the hand-crafted HOG feature (ASLA[4]_HOG), the sparse feature (ℓ_1 -APG [5]) and the data-independent feature (CT-DIF[6]).

Sequence	ASLA[4]_RAW	ASLA[4]_HOG	ℓ_1 -APG[5]	CT-DIF[6]	Ours_noadp	Ours_adp
Board	0.74	0.72	0.12	0.33	0.73	0.83
Car11	0.81	0.71	0.34	0.23	0.80	0.85
Caviar	0.84	0.40	0.06	0.33	0.87	0.88
David	0.79	0.27	0.19	0.56	0.80	0.81
Faceocc2	0.82	0.52	0.61	0.68	0.87	0.88
Singer1	0.81	0.79	0.14	0.34	0.82	0.85
Stone	0.56	0.57	0.16	0.33	0.59	0.60
Woman	0.78	0.35	0.20	0.41	0.81	0.84

module not only includes the appearance information of the specific target object, but also preserves the pre-learned features’ robustness to complicated motion transformations.

3.4.3 Evaluation on Capability of Handling Typical Problems in Visual Object Tracking

The 8 sequences in ASLA [4] are also used to evaluate the capability of handling typical problems in visual object tracking, e.g. illumination change, occlusion and cluttered background. First, the proposed tracker is quantitatively

compared with 4 baseline trackers, ASLA[4]_RAW, ASLA[4]_HOG, ℓ_1 -APG [5] and CT_DIF[6], in which the raw pixel values, the hand-crafted HOG feature, the sparse representation and the data-independent feature are used respectively. From Table 3.3 and Table 3.4, it can be found that the learned features are more competitive than the other 4 feature representations for handling typical issues in object tracking. Then, the qualitative results are presented in Figure 3.6. Since ASLA[4]_RAW has achieved promising results, the proposed tracker is not compared with ASLA[4]_RAW in the qualitative results. Next, the qualitative results are explained as follows.

Illumination change In the sequences of “car11”, “David” and “Singer”, target objects suffer the illumination change. ASLA_HOG misses the face early in “David” and cannot accurately track the rear of the car at the end of “Car11”. ℓ_1 -APG fails to track the target objects in these three sequences. CT_DIF quickly misses the target car at the beginning of “Car11” and also cannot adapt to the appearance changes caused by changing illumination in “David” and “Singer1”. Although the illumination changes are significant in three sequences, the proposed tracker can always track the objects with a high degree of accuracy.

Occlusion In the sequences of “Faceocc2”, “Caviar” and “Woman”, target objects are partially or fully occluded. ASLA_HOG has drifted away from the target face after it is partially occluded by a book. It totally misses the woman as the partial occlusion is severe. ASLA_HOG also fails to catch up the man in “Caviar” after he is fully occluded by another man. ℓ_1 -APG had already lost the man in “Caviar” and the woman before the occlusions. CT_DIF cannot accurately track the face in “Faceocc2” and misses persons when occlusions occur in “Caviar” and “Woman”. Although our tracker is not designed to handle occlusions, it can still track the objects effectively in these sequences because occlusion here is not heavy. If full occlusion lasts for a long time, the proposed tracker could fail as shown in

Figure 3.6: Qualitative results on sequences with typical issues such as illumination change, occlusion and cluttered background. The green, cyan, blue and red bounding boxes refer to ASLA[4]_HOG, ℓ_1 -APG [5], CT_DIF [6] and the proposed tracker respectively. This figure is better viewed in color.

Table 3.5: Average center error (in pixels). The best two results are shown in red and blue fonts. The proposed tracker is compared with FragT [14], IVT [7], ℓ_1 T [15], MIL [16], TLD [17], VTD [17], LSK [18], CT [6], ASLA [4], ℓ_1 APG [5], MTT [19], SCM [20], OSPT [21] and LSST [22]. The proposed tracker outperforms the state-of-the-art tracking algorithms.

Sequence	FragT[14]	IVT[7]	ℓ_1 T[15]	MIL[16]	TLD[17]	VTD[17]	LSK[18]	CT[6]	ASLA[4]	ℓ_1 APG[5]	MTT[19]	SCM[20]	OSPT[21]	LSST[22]	Ours
Car4	179.8	2.9	9.0	60.1	18.8	12.3	3.3	229.7	4.3	16.4	37.2	3.5	3.0	2.9	3.0
Car11	63.9	2.1	33.3	43.5	25.1	27.1	4.1	78.0	2.0	1.7	1.8	1.8	2.2	1.6	1.4
Caviar	94.2	66.2	65.9	83.9	53.0	60.9	55.3	65.5	2.3	68.6	67.5	2.2	45.7	3.1	2.2
David	76.7	3.6	7.6	16.1	9.7	13.6	6.3	12.8	3.6	10.8	13.4	3.4	3.2	4.3	3.2
Deer	50.4	127.5	140.5	55.6	25.7	11.9	69.8	10.5	8.0	38.4	9.2	36.8	8.5	10.0	5.5
Faceocc1	4.6	16.3	6.3	19.2	17.6	11.1	5.3	19.9	10.8	6.8	14.1	3.2	4.7	5.3	3.9
Faceocc2	15.5	10.2	11.1	14.1	18.6	10.4	58.6	12.8	3.8	6.3	9.2	4.8	4.0	3.1	3.1
Football	16.9	42.5	48.6	6.6	11.8	4.1	14.1	11.6	18.0	12.4	6.5	10.4	33.7	7.6	6.5
Jumping	58.4	36.8	12.5	9.9	3.6	63.0	55.2	53.0	39.1	8.8	19.2	3.9	5.0	4.8	4.9
Singer1	22.0	8.5	4.6	15.2	32.7	4.1	14.5	13.7	4.8	3.1	41.2	3.8	4.7	3.5	4.0
Average	58.2	31.7	33.9	32.4	21.7	21.9	28.7	50.8	9.7	17.3	21.9	7.4	11.5	4.6	3.8

Figure 3.7.

Cluttered background In the sequences “Board” and “Stone”, the target objects move in cluttered backgrounds. Although ASLA_HOG can track the stone at the end of “Stone”, it actually drifts to the background as soon as the 96th frame. ℓ_1 -APG quickly misses the target objects in both sequences. CT_DIF is distracted by another bigger stone which is very similar to the target one in “Stone”. The proposed tracker can always track the objects in both sequences.

3.4.4 Comparison with the State-of-the-art Trackers

The proposed tracker is compared against 14 state-of-the-art algorithms on 10 video sequences used in previous works [7][16][17][138][139]. Table 3.5 and Table 3.6 respectively show the average center location errors and the average overlap rates of different tracking algorithms. The proposed tracker outperforms the other state-of-the-art tracking algorithms in most cases and especially improves the baseline ASLA [4]. This success is due to the learned hierarchical features.

Table 3.6: Average overlap rate (%). The best two results are shown in red and blue fonts. The proposed tracker is compared with FragT [14], IVT [7], ℓ_1 T [15], MIL [16], TLD [17], VTD [17], LSK [18], CT [6], ASLA [4], ℓ_1 APG [5], MTT [19], SCM [20], OSPT [21] and LSST [22]. The proposed tracker outperforms the state-of-the-art tracking algorithms.

Sequence	FragT[14]	IVT[7]	ℓ_1 T[15]	MIL[16]	TLD[17]	VTD[17]	LSK[18]	CT[6]	ASLA[4]	ℓ_1 APG[5]	MTT[19]	SCM[20]	OSPT[21]	LSST[22]	Ours
Car4	0.22	0.92	0.78	0.34	0.64	0.73	0.91	0.28	0.89	0.70	0.53	0.89	0.92	0.92	0.91
Car11	0.09	0.81	0.44	0.17	0.38	0.43	0.49	0.23	0.81	0.83	0.58	0.79	0.81	0.84	0.85
Caviar	0.19	0.21	0.20	0.19	0.21	0.19	0.58	0.33	0.84	0.13	0.14	0.87	0.25	0.85	0.88
David	0.19	0.72	0.63	0.45	0.60	0.53	0.72	0.56	0.79	0.63	0.53	0.75	0.76	0.75	0.81
Deer	0.08	0.22	0.07	0.21	0.41	0.58	0.35	0.60	0.62	0.45	0.60	0.46	0.61	0.58	0.68
Faceoc1	0.90	0.85	0.89	0.59	0.65	0.77	0.90	0.74	0.83	0.87	0.79	0.93	0.91	0.89	0.91
Faceocc2	0.60	0.59	0.67	0.61	0.49	0.59	0.33	0.68	0.82	0.70	0.72	0.82	0.84	0.86	0.88
Football	0.57	0.55	0.11	0.55	0.56	0.81	0.63	0.46	0.57	0.68	0.71	0.69	0.62	0.69	0.72
Jumping	0.14	0.28	0.55	0.53	0.69	0.08	0.09	0.07	0.24	0.59	0.30	0.73	0.69	0.65	0.70
Singer1	0.34	0.66	0.70	0.33	0.41	0.79	0.52	0.34	0.81	0.83	0.32	0.85	0.82	0.80	0.85
Average	0.33	0.58	0.50	0.40	0.50	0.55	0.55	0.45	0.72	0.64	0.52	0.78	0.72	0.78	0.82

Table 3.7: Comparison between DLT [23] and the proposed tracker on 8 sequences. The better results are shown in red fonts. “BetterCount” means the number of sequences on which the performance of the current tracker is better than the other one.

	Car4	Car11	David	Deer	Shaking	Singer1	Trellis	Woman	BetterCount
DLT	6.0	1.2	7.1	10.2	11.5	3.3	3.3	9.4	3
Ours	3.0	1.4	3.2	5.5	15.2	4.0	3.0	2.2	5

3.4.5 Comparison with DLT

The comparison results in terms of average center error (in pixels) between DLT [23] and the proposed tracker are presented in Table 3.7. It can be observed that the proposed tracker outperforms DLT on 5 of 8 sequences.

3.4.6 Failure Cases

One limitation of the proposed feature learning algorithm is its weak ability to handle heavy occlusions. It is because the training data which has been used does not include any occlusion samples, and it is hard to learn features which are invariant to occlusion because there are many different types of occlusions. Some tracking algorithms, e.g. ASLA [4], use the local appearance model to handle partial occlusions. However, full occlusion lasting for a long time is still a challenging

Figure 3.7: Failure cases due to heavy occlusions. The purple, green, cyan, blue and red bounding boxes refer to ASLA[4]_RAW, ASLA[4]_HOG, ℓ_1 -APG [5], CT-DIF [6] and the proposed tracker respectively. This figure is better viewed in color.

issue in object tracking. Figure 3.7 shows the qualitative results on two failure sequences.

Soccer The sequence aims to track the face of a soccer player, which suffers heavy occlusion. Both the baseline trackers and the proposed one fail to track the object through the sequence.

Surfer The sequence depicts a surfing guy who is heavily occluded by a wave in about 50 frames. Both the baseline trackers and the proposed one drift to the background after the target is fully and continuously occluded.

3.4.7 Evaluation on Learned Features' Generalizability

To demonstrate the generalizability of the learned features, the proposed feature learning algorithm is integrated into another baseline tracker which is called the incremental learning tracker (IVT) [7]. The performances on both the original IVT and the integrated tracker (deepIVT) in terms of the average center errors and the the average overlap rates are presented in Figure 3.8. It can be observed that the integrated tracker (deepIVT) outperforms the original IVT in most of 12 test sequences. Due to IVT's limited performance, the integrated tracker also misses objects in some sequences. However, the figures presented here aim to show that the learned features can boost the performance of the baseline tracker. In

Figure 3.8: Average center error (in pixels) and average overlap rate (%). The performances between the original IVT [7] and the integrated IVT (deepIVT) using the learned features are compared.

addition, the learned features’ generalizability is verified by using ℓ_1 -APG [5] as the baseline tracker to evaluate the performance on the same 12 sequences as used for IVT. As shown in Table 3.1 and Table 3.2, the original ℓ_1 -APG can hardly handle these challenging sequences with complicated motion transformations. In contrast, integrating the learned features into ℓ_1 -APG can succeed in tracking objects in 6 (David2, FleetFace, Freeman1, Freeman3, MountainBike and Sylvester) of 12 sequences. It can be concluded that the learned features are not only beneficial to ASLA [4], but also generally helpful to the other trackers.

3.4.8 Evaluation on A Larger Benchmark

Tracking performance is further evaluated on a recent public benchmark [8] containing 50 sequences which covers almost all challenging issues such as illumination changes, pose variations, occlusion, in/out-of-plane motions and cluttered background. The benchmark dataset reports the results from 29 trackers. Here, the proposed tracker using Learned Hierarchical Features (LHF) is compared with the top 6 trackers in the benchmark: ASLA [4], CXT [140], SCM [141], Struck [12], TLD [74] and VTS [142]. In addition, the proposed tracker is also compared with 3 recent state-of-the-art trackers: KCF [11], MEEM [9] and TGPR [10], which have reported their results on the benchmark.

Figure 3.9: Precision and success plots of the tracking results from the top 10 trackers on all sequences of the benchmark [8].

The precision plot and the success plot of tracking results from the top 10 trackers on all sequences of the benchmark [8] are presented in Figure 3.9. It can be found that the proposed tracker outperforms the baseline ASLA [4] in terms of both precision and success rate with large margins. This significant improvement can be owed to learned hierarchical features robust to both complicated motion transformations and appearance changes of specific target objects. Also, it can be observed that the proposed tracker can achieve comparable performance against the state-of-the-art trackers.

3.5 Concluding Remarks

In this chapter, a feature learning algorithm is proposed for visual object tracking. First, hierarchical features are learned from auxiliary video data by using a two-layer neural network with the temporal slowness constraint. Then, a domain adaptation module is proposed to adapt the pre-learned features according to specific target objects. As a result, learned hierarchical features are robust to both complicated motion transformations and appearance changes of specific tar-

get objects. Experimental results demonstrate that the proposed feature learning algorithm can significantly improve visual object tracking performance.

Chapter 4

Multiple Object Tracking Using Learned Hierarchical Features with A Joint Learning Strategy

In this chapter, a feature learning algorithm is developed for multiple object tracking by using a joint learning strategy. First, generic features are learned from auxiliary video data. Then, a domain adaptation module is proposed to jointly conduct the feature adaptation according to multiple target objects by using a joint learning strategy which can simultaneously learn common features shared by all target objects and individual features for each object. As a result, the feature adaptation module can achieve better performance because it makes use of more training data. Experimental results demonstrate that the proposed feature learning algorithm with the joint learning strategy can significantly improve multiple object tracking performance.

4.1 Introduction

Multiple object tracking aims to find the locations of multiple target objects in an image sequence given their initial locations in the first frame. In fact, it is an extension of single object tracking, most of which could be naturally extended to the cases of multiple object tracking. For example, multiple target objects can be tracked by simply running multiple instances of a single object tracker on all target objects. In this work, learning hierarchical features for multiple object tracking is investigated.

Given multiple target objects, it is straightforward to apply the feature learning algorithm developed in the first work independently for each target object. In this way, the adaptation module for each target object will only use its own annotation data, which is actually still limited for feature adaptation. To solve this problem, a joint learning strategy using annotation data of all target objects is proposed to conduct feature adaptation jointly according to multiple target objects. As a result, the feature adaptation module can achieve better performance since it makes use of more training data.

The overview of the proposed feature learning algorithm is shown in Figure 4.1. First, the generic features are learned from auxiliary video data under the temporal slowness constraint [2]. Then, a domain adaptation module using all target annotation data together is proposed to jointly conduct feature adaptation by using a joint learning strategy, which can simultaneously learn common features shared by all target objects and individual features for each object. The proposed feature learning algorithm is similar to but intrinsically different from a recent work [77]. First, the focus of the work [77] is to learn online regression weights of multiple object classifiers, whereas the proposed method aims to learn hierarchical features for multiple target objects. Second, $\ell_{2,1}$ norm is used to relate linear regression vectors of multiple object classifiers in the work [77]. However, the transformation

Figure 4.1: Overview of the proposed feature learning algorithm. First, the generic feature W_p is pre-learned from K auxiliary video sequences. Then, the individual feature $W_{i,i=1,\dots,M}$ for each target object and the common feature W_c shared by M target objects are simultaneously learned in the feature adaptation module using a joint learning strategy. Each row of a transformation matrix W can be visualized as a filter patch as shown in the figure. There are 25 filter patches which are visualized for the corresponding W .

matrices of the deep neural networks in the proposed feature learning algorithm cannot be related by simply using $\ell_{2,1}$ norm. To solve this problem, inspired by another work [143], two regularization terms are proposed for the feature adaptation module.

The main contribution of this work is to learn hierarchical features for multiple object tracking by using a joint learning strategy based on all target annotation data. Learned hierarchical features have the capability of discriminating target objects from each other. The proposed feature learning algorithm is general and could be integrated into any multiple object tracking system. In this chapter, the proposed algorithm is integrated into the baseline tracker no-SPOT [78] which is a variant of the SPOT tracker and does not use the spatial constraints between multiple target objects. Experimental results demonstrate that the proposed feature learning algorithm using the joint learning strategy can significantly improve multiple object tracking performance.

Figure 4.2: Illustration of the basic learning module and the two-layer architecture. The input to the first layer is raw pixel values of local patches (16×16), which are obtained by convolutions in the holistic image region (32×32). The input to the second layer is the output of the first layer whitened by PCA. The output of the first layer is the concatenated vector of mapping features from all local patches (16×16) in the holistic image (32×32).

4.2 Feature Pre-learning from Auxiliary Data

This step is essentially the same as the one presented in the previous chapter. It is included here to make the chapter self-contained and consistent in notations. The generic features are pre-learned from Hans van Hateren natural scene videos [3] by using a two-layer neural network, each of which comprises a linear transformation and a pooling step. The basic learning modules and the two-layer architecture are illustrated in Figure 4.2.

Suppose there is a training video sequence with T frame images. The input data to the basic learning module is denoted as $\mathbf{x} = [x_1, x_2, \dots, x_T]$. The mapping feature of x_t through the learning module is denoted by $z_t = \sqrt{H(W_p x_t)^2}$, where H is fixed for subspace pooling. The feature transformation matrix W_p is pre-learned by solving the following optimization with the temporal slowness constraint [2].

$$\min_W \lambda \sum_{t=1}^{T-1} \|z_t - z_{t+1}\|_1 + \sum_{t=1}^T \|x_t - W_p^T W_p x_t\|_2^2 \quad (4.1)$$

The input to the first layer is the raw pixel values of local patches (16×16), which are obtained by convolutions in the holistic image region (32×32). Then, the mapping features of all local patches in an image (32×32) are concatenated and whitened by PCA to form the input to the second layer. The two-layer architecture can pre-learn the generic features which are invariant to diverse motion transformations. However, the pre-learned features do not include the appearance information of the target objects. In the following section, the feature adaptation module will be discussed.

4.3 Feature Adaptation Using the Proposed Joint Learning Strategy

It is straightforward to adapt the pre-learned features according to multiple target objects independently. In this way, the feature adaptation module for each target object only uses its own annotation data. However, the training data in visual object tracking is actually still very limited for feature adaptation. To solve this problem, a joint learning strategy using all target annotation data together is proposed to conduct feature adaptation jointly according to multiple target objects.

The proposed joint learning strategy can simultaneously learn common features shared by all target objects and individual features for each object.

Suppose there are M target objects. $\mathbf{x}_i = [x_i^1, x_i^2, \dots, x_i^N]$ indicates the annotation data of the i^{th} target object. The corresponding feature mapping is denoted as $z_i^j = \sqrt{H(W_i x_i^j)^2}$, where W_i is the transformation matrix for the i^{th} target object. In recent the work [77], feature regression vectors from different object classifiers are formulated into $\ell_{2,1}$ norm to seek the feature dimensions shared by different objects. However, $\ell_{2,1}$ norm cannot be easily used to relate the transformation matrices of the deep neural networks in the proposed feature learning algorithm. Inspired by [143], the feature adaptation is formulated as follows.

$$\begin{aligned} \min_{W_c, W_i} \lambda \sum_{i=1}^M \sum_{j=1}^{N-1} \|z_i^j - z_i^{j+1}\|_1 &+ \sum_{i=1}^M \sum_{j=1}^N \|x_i^j - W_i^T W_i x_i^j\|_2^2 \\ &+ \beta \sum_{i=1}^M \sum_{k=1; k \neq i}^M \|W_i W_k^T\|_F^2 + \gamma \sum_{i=1}^M \|W_i W_c^T\|_F^2, \end{aligned} \quad (4.2)$$

where W_c and W_i refer to the common and the individual transformation matrices respectively. $\|\cdot\|_F$ is the Frobenius norm. Suppose $W \in \mathbb{R}^{d_{ou} \times d_{in}}$, where d_{in} and d_{ou} indicates the dimensionality of the input and the output. Each row of the transformation matrix is a filter of the input data as shown in Figure 4.1. These filters map the input data to the output feature space.

The third term in Equation 4.2 aims to make filters of different transformation matrices orthogonal to each other so that feature spaces of different target objects are separated from each other. This regularization term enhances discriminative capabilities of the individual features. The last term in Equation 4.2 intends to make the common feature distinct to all individual ones so that the individual features for each object could be more compact.

4.4 Optimization

To optimize Equation 4.2, it is proposed to alternately update the common transformation matrix W_c and the individual one W_i . For simplicity, only the derivations of the proposed two regularization terms are presented below.

First, W_c and $W_{k;k \neq i}$ are fixed to update W_i . The third term can be denoted as follows.

$$\Phi(W_i) = \sum_{k=1; k \neq i}^M \|W_i W_k^T\|_F^2 = \sum_{k=1; k \neq i}^M \text{tr}(W_i W_k^T W_k W_i^T). \quad (4.3)$$

Similarly, the last term can be denoted as follows.

$$\Psi(W_i) = \|W_i W_c^T\|_F^2 = \text{tr}(W_i W_c^T W_c W_i^T). \quad (4.4)$$

Suppose there is $f(W_i) = W_i W_k^T$. To obtain the first derivative of $\Phi(W_i)$ w.r.t. W_i , the derivation is presented as follows.

$$\begin{aligned} \frac{\partial \Phi(W_i)}{\partial W_i} &= \sum_{k=1; k \neq i}^M \frac{\partial \text{tr}(f(W_i) W_k W_i^T)}{\partial W_i} \\ &= \sum_{k=1; k \neq i}^M \frac{\partial \text{tr}(f(W_i) W_k W_i^T)}{\partial W_i} + \left(\frac{\partial \text{tr}(W_i^T f(W_i) W_k)}{\partial W_i^T} \right)^T \\ &= \sum_{k=1; k \neq i}^M (W_k W_i^T)^T f'(W_i) + ((f(W_i) W_k)^T)^T \\ &= 2 \sum_{k=1; k \neq i}^M W_i W_k^T W_k. \end{aligned} \quad (4.5)$$

Similarly, there is

$$\frac{\partial \Psi(W_i)}{\partial W_i} = 2 W_i W_c^T W_c. \quad (4.6)$$

Then, there is

$$\frac{\partial \Phi(W_i)}{\partial W_i} + \frac{\partial \Psi(W_i)}{\partial W_i} = 0. \quad (4.7)$$

Finally, there is

$$W_i(W_c^T W_c + \sum_{k=1; k \neq i}^M W_k^T W_k) = 0. \quad (4.8)$$

The solution of Equation 4.8 is as follows

$$W_i = \Gamma(\text{Null}((W_c^T W_c + \sum_{k=1; k \neq i}^M W_k^T W_k)^T))^T, \quad (4.9)$$

where $\Gamma(V)$ indicates a matrix of any linear combination of the vectors of V .

Subsequently, $W_{i=1, \dots, M}$ is fixed to update W_c . The last term in Equation 4.2 is denoted as follows.

$$\mathcal{R}(W_c) = \sum_{i=1}^M \|W_i W_c^T\|_F^2 = \sum_{i=1}^M \text{tr}(W_i W_c^T W_c W_i^T) \quad (4.10)$$

Similar to the derivation in Equation 4.5, there is

$$\frac{\partial \mathcal{R}(W_c)}{\partial W_c} = \sum_{i=1}^M \frac{\partial \text{tr}(W_i W_c^T W_c W_i^T)}{\partial W_c} = 2 \sum_{i=1}^M W_c W_i^T W_i. \quad (4.11)$$

Then, there is

$$\frac{\partial \mathcal{R}(W_c)}{\partial W_c} = 2W_c \sum_{i=1}^M W_i^T W_i = 0. \quad (4.12)$$

Finally, the solution to Equation 4.12 is as follows.

$$W_c = \Gamma(\text{Null}((\sum_{i=1}^M W_i^T W_i)^T))^T, \quad (4.13)$$

where $\Gamma(V)$ is a matrix of any linear combination of the vectors of V .

The proposed feature learning algorithm is summarized in Algorithm 3. Note that it is carried out in two layers. For simplicity, only the calculation of transformation matrices in one layer is described.

Algorithm 3 Learning Hierarchical Features for Multiple Object Tracking by Using the Proposed Joint Learning Strategy

- 1: **Input:** Auxiliary video data, a test video with M target objects.
 - 2: Use Equation 4.1 to pre-learn the generic transformation matrix W_p ;
 - 3: Initialize the common and the individual feature transformation matrix W_c and $W_{i,i=1,\dots,M}$ as the generic one W_p ;
 - 4: **while** Not Converge **do**
 - 5: **for** $i = 1, 2, \dots, M$ **do**
 - 6: Fix W_c and $W_{k;k \neq i}$ and use Equation 4.9 to update W_i ;
 - 7: **end for**
 - 8: Fix $W_{i,i=1,\dots,M}$ and use Equation 4.13 to update W_c ;
 - 9: **end while**
 - 10: **Output:** The pre-learned transformation matrix W_p , the common transformation matrix W_c and the individual transformation matrix $W_{i,i=1,\dots,M}$.
-

4.5 Experiments

The proposed feature learning algorithm is integrated into the no-SPOT tracker [78] which does not consider the spatial constraint between the target objects. In the experiment, the tracker using the proposed feature learning algorithm is compared with the baseline no-SPOT tracker and the state-of-the-art SPOT tracker [78], both of which use the HOG feature [1]. The proposed joint learning strategy in the feature adaptation module is evaluated by comparing two variants of the feature learning algorithm: Our_adpNaive and Our_adpJoint. Our_adpNaive

Table 4.1: Performances of different trackers on 9 videos. It is reported in terms of two measurements: average center errors (ACE, lower is better) and average success rates (ASR, higher is better). ACE is the average distance in pixels between centers of the tracked and the ground truth bounding boxes. ASR is the average percentage of frames which are labeled as success if the overlap between the tracked and the ground truth bounding boxes is larger than 50%. ACE is averaged over all target objects and over all frames. ASR is averaged over all target objects. The best two results are shown in red and blue fonts.

Sequence	no-SPOT[78] ACE / ASR	SPOT[78] ACE / ASR	Our_adpNaive ACE / ASR	Our_adpJoint ACE / ASR
Air Show	8.8 / 0.92	6.9 / 0.92	7.1 / 0.96	6.5 / 0.98
Basketball	63.3 / 0.67	12.7 / 0.85	15.7 / 0.84	9.2 / 0.89
Car Chase	24.8 / 0.78	9.2 / 0.89	14.7 / 0.80	9.5 / 0.87
Hunting	171.7 / 0.07	17.9 / 0.87	19.4 / 0.83	11.2 / 0.93
Parade	62.3 / 0.29	8.6 / 0.69	13.6 / 0.72	7.9 / 0.85
Red Flowers	50.6 / 0.38	8.2 / 0.99	15.5 / 0.86	10.4 / 0.92
Shaking	58.3 / 0.47	9.8 / 0.97	14.9 / 0.81	8.5 / 0.93
Skating	122.2 / 0.35	14.9 / 0.85	32.5 / 0.77	11.9 / 0.87
Sky Diving	51.4 / 0.48	6.7 / 0.98	18.4 / 0.89	7.1 / 0.97
Ave. Rank	4.0 / 3.9	1.7 / 1.8	3.0 / 2.8	1.3 / 1.4

independently adapts the pre-learned generic features according to multiple target objects, while Our_adpJoint conducts the feature adaptation module by using the proposed joint learning strategy.

Table 4.1 presents the quantitative results of different trackers on nine videos used in [78]. First, it can be observed that both of two variant trackers outperform the baseline no-SPOT tracker on all videos. It demonstrates that the proposed feature learning algorithm can significantly improve multiple object tracking performance. Then, it can be observed that the proposed joint learning strategy can further enhance the feature adaptation module. It validates that using more training data is beneficial for feature adaptation. Also, it can be observed that the proposed tracker using learned hierarchical features outperforms the SPOT tracker [78] in 5 out of 9 test videos. Also, the average rankings of the proposed tracker are higher than the ones of the SPOT tracker in terms of both measurements. The proposed feature learning algorithm improves the baseline tracker no-SPOT by

using learned hierarchical features, which is different from that of using spatial constraints between target objects as in the SPOT tracker. Table 4.1 shows that these two distinct cues lead to comparable performance. Combining these two cues may lead to further performance boost. As fusing the two cues requires careful consideration, it will be considered as future work.

The qualitative results of the proposed tracker are presented in Figure 4.3, from which it can be observed that the proposed tracker using learned hierarchical features can achieve very promising multiple object tracking performance.

4.6 Concluding Remarks

In this chapter, a feature learning algorithm is developed for multiple object tracking by using a joint learning strategy which is intended to make use of annotation data from all target objects. The framework of the feature learning algorithm follows the first work in this thesis but for multiple object tracking. To achieve better performance of feature adaptation, a joint learning strategy is proposed to simultaneously learn common feature shared by all target objects and individual features for each object. The proposed regularization terms expect that individual features are distinct to each other and common features are perpendicular to all individual ones. As a result, learned hierarchical features are more discriminative and compact. Experimental results demonstrate that the proposed feature learning algorithm can significantly enhance the multiple object tracking performance.

Figure 4.3: Qualitative results obtained from the proposed tracker on nine videos. The rectangles with different colors denote multiple target objects. The figure is better viewed in color.

Chapter 5

Visual Object Tracking Using Learned Hierarchical Features with Recursive Neural Networks

Recently, deep learning has achieved very promising results in visual object tracking. Deep neural networks in existing tracking methods require a lot of training data to learn a large number of parameters. However, training data is not sufficient for visual object tracking as annotations of a target object are only available in the first frame of a test sequence. In this chapter, a feature learning algorithm is proposed for visual object tracking to learn hierarchical features by using tree structure based Recursive Neural Networks (RNN), which have fewer parameters than other deep neural networks, e.g. Convolutional Neural Networks (CNN). First, RNN parameters are learned to discriminate between target and background in the first frame of a test sequence. Tree structure over local patches of an exemplar region is randomly generated by using a bottom-up greedy search strategy. Given the learned RNN parameters, two dictionaries regarding target regions and corresponding local patches are created based on learned hierarchical features from both

top and leaf nodes of multiple random trees. In each of the subsequent frames, sparse dictionary coding is conducted on all candidates to select the best candidate as the new target location. In addition, two dictionaries are online updated to handle appearance changes of target objects. Experimental results demonstrate that the proposed feature learning algorithm can significantly improve tracking performance on benchmark datasets.

5.1 Introduction

Visual object tracking aims to locate a target object in a video sequence given its location in the first frame. It is a very challenging problem because target appearance may vary dramatically due to illumination change, partial occlusion, object deformation, etc. To solve these issues, some trackers [4] employ local patch based appearance models to achieve very promising performance.

Feature representation is very important to object tracking. Recently, deep learning based trackers [23][144][145][146][147][148] have achieved very promising performance by using learned hierarchical features rather than raw pixel values or hand-crafted features. Deep neural networks usually require a lot of training data to learn a large number of parameters. However, training data is not sufficient for visual object tracking as annotations of a target object are only available in the first frame of a test sequence.

To overcome this problem, existing feature learning based trackers pre-train their neural networks by using auxiliary data and then fine-tune network parameters according to specific target objects. Different from these methods, a feature learning algorithm is proposed for visual object tracking to learn hierarchical features by using tree structure based Recursive Neural Networks (RNN), which have fewer parameters than other deep neural networks, e.g. Convolutional Neural Net-

works (CNN). As a result, the proposed feature learning algorithm does not require any network pre-training on auxiliary data and will not suffer from fine-tuning network parameters.

First, RNN parameters are learned to discriminate between target object and background in the first frame of a test sequence. Tree structure over local patches of an exemplar region is randomly generated by using a bottom-up greedy search strategy. Given the learned RNN parameters, two dictionaries regarding target regions and corresponding local patches are created based on the learned hierarchical features from both top and leaf nodes of multiple random trees. In each of the subsequent frames, sparse dictionary coding is conducted on all candidates to select the best candidate as the new target location. In addition, two dictionaries are online updated to handle appearance changes of target objects.

The main contribution is that hierarchical features are learned to discriminate between target and background by using RNN which can successfully encode spatial information among local patches of a target object based on multiple random trees. RNN features learned at top nodes of random trees are able to capture structural information of target objects, which are robust to holistic appearance changes caused by illumination change or object deformation. Moreover, RNN features learned at leaf nodes represent local patches and capture local appearance changes due to partial occlusion. Therefore, the hierarchical features learned from both top and leaf nodes are beneficial for visual object tracking. Experimental results demonstrate that the proposed feature learning algorithm can significantly improve tracking performance on the benchmark dataset [8].

5.2 Related Work

Visual Object Tracking. During the past few decades, visual object tracking have received much attention. Many tracking algorithms have achieved very promising results. Interested readers can refer to some recent surveys [8][149]. The proposed feature learning algorithm is integrated into a baseline tracker ASLA [4] belonging to generative trackers which focus on modeling target appearance without considering background. Other generative trackers utilize subspace learning [27][7], sparse coding [15][36][5], Gaussian mixture model [28], kernel-based model [31], visual decomposition model [17], etc. Although the baseline tracker in this work is generative, hierarchical features are learned discriminatively to differentiate target from background. Discriminative trackers formulate object tracking as a binary classification problem. They use many machine learning algorithms such as SVM [39][45][12][53], boosting [43][49][50], graph embedding [46], multiple instance learning [16], metric learning [150][54], Gaussian process regression [10], etc.

Deep Learning. Recently, deep learning has achieved very promising results in visual object tracking [23][144][145][146][147][148]. Usually, deep neural networks require a lot of training data. However, only the first frame of a test sequence is annotated in visual object tracking. To overcome this problem, some deep learning based trackers pre-train their neural networks by using auxiliary data, e.g. 80 million tiny images dataset [23], face detection dataset [144] and Hans van Hateren natural scene videos [145]. In contrast, the proposed feature learning method does not require any network pre-training on auxiliary data. Some other deep learning based trackers adopt existing deep neural networks, e.g. R-CNN model built upon Caffe Library [146] and VGG network pre-trained on ImageNet [147], and then fine-tune network parameters based on training data of specific target objects. Different from these methods, in this work, RNN parameters are learned in

the first frame of a test sequence and fixed in the subsequent frames. Therefore, the proposed feature learning algorithm does not suffer from fine-tuning network parameters.

Recursive Neural Networks. RNN has been successfully applied to natural language processing for sentiment analysis [151], phrase and sentence modeling [152] and paraphrase detection [153]. Also, RNN is applied for parsing natural scene [154] and 3D object classification [155]. This work is the first to learn hierarchical features by using RNN for visual object tracking.

5.3 Learning Hierarchical Features Using RNN

Deep learning has achieved very promising performance for visual object tracking. Some deep learning based trackers [23][144][145] require auxiliary data to pre-train their neural networks. This kind of pre-training is necessary for feature learning but inconvenient for visual object tracking. Moreover, some other trackers [146][147] employ deep neural networks pre-trained already and then fine-tune them during tracking. However, annotation of a target object in a test sequence is still limited for fine-tuning such deep neural networks with large numbers of parameters. To avoid inconvenient pre-training and fine-tuning, it is proposed to learn hierarchical features for visual object tracking by using tree-structure based Recursive Neural Networks (RNN) which have fewer parameters and hence require less training data. The RNN parameters are learned to discriminate target from background by using target annotation in the first frame of a test sequence and then fixed for extracting hierarchical features from candidate regions in the subsequent frames. Moreover, learned hierarchical features are able to capture spatial information over local patches of a target region based on multiple random trees. This capability is beneficial for visual object tracking.

In this section, the details of the proposed feature learning algorithm are presented. First, RNN is briefly introduced. Then, how to extract hierarchical features based on RNN is explained. Next, how to generate trees over local patches of target regions is described. Last, how to discriminatively learn RNN parameters is depicted.

5.3.1 Overview of RNN

RNN is a deep neural network established by applying the same set of parameters recursively over a certain structure. In this work, RNN is based on tree structure over local patches of a target object. RNN has three types of parameters: W^{raw} , W^{rnn} and W^{label} . W^{raw} and W^{rnn} are used to extract hierarchical features from candidate regions based on multiple random trees. W^{label} is to map extracted features regarding candidate regions into different classes (target object and background in this work). In the first frame of a test sequence, these three types of parameters are jointly learned together to discriminate target from background. In the subsequent frames, the learned RNN parameters W^{raw} and W^{rnn} are fixed and used to extract hierarchical features for candidate regions.

5.3.2 Extracting Features Using RNN

Given a tree over local patches of a target region (see details in Section 5.3.3) and learned RNN parameters W^{raw} and W^{rnn} (see details in Section 5.3.4), Figure 5.1 illustrates an instance of extracting hierarchical features from a target region. The local patch setting in ASLA [4] is employed to decompose a target object observation $x \in \mathbb{R}^{32 \times 32}$ into 9 overlapping local patches $p \in \mathbb{R}^{16 \times 16}$. Each local patch can be vectorized into an 256-dimensional raw pixel value feature $V_i \in \mathbb{R}^{256 \times 1}, i = 1, \dots, 9$. Then, a neural network layer is utilized to map raw pixel values (orange circles in Figure 5.1) into a n -dimensional RNN feature space (blue circles). These RNN

Figure 5.1: Illustration of extracting hierarchical features using RNN with a known tree structure over the local patches of a target object.

features at leaf nodes can be calculated as follows:

$$\zeta_i = f(W^{raw}V_i + b^{raw}), \quad (5.1)$$

where $W^{raw} \in \mathbb{R}^{n \times 256}$ is the transformation matrix, b^{raw} is the bias, f is the sigmoid function $f(x) = 1/(1 + e^{-x})$ and $\zeta_i \in \mathbb{R}^{n \times 1}$ is the RNN feature at leaf nodes.

In the given tree, each node is associated with the same basic neural network illustrated in Figure 5.1. The basic network computes parent features based on two child input nodes as follows:

$$\eta_{(i,j)} = f([W^{rnn}, W^{rnn}][\tau_i; \tau_j] + b^{rnn}), \quad (5.2)$$

where $W^{rnn} \in \mathbb{R}^{n \times n}$ is the transformation matrix, b^{rnn} is the bias, f is the sigmoid function $f(x) = 1/(1 + e^{-x})$, $[\tau_i; \tau_j] \in \mathbb{R}^{2n \times 1}$ is the concatenated feature for two child nodes and $\eta_{(i,j)} \in \mathbb{R}^{n \times 1}$ is the parent feature. Note that the child node pair $([\tau_i; \tau_j])$ possibly includes two leaf nodes $([\zeta_i; \zeta_j])$, or two non-terminal nodes $([\eta_i; \eta_j])$, or one leaf node and one non-terminal node $([\zeta_i; \eta_j])$. Given the tree structure and the RNN parameters, hierarchical features can be extracted from a target region at the top node of the RNN tree by recursively using the same basic neural network in a bottom-up manner. As a result, learned hierarchical features can capture spatial information over local patches of the target object.

5.3.3 Generating Random Trees

To extract hierarchical features using RNN, a tree structure over local patches of a candidate region is required. There is no objective criterion to identify the best tree. Therefore, trees are randomly generated by using a greedy bottom-up searching strategy. A number of trees are employed together to extract RNN features, which are expected to be robust to certain noisy tree structure.

First, it can be defined an adjacency matrix A , where $A(i, j) = 1$ if local patch i and j are spatially neighboring. It means that local patch i and j can be merged during generating random trees. Figure 5.2 illustrates the spatial layout of local patches in a target object and the corresponding adjacency matrix.

Given the adjacency matrix A , the set of pairs of neighboring local patches can be generated and denoted as follows:

$$C = \{[p_i, p_j] | A(i, j) = 1\}, \quad (5.3)$$

where p_i indicates the i th local patch. At the beginning (see Figure 5.2), there are the following pairs: $\{[p_1, p_2], [p_1, p_4], \dots, [p_9, p_6], [p_9, p_8]\}$. Suppose the pair $[p_i, p_j]$

Figure 5.2: Illustration of the spatial layout of local patches in a target object and the corresponding adjacency matrix. For example, patch 1 and 2 can be merged because they are spatially neighboring, so the corresponding matrix element is 1.

is randomly selected, all pairs with either p_i and p_j are removed from the set C as follows:

$$C = C - \{[p_i, p_m] | m \in Neigh(i)\} - \{[p_j, p_n] | n \in Neigh(j)\}, \quad (5.4)$$

where $Neigh()$ denotes the neighborhood of a local patch. Then, new child pairs are added to the set C as follows:

$$C = C \cup \{[p_{(i,j)}, p_k] | k \in Neigh(i) \cup Neigh(j), k \neq i, j\}, \quad (5.5)$$

where another pair is randomly selected to merge. In the same way, the set C is updated. Then, the previous steps are repeated until only one child node pair is left in the set C . Finally, the top node of the tree over all local patches of a target region can be achieved by merging the last child node pair in the set C .

The RNN tree shown in Figure 5.1 is used as an example. Circles on top of nodes indicate the index numbers of nodes. It can be observed that local patch 8 and 9 merge first and the corresponding child node pair $[p_8, p_9]$ is selected. Consequently, the set of pairs of neighboring local patches is updated accordingly as $C = \{[p_1, p_2], [p_1, p_4], [p_2, p_1], [p_2, p_3], [p_2, p_5], [p_3, p_2], [p_3, p_6], [p_4, p_1], [p_4, p_5], [p_4, p_7], [p_5, p_2],$

$[p_5, p_4], [p_5, p_6], [p_6, p_3], [p_6, p_5], [p_7, p_4], [p_{(8,9)}, p_5], [p_{(8,9)}, p_6], [p_{(8,9)}, p_7], [p_5, p_{(8,9)}], [p_6, p_{(8,9)}], [p_7, p_{(8,9)}]\}$. Next, another pair is randomly selected in the updated C . In this example, local patch 2 and 5 are subsequently merged. Then, the previous manipulations are repeated until only one child node pair is left in the set C . As illustrated in Figure 5.1, the tree over local patches of the target region is finally obtained after merging local patch 1 and non-terminal node 16.

5.3.4 Discriminative Learning of RNN Parameters

Previously, deep learning based tracking methods require pre-training from auxiliary data or fine-tuning neural networks pre-trained already according to specific target object. In contrast, the RNN parameters of the proposed feature learning algorithm are learned only based on target annotation and background data in the first frame of a test sequence. Then, the learned RNN parameters W^{raw} and W^{rnn} are fixed for extracting hierarchical features from candidate regions in the subsequent frames.

To discriminatively learn RNN parameters, a softmax layer is added with the parameter W^{label} , which connects RNN features at top nodes and class labels. Suppose there are N training samples $\mathbf{X} = \{x_1, \dots, x_N\}$ in terms of target and background, which are respectively sampled nearby and away from the target location in the first frame. Also, there are the corresponding labels $\mathbf{L} = \{l_1^{GT}, \dots, l_N^{GT}\}$, $l_i^{GT} \in \{[1; 0], [0; 1]\}$ ($[1; 0]$ and $[0; 1]$ indicate target and background respectively in the implementation). For each training sample x_i , RNN features can be extracted at the top node of a random tree based on the parameters W^{raw} and W^{rnn} . Then, the softmax layer illustrated in Figure 5.1 is applied for predicting class label:

$$l_i^{RNN} = \text{softmax}(W^{label}(\eta_{top}|T_i)), \quad (5.6)$$

where $W^{label} \in \mathbb{R}^{2 \times n}$ and η_{top} is the learned RNN features at the top node of a random tree $T_i \in \mathcal{T}(x_i)$. $\mathcal{T}(x_i)$ is all possible trees over local patches of x_i . Consequently, the loss between the predicted label and the ground truth label can be computed for each training sample by using the cross-entropy error as follows:

$$\delta(x_i, l_i^{GT} | \theta, T_i) = - \langle l_i^{GT}, \log(l_i^{RNN}) \rangle, \quad (5.7)$$

where $\theta = \{W^{raw}, W^{rnn}, W^{label}\}$. Finally, the objective function over all training samples can be denoted as follows:

$$\Phi(\theta) = \frac{1}{N} \sum_{i=1}^N \delta(\theta) + \frac{\lambda}{2} \|\theta\|^2 \quad (5.8)$$

where λ is the regularization parameter. The gradient of the objective function of Equation 5.8 w.r.t. the parameter set θ can be computed as follows:

$$\frac{\partial \Phi}{\partial \theta} = \frac{1}{N} \sum_i \frac{\partial \delta(\theta)}{\partial \theta} + \lambda \theta, \quad (5.9)$$

The optimization can be performed by using backpropagation through structure [156], which splits error messages at each node and then propagates to the child nodes. Then, L-BFGS [113] is employed to optimize the objective function.

5.4 Proposed Tracking System

Although the softmax layer of RNN can classify a candidate into target or background, it is relatively weak compared to the classifiers in state-of-the-art discriminative trackers. Therefore, the softmax layer is used to learn RNN parameters only and then learned hierarchical features are integrated into a state-of-the-art tracker ASLA [4] with an adaptive local sparse appearance model. Interested readers may

refer to ASLA [4] for more details.

Suppose there is an observation set of target $x_{1:t} = \{x_1, \dots, x_t\}$ up to the t^{th} frame and a corresponding feature representation set $z_{1:t} = \{z_1, \dots, z_t\}$. The target state y_t can be calculated as follows:

$$y_t = \arg \max_{y_t^i} p(y_t^i | z_{1:t}), \quad (5.10)$$

where y_t^i denotes the state of the i^{th} sample in the t^{th} frame. The posterior probability $p(y_t | z_{1:t})$ can be inferred by the Bayes' theorem as follows:

$$p(y_t | z_{1:t}) \propto p(z_t | y_t) \int p(y_t | y_{t-1}) p(y_{t-1} | z_{1:t-1}) dy_{t-1}, \quad (5.11)$$

where $z_{1:t}$ denotes the feature representation, $p(y_t | y_{t-1})$ denotes the motion model and $p(z_t | y_t)$ denotes the appearance model. In ASLA [4], the representations $z_{1:t}$ simply use raw pixel values of local patches, and the appearance model $p(z_t | y_t)$ employs sparse coding. In the proposed tracker, hierarchical features are learned from raw pixel values by using RNN. First, RNN parameters are learned by using 20 positive (nearby target) and 100 negative (background) samples in the first frame of a test sequence. Then, two dictionaries regarding target regions and corresponding local patches are created based on learned RNN features respectively from top and leaf nodes of multiple random trees in the first 10 frames. In each of subsequent frames, sparse coding is conducted on all candidate regions in terms of two dictionaries regarding target regions and local patches respectively. In addition, two dictionaries are online updated every 5 frames. In the implementations, the dimension size of RNN feature space, the regularization parameter in Equation 5.8 and the motion parameters in ASLA [4] are fixed as $n = 50$, $\lambda = 0.0001$ and $[10, 10, 0.01, 0, 0.005, 0]$ respectively for all test sequences. The proposed tracker using learned RNN features is summarized in Algorithm 4.

Algorithm 4 Proposed Tracker Using Learned RNN Features

- 1: **Input:** the last target state y_{t-1} , the RNN parameters θ learned in the first frame and two existing dictionaries respectively regarding target regions and local patches.
 - 2: Generate 600 target state candidates y_t^i and the corresponding image observations x_t^i nearby the previous target state y_{t-1} .
 - 3: Extract hierarchical features z_t^i from each image observation x_t^i by using the RNN parameter θ learned in the first frame.
 - 4: Calculate the posterior probability $p(y_t^i | z_{1:t})$ according to Equation 5.11.
 - 5: Predict the target state by $y_t = \arg \max_{y_t^i} p(y_t^i | z_{1:t})$.
 - 6: Update two dictionaries every 5 frames with learned RNN features respectively from top and leaf nodes of multiple random trees over the target region regarding the predicted state y_t .
 - 7: **Output:** the predicted target state y_t and two updated dictionaries.
-

5.5 Discussions

Parameter Size. As sharing parameters in different layers of tree structures, RNN has fewer parameters than other deep neural networks, e.g. Convolutional Neural Networks (CNN). The parameter set for RNN is $\theta = \{W^{raw}, W^{rnn}, W^{label}\}$, where $W^{raw} \in \mathbb{R}^{50 \times 256}$, $W^{rnn} \in \mathbb{R}^{50 \times 50}$ and $W^{label} \in \mathbb{R}^{2 \times 50}$. It can be found that the RNN model has totally 15400 parameters, which are much less than CNN (e.g. 60 million parameters in [26]). Therefore, the proposed feature learning algorithm using RNN does not require any pre-training and fine-tuning, which are actually not easy for visual tracking applications.

Computational Cost. Experiments are run on a PC without using any multi-core setting. The baseline tracker ASLA [4] can achieve about 3 fps. The proposed tracker using learned hierarchical features can achieve about 1.5 fps. There are two major factors affecting efficiencies of the proposed tracker: i) Learning RNN parameters in the first frame of a test sequence; ii) Extracting hierarchical features from candidate regions based on multiple random trees in each frame of the subsequent frames. As extracting features from 600 candidates consumes too much time, the proposed tracker conducts a coarse-to-fine strategy which first identifies

20 promising candidates according to tracking results from the baseline tracker ASLA [4] and then ranks these 20 candidates based on learned hierarchical features. Anyway, the main objective in this work is to show that learned hierarchical features using RNN can improve tracking performance. The efficiency of the proposed tracker could be enhanced by using parallel programming skills (e.g. parfor in MATLAB) or other advanced hardware (e.g. GPU).

5.6 Experiments

Benchmark. Tracking performance is evaluated on a recent public benchmark [8] containing 50 sequences which covers almost all challenging issues such as illumination changes, pose variations, occlusion, in/out-of-plane motions and cluttered background. The benchmark dataset reports the results from 29 trackers. Here, the proposed tracker “RNN” is compared with the top 6 trackers in the benchmark: ASLA [4], CXT [140], SCM [141], Struck [12], TLD [74] and VTS [142]. In addition, the proposed tracker is also compared with 3 recent state-of-the-art trackers: KCF [11], MEEM [9] and TGPR [10], which have reported their results on the benchmark.

The benchmark uses two measurements: i) Precision vs. Location error threshold, the percentage of the frames in which distances between tracking results and ground truth are below certain thresholds. ii) Success rate vs. Overlap threshold, the percentage of the frames in which overlapping percentages of tracking results against ground truth are higher than certain thresholds. Different trackers are ranked according to location error thresholding at 20 pixels for precision and Area Under Curve (AUC) for success rate. In addition, the one-pass evaluation (OPE) setting in the benchmark is used to generate the following experimental results.

Quantitative results. The precision plot and the success plot of tracking

Figure 5.3: Precision and success plots of the tracking results from the top 10 trackers on all sequences of the benchmark [8].

results from the top 10 trackers on all sequences of the benchmark [8] are presented in Figure 5.3. It can be found that the proposed tracker outperforms the baseline ASLA [4] in terms of both precision and success rate with large margins. This significant improvement can be owed to learned hierarchical features using tree structure based RNN which successfully encodes spatial information over local patches of target objects. Also, it can be observed that the proposed tracker using RNN features can achieve comparable performance against the state-of-the-art trackers.

To further evaluate tracking performance, in Table 5.1 and Table 5.2 are presented the comparison results from the top 10 trackers in terms of 11 attributes mentioned in the benchmark [8]. It can be found that the proposed tracker using learned RNN features can consistently improve the baseline ASLA [4] in terms of handling variational tracking challenges. Also, it can be observed that the proposed tracker achieves comparable performance against the state-of-the-art trackers in all attributes.

To investigate the effects of the numbers of random trees used in the RNN model, in Figure 5.4 are presented the comparison results from different variants of

Table 5.1: Average precision scores on different attributes: background clutter (BC), deformation (DEF), fast motion (FM), in-plane rotation (IPR), illumination variation (IV), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out-of-view (OV) and scale variation (SV). The best and the second best results are in red and green respectively.

	ASLA [4]	CXT [140]	KCF [11]	MEEM [9]	SCM [20]	Struck [12]	TGPR [10]	TLD [74]	VTS [142]	RNN
BC	0.496	0.443	0.753	0.808	0.578	0.585	0.761	0.428	0.578	0.779
DEF	0.445	0.422	0.740	0.859	0.586	0.521	0.768	0.512	0.487	0.762
FM	0.253	0.515	0.602	0.757	0.333	0.604	0.575	0.551	0.353	0.624
IPR	0.511	0.610	0.725	0.809	0.597	0.617	0.706	0.584	0.579	0.766
IV	0.517	0.501	0.728	0.778	0.594	0.558	0.687	0.537	0.573	0.726
LR	0.156	0.371	0.381	0.494	0.305	0.545	0.539	0.349	0.187	0.660
MB	0.278	0.509	0.650	0.740	0.339	0.551	0.578	0.518	0.375	0.623
OCC	0.460	0.491	0.749	0.814	0.640	0.564	0.708	0.563	0.534	0.743
OPR	0.518	0.574	0.729	0.853	0.618	0.597	0.741	0.596	0.604	0.780
OV	0.333	0.510	0.650	0.730	0.429	0.539	0.495	0.576	0.455	0.556
SV	0.552	0.550	0.679	0.808	0.672	0.639	0.703	0.606	0.582	0.750
Average	0.532	0.575	0.740	0.840	0.649	0.656	0.766	0.608	0.575	0.798

Table 5.2: Average success rates on different attributes: background clutter (BC), deformation (DEF), fast motion (FM), in-plane rotation (IPR), illumination variation (IV), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out-of-view (OV) and scale variation (SV). The best and the second best results are in red and green respectively.

	ASLA [4]	CXT [140]	KCF [11]	MEEM [9]	SCM [20]	Struck [12]	TGPR [10]	TLD [74]	VTS [142]	RNN
BC	0.408	0.338	0.535	0.578	0.450	0.458	0.543	0.345	0.428	0.587
DEF	0.372	0.324	0.534	0.582	0.448	0.393	0.556	0.378	0.368	0.570
FM	0.247	0.388	0.459	0.568	0.296	0.462	0.441	0.417	0.300	0.482
IPR	0.425	0.452	0.497	0.535	0.458	0.444	0.487	0.416	0.416	0.576
IV	0.429	0.368	0.493	0.548	0.473	0.428	0.486	0.399	0.429	0.563
LR	0.157	0.312	0.312	0.367	0.279	0.372	0.351	0.309	0.168	0.520
MB	0.258	0.369	0.497	0.565	0.298	0.433	0.440	0.404	0.304	0.485
OCC	0.376	0.372	0.514	0.563	0.487	0.413	0.494	0.402	0.398	0.565
OPR	0.422	0.418	0.495	0.569	0.470	0.432	0.507	0.420	0.425	0.579
OV	0.312	0.427	0.550	0.597	0.361	0.459	0.431	0.457	0.443	0.477
SV	0.452	0.389	0.427	0.510	0.518	0.425	0.443	0.421	0.400	0.573
Average	0.434	0.426	0.514	0.572	0.499	0.474	0.529	0.437	0.416	0.606

Figure 5.4: Precision and success plots of the tracking results from different variants of the proposed tracker on all sequences of the benchmark [8].

the proposed tracker in terms of both precision and success plots. It can be found that increasing the number of random trees can significantly enhance tracking performance. It is because that learned hierarchical features based on more tree structures can encode more spatial information over local patches of target objects.

Qualitative results. Some qualitative results from MEEM [9], TGPR [10], KCF [11], Struck [12] and the proposed tracker using RNN features on 10 sequences (car4, soccer, ironman, singer1, jogging-1, motorRolling, walking2, free-man4, carScale, tiger1) are presented in Figure 5.5, from which it can be observed that the proposed tracker can handle a large variety of tracking challenges, e.g. scale variation, occlusion, motion blur, illumination variation and in-plane rotation. In particular, the proposed tracker can achieve very promising performance in handling scale variation (car4, singer1, walking2, carScale). It is because that: i) the baseline tracker can generate candidates with different scales; ii) learned hierarchical features encode spatial information over local patches of target regions; iii) RNN features are learned to discriminate target from background. The later two factors further enhance the first factor for handling scale changes. As a result, the bounding boxes of the tracking results from the proposed tracker can be

Figure 5.5: Qualitative results comparing the proposed tracker (red) with the other 4 state-of-the-art trackers: MEEM [9] (Green), TGPR [10] (dark blue), KCF [11] (orange), Struck [12] (light blue) on 10 sequences (car4, soccer, ironman, singer1, jogging-1, motorRolling, walking2, freeman4, carScale, tiger1).

accurately fit to target regions.

5.7 Concluding Remarks

In this work, a feature learning algorithm is proposed for visual object tracking to learn hierarchical features by using tree structure based RNN which can encode spatial information over local patches of target objects. For tree structures, a bottom-up greedy searching strategy is used to generate random trees. Given a test sequence, RNN parameters are learned to discriminate target from background in the first frame and then fixed for extracting RNN features from candidate regions in the subsequent frames. With learned RNN parameters, two dictionaries regarding target regions and corresponding local patches are created based on learned RNN features respectively from top and leaf nodes of multiple random trees in the first 10 frames. In each of the subsequent frames, sparse dictionary coding is conducted on all candidate regions to find the optimal candidate as the new target location of the current frame. Additionally, two dictionaries are online updated to handle target appearance changes. Experimental results demonstrate that the proposed feature learning algorithm using RNN can significantly improve tracking performance.

Chapter 6

Object Detection with Occlusion Handling by Over-segmentation and Clustering on Foreground Regions

In this chapter, an occlusion handling method is proposed for object detection by using “3-D” data comprising image plane coordinates and depth values of pixels in foreground regions. First, background subtraction is used to extract foreground regions based on depth data. Then, each foreground region is over-segmented into subregions. Next, a number of segments are obtained by clustering subregions based on the proposed boundary based similarity metric using “3-D” data. As a result, occluded objects in a foreground region can be separately detected. Experimental results demonstrate that the proposed occlusion handling method using “3-D” data can significantly improve object detection performance.

6.1 Introduction

Object detection plays an important role in a wide range of computer vision applications such as video surveillance and human-machine interaction. Although many promising methods have been proposed for object detection, it is still challenging due to the occlusion problem. Recently, affordable depth sensors, such as the Microsoft Kinect depth sensor, have become available. The depth data is in fact a less ambiguous representation of the objects due to depth coherence, making it a better option for object detection. In this chapter, it will be argued that it is beneficial to make use of the “3-D” data for handling object occlusion.

The motivation of the proposed method is that 3-D data could be more discernable when coming to separate occluded objects than 2-D image based features due to the fact that occluded objects occupy different depth values. The straightforward idea is to cluster the pixels in a foreground region into segments according to the image plane coordinates and the depth values of all pixels. However, the number of the pixels in a foreground region could be larger than 10^4 . Due to the potential computational problem and the large ambiguity on determining the number of clusters, the existing clustering algorithms cannot easily handle this kind of large-scale data. In this chapter, a so-called “split-merge” approach is proposed to detect objects on foreground regions in two steps. First, the foreground region is over-segmented based on depth data. Then, the over-segmented subregions like the “superpixels” in [157] are clustered into segments using “3-D” data, which refers to the spatial coordinates (x, y) in the image plane and the depth value d . Although the idea is intuitive, clustering the over-segmented subregions is nontrivial because the depth values of an object could vary due to the typical surveillance camera setup, i.e. looking down to the scene at an angle instead of at level. This makes depth-plane slicing not feasible as the depth ranges of two objects in occlusion could overlap. To conquer this difficulty, a boundary based similarity metric is

proposed for clustering the subregions adequately into the segments corresponding to the detected objects.

6.2 Related Work

Over the past decades, many promising object detection methods have been proposed for many class-specific objects such as face [158] and pedestrian [1]. However, they often fail to detect occluded objects because the non-occluded samples are usually used as the training data. To overcome the occlusion issues, some object detection algorithms [159–161] have been proposed to employ a range of specific domain knowledge. In a more general way, Felzenszwalb et al. [13] propose to discriminatively train deformable part based models to detect occluded objects. However, when the occlusion is severe, Felzenszwalb’s method cannot work effectively as shown in the first and the third rows of Figure 6.4. Recently, inspired by deep learning, Ouyang et al. [162–164] propose to formulate the occlusion problem into the deep models which have achieved very promising performance in the computer vision community.

Before the prevalence of Kinect depth cameras, some object detection methods [165–167] had already made use of the depth data. After Kinect depth cameras becoming popular, many promising object detection methods [168–171] have been proposed based on the depth data obtained from Kinect. These methods can be divided into two categories, discriminative [168, 169] and generative [170, 171]. In the generative methods, Xia et al. [171] propose to use a 2-D head contour model and a 3-D head surface model to detect the human head region and then find the remaining human body by using a region growing algorithm. This method highly depends on the 3-D head surface model which is not reliable because the depth data around hair is very fragmented. Moreover, the region growing algorithm in

Figure 6.1: Illustration of the proposed method for object detection on foreground regions with occlusion handling. In the output, the bounding boxes with yellow dashed lines refer to the foreground regions; the bounding boxes with different colors and solid lines refer to the detected objects.

this work often fails because the depth values on a person could vary significantly.

6.3 Proposed Method

The proposed method is illustrated in Figure 6.1. First, the background subtraction is applied on the depth data to obtain the foreground regions. Then, the gradient based watershed transform [172] is used in each foreground to obtain the over-segmented subregions. Subsequently, a boundary based similarity metric is proposed to construct a graph in which the subregions are connected as nodes. At last, spectral clustering [173, 174] is employed to cluster the subregions into a number of segments which refer to the detected objects. As a result, the foreground objects occluded by each other can be detected separately.

6.3.1 Boundary Based Similarity Metric

Given the subregions in a foreground region as shown in Figure 6.2(a), a fully-connected graph $\mathcal{G} = (V, E)$ can be constructed for clustering the over-segmented subregions into a number of segments which refer to the detected objects. Each node $i \in V$ denotes a subregion in the foreground region and each edge (i, j)

Figure 6.2: Illustration of the boundary based similarity metric. (a) Over-segmented subregions in a foreground region; (b) The subset of K ($K = 10$) pairs of points from the i th to the j th boundary; (c) The subset of K ($K = 10$) pairs of points from the j th to the i th boundary.

denotes the pairwise similarity between two subregions. Since the depth values on one object could vary in a certain range and the depth values on different objects could overlap with each other, it is nontrivial to define a similarity metric which is capable of clustering the subregions correctly into the segments. In this subsection, a novel boundary based similarity metric will be presented.

To compute the similarities between the subregions, the “3-D” data comprising the spatial coordinates (x, y) in the image plane and the depth value d at each pixel is utilized. First, the boundaries are used to represent the subregions because the boundaries are able to depict the connectivities between two subregions. Moreover, using the pixels on the boundary of the subregion rather than all pixels in the subregion can reduce much computational cost. Therefore, the similarities between the subregions will be computed based on the boundaries.

Given two boundaries B_i and B_j , a straightforward way of computing the similarity between them is to compute the average similarity of all pairs of points on these two boundaries. However, when two adjacent boundaries are on the same object, the depth values of the pixels from these two boundaries could be significantly different, which could yield to a low average similarity between these two boundaries. Moreover, when two adjacent boundaries are on different objects, the depth values of the pixels from these two boundaries could be similar to each

other, which could lead to a high average similarity between these two boundaries. Therefore, using all pixels of the boundary cannot obtain the correct similarity between two adjacent boundaries on the same object.

To solve this issue, a subset of points on two boundaries (K pairs) are used to compute the similarity. The bi-directional subsets from B_i to B_j and from B_j to B_i are illustrated in Figure 6.2(b) and (c) respectively. For example, the subset $(p_k^i, p_k^j)_{k=1, \dots, K}$ from B_i to B_j is defined as follows

$$(p_k^i, p_k^j)_{k=1, \dots, K}^{p_k^i \in B_i, p_k^j \in B_j} = \min_{p_i \in B_i} \min_{p_j \in B_j}^1 d_{xy}(p_i, p_j), \quad (6.1)$$

where K indicates the number of the point pairs in the subset and $d_{xy}(p_i, p_j)$ denotes the Euclidean distance between two points with only considering spatial coordinates (x, y) in the image plane. Moreover, $\min_{p_j \in B_j}^1$ means to select the least pairwise distance from each point on B_i to all points on B_j . Then, $\min_{p_i \in B_i}^K$ means to select K least pairwise distances among points on B_i . Similarly, the subset $(\tilde{p}_k^i, \tilde{p}_k^j)_{k=1, \dots, K}$ from B_j to B_i is defined as follows

$$(\tilde{p}_k^i, \tilde{p}_k^j)_{k=1, \dots, K}^{\tilde{p}_k^i \in B_i, \tilde{p}_k^j \in B_j} = \min_{p_j \in B_j} \min_{p_i \in B_i}^1 d_{xy}(p_i, p_j). \quad (6.2)$$

For two adjacent subregions, the point pairs in two subsets are very close in the image plane. If two subregions are on the same object, the similarity by using the proposed subsets will be high because depth values on the same object usually change very smoothly in neighborhood regions. Sometimes, the body parts on the same object could be self-occluding and the depth values could change sharply in neighborhood regions. As a result, the similarity between two subregions could still be incorrect. However, this kind of inaccuracy will not affect the global

connectivity among the subregions on the same object. The proposed boundary based similarity metric does not require that similarities between subregions on the same object must always be high. It emphasizes more on the global connectivity among the subregions on the same object. In contrast, if two subregions are on two objects, the similarity by using the subsets will be low because the depth values on two objects change significantly. Therefore, using the proposed subsets for the similarity computation is able to correctly depict the relationships between the subregions.

Since the subsets are bi-directional, the average similarity between B_i and B_j is defined as follows

$$S(B_i, B_j) = 0.5 * (S_{B_i \rightarrow B_j} + S_{B_j \rightarrow B_i}), \quad (6.3)$$

where $S_{B_i \rightarrow B_j}$ and $S_{B_j \rightarrow B_i}$ indicate the similarities based on the bi-directional subsets from B_i to B_j and from B_j to B_i respectively. $S_{B_i \rightarrow B_j}$ and $S_{B_j \rightarrow B_i}$ can be defined respectively as follows

$$S_{B_i \rightarrow B_j} = \frac{1}{K} \sum_{k=1}^K \exp \left(-(p_k^i - p_k^j)^T \text{Diag}(\sigma)(p_k^i - p_k^j) \right), \quad (6.4)$$

$$S_{B_j \rightarrow B_i} = \frac{1}{K} \sum_{k=1}^K \exp \left(-(\tilde{p}_k^i - \tilde{p}_k^j)^T \text{Diag}(\sigma)(\tilde{p}_k^i - \tilde{p}_k^j) \right), \quad (6.5)$$

where p_k^i , p_k^j , $\tilde{p}_k^i$ and $\tilde{p}_k^j$ refer to the 3-D coordinates $[x, y, d]^T$ of the points on the boundaries B_i and B_j . (x, y) denotes the coordinate in the image plane and d denotes the depth value. K is the number of the point pairs in the subset and is

set to 10 in the experiment. The digonal matrix $\text{Diag}(\sigma)$ is defined as follows

$$\text{Diag}(\sigma) = \begin{bmatrix} \frac{1}{2\sigma_1^2} & 0 & 0 \\ 0 & \frac{1}{2\sigma_1^2} & 0 \\ 0 & 0 & \frac{1}{2\sigma_2^2} \end{bmatrix}, \quad (6.6)$$

where σ_1 and σ_2 control the neighborhood widths of the coordinates (x, y) in the image plane and the depth values d respectively. σ_1 and σ_2 are set to 10 and 100 respectively in the experiment.

Equations 6.4 and 6.5 compute the similarities by using not only the depth values d but also the coordinates (x, y) in the image plane. Using the coordinates (x, y) in the similarity computation aims to decrease the similarities between the non-adjacent subregions, which can separate the subregions with similar depth values but on different objects. Nevertheless, it will not decrease the similarities between the adjacent subregions on the same object because the points of the subsets between two adjacent boundaries are very close in the image plane.

6.3.2 Subregion Clustering

Given the boundary based similarity metric, a graph can be constructed by using the subregions as nodes. Then, spectral clustering [173] is employed to cluster the subregions into a number of segments which may refer to the detected objects. The affinity matrix A is defined as follows

$$A_{ij} = A_{ji} = S(B_i, B_j). \quad (6.7)$$

The Laplacian matrix L is defined like in [174] as follows

$$L = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}, \quad (6.8)$$

where $D = \text{Diag}(d_1, \dots, d_N)$, N is the number of subregions and $d_i = \sum_{j=1}^N A_{ij}$. The proposed algorithm for clustering the subregions in a foreground region is presented in Algorithm 5.

Algorithm 5 Clustering subregions into segments

Input: N subregions in a foreground region.

1. Compute the affinity matrix A as Equation 6.7.
2. Compute the Laplacian matrix L as Equation 6.8.
3. Select k eigenvectors $u_1, \dots, u_k$ of L corresponding to k smallest eigenvalues of L .
4. Let $U \in R^{N \times k}$ be the matrix including the eigenvectors $u_1, \dots, u_k$ as its columns.
5. Cluster row vectors $u_i, i = 1, \dots, N$ of the matrix U into k clusters $C_1, \dots, C_k$ by using the k-means algorithm.
6. Assign the i th subregion to the segment $S_j, j = 1, \dots, k$ if and only if the i th row of the matrix U belongs to the cluster C_j .

Output: Segments $S_i, i = 1, \dots, k$.

The spectral graph theory [175] believes that if there are k clusters, the first k smallest eigenvalues of the Laplacian matrix are close to 0. Therefore, a threshold value T_{eigval} on eigenvalues is used to determine whether an eigenvalue is close to 0. T_{eigval} is set to 0.01 in the experiment. The number of eigenvalues close to 0 is regarded as the number k of the segments in a foreground region.

6.3.3 Height Constraint of Target Objects

In this work, the target objects in the dataset are “person”, which has the certain range of heights (in pixels). Therefore, a height constraint for the objects in the dataset is proposed to eliminate the false positive detections caused by the noisy background. The heights (in pixels) of the target objects (person) in the dataset

are modeled as a Gaussian distribution. The probability that a detection bounding box with the height h indicates a target object (person) is defined as follows

$$p(h; \mu_h, \sigma_h) = \frac{1}{\sigma_h \sqrt{2\pi}} \exp \left\{ -\frac{1}{2} \left(\frac{h - \mu_h}{\sigma_h} \right)^2 \right\}, \quad (6.9)$$

where μ_h ($\mu_h = 143$) and σ_h ($\sigma_h = 32$) denote respectively the mean and the standard deviation of object heights in the training data, which includes 38 frame images and 5 persons. If the probability at a height is smaller than the threshold T_{prob} , the corresponding detection bounding box will be eliminated. T_{prob} is set to 0.0001 in the experiment.

6.4 Experiments

In this section, the proposed dataset used for the evaluation is firstly presented. Then, the proposed object detection method is evaluated by comparing it with the 2-D image based object detection method [13] and the 3-D data based one [171]. At last, the detection association based multi-object tracking performance is evaluated by feeding the detection results from the 2-D image based object detection method [13] and the proposed object detection method using the “3-D” data into a state-of-the-art multi-object tracker [176].

Dataset. A new dataset is built for evaluating the capability of handling object occlusions. The extent of occlusions in the dataset can be observed in Figure 6.4. The dataset consists of color images and the corresponding depth data, both of which are taken by a Kinect camera in a lab environment. The Kinect camera is mounted on a tripod and elevated to the roof so that the Kinect camera can look down onto the scene, in imitation of a typical surveillance camera scenario. The dataset contains 297 frame images with many occlusions which are challenging for object detection. 1066 persons in the dataset are manually labeled in bounding

boxes. The ground truths are used to evaluate the detection performance. The following experimental results are reported based on the proposed dataset.

Evaluation on Detection Performance. First of all, the measurement used in the detection competition of PASCAL VOC 2012 is employed to evaluate the detection performance quantitatively. In this work, the measurements include precision, recall and the average precision. Details can be found in [177]. The plots of precision versus recall on different methods are presented in Figure 6.3.

The proposed method has three baselines. The first one is to simply use the centers of the subregions to compute the similarities without the height constraint of target objects. The second baseline is to use all point pairs on the boundaries between two subregions to compute the similarities without the height constraint. The third baseline is to use the K point pairs on the boundaries between two subregions to compute the similarities without the height constraint. The proposed method uses the proposed boundary based similarity metric with the height constraint. In Figure 6.3, it can be observed that the precision and recall of the first baseline (the purple curve) is a significant improvement on the second one (the yellow curve) due to the use of all point pairs on the boundaries between two subregions. It demonstrates that the boundaries are representative of the subregions. Moreover, the precision and recall of the third baseline (the cyan curve) are higher than the second one because K point pairs on the boundaries are more adequate for depicting the relationships between two subregions than all point pairs. Furthermore, with the height constraint of target objects, the precision and recall of the third baseline (the cyan curve) can be improved to the proposed method (the red curve).

The plots in Figure 6.3 also demonstrate that the recall of Felzenszwalb's method [13] is lower than the proposed one. This is because Felzenszwalb's method fails to detect the occluded objects while the proposed one can handle the object

Figure 6.3: The plot of precision versus recall on different methods

occlusions well. Moreover, Xia's method [171] obtains low precision and recall as shown in Figure 6.3. The reasons are as follows. The first is that the 3-D hemisphere model does not work well on the proposed dataset and false positives cannot be eliminated. The second is that two or more seed regions on the same object can be detected, which yields to multiple detections for one object. These two reasons cause the low precision. The last one is that the parameters of the region growing in Xia's method is so sensitive that the object regions are not accurate especially when the occlusions happen. This reason causes the low recall.

To better evaluate the performance of the proposed method, Table 6.1 presents the average precisions on different methods. The average precision is more comprehensive because it jointly considers precisions and recalls. From Table 6.1, it can be observed that the proposed method outperforms three baselines and the other two object detection methods in terms of the average precision.

The qualitative results from Felzenszwalb's method [13] and the proposed one

Table 6.1: The average precisions on different methods

Methods	Precision	Recall	Average Precision
Xia’s method [171]	12.94%	37.80%	5.28%
Felzenszwalb’s method [13]	86.05%	62.48%	56.71%
Baseline1 of the proposed method	22.29%	52.63%	11.84%
Baseline2 of the proposed method	64.02%	81.61%	55.05%
Baseline3 of the proposed method	84.99%	89.21%	78.16%
The proposed method	90.06%	88.37%	81.69%

are shown in Figure 6.4. The first and the third rows refer to the results from Felzenszwalb’s method which only uses the color images. The second and last rows show the corresponding results from the proposed method which uses not only the color images but also the depth data. The qualitative comparisons demonstrate that the proposed method outperforms Felzenszwalb’s method. The success of the proposed method is due to the capability of handling object occlusion.

Table 6.2 presents the run-time performances of different methods on the proposed dataset. The term “Total Run-Time” in the table refers to the time for processing all frame images in the proposed dataset. The term “Average Run-Time” is computed as follows

$$\text{Average Run-Time} = \frac{\text{Total Run-Time}}{297}. \quad (6.10)$$

It is necessary to mention that Xia’s method [171] and the proposed method are performed on a PC with a Quad-Core 3.30 GHz CPU and 8 GB RAM while Felzenszwalb’s method [13] is performed on a PC with a Quad-Core 2.83 GHz CPU and 4 GB RAM. The reason for using two PCs is that Xia’s method and the proposed method both need to run on a Windows system while Felzenszwalb’s method needs to run on a Linux system. It is fair to compare the efficiency

Figure 6.4: Illustration of the qualitative results. The first and the third rows refer to the results from Felzenszwalb’s method [13] which only uses the color image data. The second and the last rows refer to the corresponding results from the proposed method using the “3-D” data.

Table 6.2: The run-time performances on different methods

Methods	Total Run-Time (sec)	Average Run-Time (sec)
Xia’s method [171]	6367	21.4
Felzenswalb’s method [13]	3230	10.9
The proposed method	1630	5.5

performance between Xia’s method and the proposed one because two methods run on the same PC. It can be found from Table 6.2 that the proposed method is faster than Xia’s method because the proposed method does not require region growing which is computationally expensive. Since Felzenswalb’s method and the proposed method run on different PCs, it is unfair to compare the run-time performances between these two methods. However, from the results in Table 6.2, it can be found that the performance of the proposed method is at least comparable to Felzenswalb’s method.

Evaluation on Tracking-By-Detection Performance. In this subsection, the tracking-by-detection performances are evaluated based on the detection results respectively from Felzenswalb’s method [13] and the proposed one. A motion cue based multi-object tracking method [176] is employed for the evaluation which still aims to compare the detection performances between Felzenswalb’s method and the proposed one.

Table 6.3 presents the tracking performances based on the detection results from Felzenswalb’s method [13] and the proposed one. Note that only the trajectories with more than 10 frames are evaluated. “Traj_ID” denotes the index of the trajectory of the specific target object. “FN_GT” means the number of the frames that the trajectory occupies in the ground truth. Similarly, “FN_Fels[13]” and “FN_Ours” indicate the number of the frames which the trajectory occupies in the results respectively from Felzenswalb’s method [13] and the proposed one. Correspondingly, “SR_Fels[13]” and “SR_Ours” denote the success rates which are

Table 6.3: Tracking performances based on the detection results from Felzenswalb’s method [13] and the proposed one. Note that only the trajectories with more than 10 frames are evaluated. “Traj_ID” denotes the index of the trajectory of the specific target object. “FN_GT” means the number of the frames that the trajectory occupies in the ground truth. Similarly, “FN_Fels[13]” and “FN_Ours” indicate the number of the frames which the trajectory occupies in the results respectively from Felzenswalb’s method [13] and the proposed one. Correspondingly, “SR_Fels[13]” and “SR_Ours” denote the success rates which are respectively defined as $\frac{FN_Fels[13]}{FN_GT}$ and $\frac{FN_Ours}{FN_GT}$. For the left four columns, higher values indicate better performances, which are in bold fonts.

Traj_ID	FN_GT	FN_Fels[13]	FN_Ours	SR_Fels[13]	SR_Ours
1	24	18	24	0.75	1.00
2	24	20	24	0.83	1.00
3	14	6	13	0.43	0.93
4	14	11	12	0.79	0.86
5	14	6	11	0.43	0.79
6	14	13	13	0.93	0.93
7	12	1	7	0.08	0.58
8	12	7	6	0.58	0.50
9	10	8	8	0.80	0.80
10	12	12	12	1.00	1.00
11	12	8	10	0.67	0.83
12	12	4	9	0.33	0.75
13	10	5	8	0.50	0.80
14	31	31	29	1.00	0.94
15	17	8	17	0.47	1.00
16	24	19	22	0.79	0.92
17	16	6	15	0.38	0.94
18	14	14	13	1.00	0.93
19	14	11	14	0.79	1.00
20	10	7	10	0.70	1.00
21	95	59	60	0.62	0.63
22	12	12	11	1.00	0.92
23	69	60	61	0.87	0.88
24	83	58	69	0.70	0.83
25	83	55	74	0.66	0.89
26	83	32	74	0.39	0.89
27	11	11	11	1.00	1.00
28	11	8	11	0.73	1.00
29	12	12	12	1.00	1.00

Figure 6.5: Illustration of the tracking-by-detection results. The first and the third rows refer to the results based on the detections from Felzenszwalb’s method [13] which only uses the color image data. The second and the last rows refer to the corresponding results based on the detections from the proposed method using the “3-D” data.

respectively defined as $\frac{FN_{Fels[13]}}{FN_{GT}}$ and $\frac{FN_{Ours}}{FN_{GT}}$. For the left four columns, higher values indicate better performances, which are in bold fonts. It can be observed that the proposed method outperforms Felzenswalb’s method [13] in most of the trajectories. The reason behind this success is because the proposed method can detect more occluded objects. Figure 6.5 presents some qualitative results, which further demonstrate that the proposed method can handle object occlusion better than Felzenswalb’s method [13].

6.5 Concluding Remarks

In this chapter, an occlusion handling method is developed for object detection by using “3-D” data comprising image plane coordinates and depth values of pixels in foreground regions. First, background subtraction is employed to extract foreground regions based on depth data. Then, each foreground region is over-segmented into subregions, which are clustered into a number of segments based on a boundary based similarity metric and “3-D” data. As a result, occluded objects in foreground regions can be separately detected. Experimental results demonstrate that the proposed occlusion handling method can significantly improve object detection performance.

Chapter 7

Conclusions and Future Works

7.1 Conclusions

In this thesis, some challenging problems in tracking and detecting objects have been studied in four pieces of works.

In the first piece of work, a feature learning algorithm is developed for visual object tracking to learn hierarchical features based on domain adaptation. First, the generic features robust to complicated motion transformations are learned from auxiliary video sequences by using a two-layer convolutional neural network under the temporal slowness constraint. Then, a domain adaptation module is proposed to adapt the pre-learned features according to specific target objects. As a result, learned hierarchical features are robust to both complicated motion transformations and appearance changes of specific target object. Experimental results demonstrate that the proposed feature learning algorithm can significantly enhance visual object tracking performance.

In the second piece of work, a feature learning algorithm is developed for multiple object tracking to learn hierarchical features by using a joint learning strategy. The framework of the feature learning algorithm follows the first work. To achieve

better performance for feature adaptation, a joint learning strategy is proposed to jointly conduct feature adaptation by using all target annotation data. The proposed feature adaptation module can simultaneously learn common features shared by all target objects and individual features for each object. The learned common features are required to be perpendicular to all individual features. Moreover, the individual features are required to be distinct to each other. As a result, learned hierarchical features are more discriminative and compact. Experimental results demonstrate that the proposed feature learning algorithm can significantly improve multiple object tracking performance.

In the the third piece of work, a feature learning algorithm is proposed for visual object tracking to learn hierarchical features by using tree structure based Recursive Neural Networks (RNN), which have fewer parameters and do not require any feature pre-learning from auxiliary data. First, RNN parameters are learned to discriminate between target and background only based on the target annotation in the first frame of a test sequence. Tree structure over local patches of an exemplar region is randomly generated by using a bottom-up greedy search strategy. Given the learned RNN parameters, two dictionaries regarding target regions and corresponding local patches are created based on learned hierarchical features from both top and leaf nodes of multiple random trees. In each of the subsequent frames, sparse dictionary coding is conducted on all candidates to select the best candidate as the new target location. In addition, two dictionaries are online updated to handle appearance changes of target objects. Experimental results demonstrate that the proposed feature learning algorithm can significantly improve visual object tracking performance.

In the last piece of work, an occlusion handling method is proposed for object detection by using “3-D” data comprising image plane coordinates and depth values of pixels in foreground regions. First, background subtraction is employed to

extract foreground regions based on depth data. Then, each foreground region is over-segmented into subregions. Next, a number of segments are obtained by clustering subregions based on the proposed boundary based similarity metric using “3-D” data. As a result, occluded objects in foreground regions can be separately detected. Experimental results demonstrate that the proposed occlusion handling method using “3-D” data can significantly improve object detection performance.

7.2 Future Works

In the first piece of work, a two-layer neural network is used to learn hierarchical features. Recently, more sophisticated deep neural networks have been used to improve feature learning performance. For example, Convolutional Neural Networks (CNN) has achieved very promising performance in some recent tracking methods [146][148][147]. In the future works, CNN can be used in the proposed domain adaptation based feature learning algorithm to facilitate visual object tracking.

In the second piece of work, the proposed feature learning algorithm does not utilize spatial constraints among multiple target objects. As shown in SPOT tracker [78], leveraging spatial relations among multiple target objects is beneficial for visual object tracking. In future work, this kind of spatial constraints can be integrated into the proposed feature learning algorithm to further improve multiple object tracking performance.

In the third piece of work, learned hierarchical features are largely dependent on tree structures over local patches of target regions, which are randomly generated in this work. Powerful feature representations will be extracted based on a “good” tree. In future work, a tree filtering scheme could be investigated to select “good” trees for extracting more useful hierarchical features.

In the last piece of work, similarities between subregions are calculated based

on only image plane coordinates and depth values of pixels in foreground regions. Recently, RGB-D descriptors [178][179] have achieved very promising performance in visual object recognition. In future work, RGB information can be incorporated into the existing occlusion handling method using only depth information to further improve object detection performance.

Author's Publications

1. **Li Wang**, Ting Liu, Bing Wang, Kap Luk Chan, and Gang Wang, “Learning Hierarchical Features for Visual Object Tracking with Recursive Neural Networks”, submitted to IEEE Conference on Computer Vision and Pattern Recognition, 2016.
2. Bing Wang, **Li Wang**, Bing Shuai, Zhen Zuo, Ting Liu, Kap Luk Chan, and Gang Wang, “Joint Learning of Convolutional Neural Networks and Temporally Constrained Metrics for Tracklet Association”, submitted to IEEE Conference on Computer Vision and Pattern Recognition, 2016.
3. Ting Liu, **Li Wang**, Bing Wang, and Gang Wang, “Visual Tracking via Large Scale Classification”, submitted to IEEE Conference on Computer Vision and Pattern Recognition, 2016.
4. Bing Wang, Ting Liu, **Li Wang**, Bing Shuai, Kap Luk Chan and Gang Wang, “Multi-Target Tracking: A Benchmark”, submitted to IEEE Conference on Computer Vision and Pattern Recognition, 2016.
5. **Li Wang**, Ting Liu, Gang Wang, Kap Luk Chan, and Qingxiong Yang, “Video Tracking Using Learned Hierarchical Features”, IEEE Transactions on Image Processing, vol. 24, no. 4, pp. 1424-1435, 2015.
6. Ting Liu, Gang Wang, **Li Wang**, and Kap Luk Chan, “Visual Tracking via

- Temporally Smooth Sparse Coding”, IEEE Signal Processing Letters, vol. 22, no. 9, pp. 1452-1456, 2015.
7. Bing Wang, Gang Wang, Kap Luk Chan, and **Li Wang**, “Tracklet Association by Online Target-Specific Metric Learning and Coherent Dynamics Estimation”, IEEE Transactions on Pattern Analysis and Machine Intelligence, under major revision, 2015.
 8. Ting Liu, Gang Wang, Qingxiong Yang, and **Li Wang**, “Part-based tracking via discriminative correlation filters”, IEEE Transactions on Circuits and Systems for Video Technology, under major revision, 2015.
 9. **Li Wang**, Nam Trung Pham, Tian-Tsong Ng, Gang Wang, Kap Luk Chan, and Karianto Leman, “Learning Deep Features for Multiple Object Tracking by Using A Multi-Task Learning Strategy”, IEEE International Conference on Image Processing, 2014, pp. 838-842.
 10. Bing Wang, Gang Wang, Kap Luk Chan, and **Li Wang**, “Tracklet Association with Online Target-Specific Metric Learning”, in IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1234-1241.
 11. **Li Wang**, Kap Luk Chan, and Gang Wang, “Human Detection with Occlusion Handling by Over-Segmentation and Clustering on Foreground Regions”, in Asian Conference on Computer Vision Workshops, 2012, pp.197-208.
 12. Junyan Wang, **Li Wang**, Kap Luk Chan, and Martin Constable, “A Linear Programming Based Method for Joint Object Region Matching and Labeling”, in Asian Conference on Computer Vision, 2012, pp. 66-78.

Bibliography

- [1] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.
- [2] W. Y. Zou, A. Y. Ng, S. Zhu, and K. Yu, “Deep learning of invariant features via simulated fixations in video,” in *Annual Conference on Neural Information Processing Systems*, 2012, pp. 3212–3220.
- [3] C. F. Cadieu and B. A. Olshausen, “Learning transformational invariants from natural movies,” in *Annual Conference on Neural Information Processing Systems*, 2008, pp. 209–216.
- [4] X. Jia, H. Lu, and M.-H. Yang, “Visual tracking via adaptive structural local sparse appearance model,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1822–1829.
- [5] C. Bao, Y. Wu, H. Ling, and H. Ji, “Real time robust l1 tracker using accelerated proximal gradient approach,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1830–1837.
- [6] K. Zhang, L. Zhang, and M.-H. Yang, “Real-time compressive tracking,” in *European Conference on Computer Vision*, 2012, pp. 864–877.

-
- [7] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, “Incremental learning for robust visual tracking,” *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 125–141, 2008.
 - [8] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418.
 - [9] J. Zhang, S. Ma, and S. Sclaroff, “MEEM: robust tracking via multiple experts using entropy minimization,” in *European Conference on Computer Vision*, 2014, pp. 188–203.
 - [10] J. Gao, H. Ling, W. Hu, and J. Xing, “Transfer learning based visual tracking with gaussian processes regression,” in *European Conference on Computer Vision*, 2014, pp. 188–203.
 - [11] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.
 - [12] S. Hare, A. Saffari, and P. H. S. Torr, “Struck: Structured output tracking with kernels,” in *IEEE International Conference on Computer Vision*, 2011, pp. 263–270.
 - [13] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.
 - [14] A. Adam, E. Rivlin, and I. Shimshoni, “Robust fragments-based tracking using the integral histogram,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 798–805.

- [15] X. Mei and H. Ling, “Robust visual tracking using ℓ_1 minimization,” in *IEEE International Conference on Computer Vision*, 2009, pp. 1436–1443.
- [16] B. Babenko, M.-H. Yang, and S. J. Belongie, “Visual tracking with online multiple instance learning,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 983–990.
- [17] J. Kwon and K. M. Lee, “Visual tracking decomposition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1269–1276.
- [18] B. Liu, J. Huang, L. Yang, and C. A. Kulikowski, “Robust tracking using local sparse appearance model and k-selection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1313–1320.
- [19] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, “Robust visual tracking via multi-task sparse learning,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2042–2049.
- [20] W. Zhong, H. Lu, and M.-H. Yang, “Robust object tracking via sparsity-based collaborative model,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1838–1845.
- [21] D. Wang, H. Lu, and M.-H. Yang, “Online object tracking with sparse prototypes,” *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 314–325, 2013.
- [22] D. Wang, H. Lu, and M. H. Yang, “Least soft-threshold squares tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2371–2378.

- [23] N. Wang and D.-Y. Yeung, “Learning a deep compact image representation for visual tracking,” in *Annual Conference on Neural Information Processing Systems*, 2013, pp. 809–817.
- [24] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [25] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, “Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.
- [26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Annual Conference on Neural Information Processing Systems*, 2012, pp. 1106–1114.
- [27] M. J. Black and A. D. Jepson, “Eigentracking: Robust matching and tracking of articulated objects using a view-based representation,” *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.
- [28] S. J. McKenna, Y. Raja, and S. Gong, “Tracking colour objects using adaptive mixture models,” *Image and Vision Computing*, vol. 17, no. 3-4, pp. 225–231, 1999.
- [29] X. Hou, S. Z. Li, H. Zhang, and Q. Cheng, “Direct appearance models,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 828–833.
- [30] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, “Robust online appearance models for visual tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1296–1311, 2003.

- [31] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–575, 2003.
- [32] C. Yang, R. Duraiswami, and L. S. Davis, “Efficient mean-shift tracking via a new similarity measure,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 176–183.
- [33] K. Lee and D. J. Kriegman, “Online learning of probabilistic appearance manifolds for video-based recognition and tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 852–859.
- [34] H. Wang, D. Suter, K. Schindler, and C. Shen, “Adaptive object tracking based on an effective appearance filter,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1661–1667, 2007.
- [35] A. Yilmaz, “Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- [36] H. Li, C. Shen, and Q. Shi, “Real-time visual tracking using compressive sensing,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1305–1312.
- [37] X. Li, A. R. Dick, C. Shen, A. van den Hengel, and H. Wang, “Incremental learning of 3d-dct compact representations for robust visual tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 4, pp. 863–881, 2013.
- [38] X. Li, W. Hu, C. Shen, Z. Zhang, A. R. Dick, and A. van den Hengel, “A survey of appearance models in visual object tracking,” *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 4, p. 58, 2013.

-
- [39] S. Avidan, “Support vector tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1064–1072, 2004.
 - [40] R. Lin, M. Yang, and S. E. Levinson, “Object tracking using incremental fisher discriminant analysis,” in *International Conference on Pattern Recognition*, 2004, pp. 757–760.
 - [41] O. Williams, A. Blake, and R. Cipolla, “Sparse bayesian learning for efficient visual tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1292–1304, 2005.
 - [42] J. Wang, X. Chen, and W. Gao, “Online selecting discriminative tracking features using particle filter,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 1037–1042.
 - [43] H. Grabner and H. Bischof, “On-line boosting and vision,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 260–267.
 - [44] H. T. Nguyen and A. W. M. Smeulders, “Robust tracking using foreground-background texture discrimination,” *International Journal of Computer Vision*, vol. 69, no. 3, pp. 277–293, 2006.
 - [45] F. Tang, S. Brennan, Q. Zhao, and H. Tao, “Co-tracking using semi-supervised support vector machines,” in *IEEE International Conference on Computer Vision*, 2007, pp. 1–8.
 - [46] X. Zhang, W. Hu, S. J. Maybank, and X. Li, “Graph based discriminative learning for robust and efficient object tracking,” in *IEEE International Conference on Computer Vision*, 2007, pp. 1–8.

- [47] H. Grabner, C. Leistner, and H. Bischof, “Semi-supervised on-line boosting for robust tracking,” in *European Conference on Computer Vision*, 2008, pp. 234–247.
- [48] G. Li, D. Liang, Q. Huang, S. Jiang, and W. Gao, “Object tracking using incremental 2d-lda learning and bayes inference,” in *IEEE International Conference on Image Processing*, 2008, pp. 1568–1571.
- [49] R. Liu, J. Cheng, and H. Lu, “A robust boosting tracker with minimum error bound in a co-training framework,” in *IEEE International Conference on Computer Vision*, 2009, pp. 1459–1466.
- [50] B. Zeisl, C. Leistner, A. Saffari, and H. Bischof, “On-line semi-supervised multiple-instance boosting,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, p. 1879.
- [51] Y. Zha, Y. Yang, and D. Bi, “Graph-based transductive learning for robust visual tracking,” *Pattern Recognition*, vol. 43, no. 1, pp. 187–196, 2010.
- [52] W. Luo, X. Li, W. Li, and W. Hu, “Robust visual tracking via transfer learning,” in *IEEE International Conference on Image Processing*, 2011, pp. 485–488.
- [53] Y. Bai and M. Tang, “Robust tracking via weakly supervised ranking SVM,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1854–1861.
- [54] N. Jiang, W. Liu, and Y. Wu, “Order determination and sparsity-regularized metric learning adaptive visual tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1956–1963.

- [55] R. Lin, D. A. Ross, J. Lim, and M. Yang, “Adaptive discriminative generative model and its applications,” in *Annual Conference on Neural Information Processing Systems*, 2004, pp. 801–808.
- [56] H. Grabner, P. M. Roth, and H. Bischof, “Eigenboosting: Combining discriminative and generative information,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- [57] Y. Lei, X. Ding, and S. Wang, “Visual tracker using sequential bayesian learning: Discriminative, generative, and hybrid,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 38, no. 6, pp. 1578–1591, 2008.
- [58] Q. Yu, T. B. Dinh, and G. G. Medioni, “Online tracking and reacquisition using co-trained generative and discriminative trackers,” in *European Conference on Computer Vision*, 2008, pp. 678–691.
- [59] M. Yang, Z. Fan, J. Fan, and Y. Wu, “Tracking nonstationary visual appearances by data-driven adaptation,” *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1633–1644, 2009.
- [60] C. Shen, J. Kim, and H. Wang, “Generalized kernel-based visual tracking,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 1, pp. 119–130, 2010.
- [61] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *International Joint Conference on Artificial Intelligence*, 1981, pp. 674–679.
- [62] R. T. Collins, Y. Liu, and M. Leordeanu, “Online selection of discriminative tracking features,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1631–1643, 2005.

- [63] H. Grabner, M. Grabner, and H. Bischof, “Real-time tracking via on-line boosting,” in *British Machine Vision Conference*, 2006, pp. 47–56.
- [64] F. Porikli, O. Tuzel, and P. Meer, “Covariance tracking using model update based on lie algebra,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 728–735.
- [65] S. Avidan, “Ensemble tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 261–271, 2007.
- [66] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade, “Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1728–1740, 2008.
- [67] J. Kwon and K. M. Lee, “Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1208–1215.
- [68] Y. Wu, J. Cheng, J. Wang, H. Lu, J. Wang, H. Ling, E. Blasch, and L. Bai, “Real-time probabilistic covariance tracking with efficient model update,” *IEEE Transactions on Image Processing*, vol. 21, no. 5, pp. 2824–2837, 2012.
- [69] D. Wang, H. Lu, and M.-H. Yang, “Online object tracking with sparse prototypes,” *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 314–325, 2013.
- [70] M. Isard and A. Blake, “CONDENSATION - conditional density propagation for visual tracking,” *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

-
- [71] J. Fan, Y. Wu, and S. Dai, “Discriminative spatial attention for robust tracking,” in *European Conference on Computer Vision*, 2010, pp. 480–493.
 - [72] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, “Robust online appearance models for visual tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 415–422.
 - [73] I. Matthews, T. Ishikawa, and S. Baker, “The template update problem,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 810–815, 2004.
 - [74] Z. Kalal, J. Matas, and K. Mikolajczyk, “P-n learning: Bootstrapping binary classifiers by structural constraints,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 49–56.
 - [75] W. Hu, X. Li, W. Luo, X. Zhang, S. J. Maybank, and Z. Zhang, “Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2420–2440, 2012.
 - [76] X. Zhao, D. Gong, and G. G. Medioni, “Tracking using motion patterns for very crowded scenes,” in *European Conference on Computer Vision*, 2012, pp. 315–328.
 - [77] W. Luo and T.-K. Kim, “Generic object crowd tracking by multi-task learning,” in *British Machine Vision Conference*, 2013.
 - [78] L. Zhang and L. van der Maaten, “Structure preserving object tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1838–1845.

- [79] W. Luo, T. Kim, B. Stenger, X. Zhao, and R. Cipolla, “Bi-label propagation for generic multiple object tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1290–1297.
- [80] L. Wang, N. T. Pham, T.-T. Ng, G. Wang, K. L. Chan, and K. Leman, “Learning deep features for multiple object tracking by using a multi-task learning strategy,” in *IEEE International Conference on Image Processing*, 2014.
- [81] W. Luo, X. Zhao, and T. Kim, “Multiple object tracking: A review,” *Computing Research Repository*, vol. abs/1409.7618, 2014.
- [82] I. Haritaoglu, D. Harwood, and L. S. Davis, “W4: who? when? where? what? A real time system for detecting and tracking people,” in *International Conference on Face and Gesture Recognition*, 1998, pp. 222–227.
- [83] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 1999, pp. 2246–2252.
- [84] S. J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, “Tracking groups of people,” *Computer Vision and Image Understanding*, vol. 80, no. 1, pp. 42–56, 2000.
- [85] N. Oliver, B. Rosario, and A. Pentland, “A bayesian computer vision system for modeling human interactions,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843, 2000.
- [86] A. Cavallaro and T. Ebrahimi, “Accurate video object segmentation through change detection,” in *IEEE International Conference on Multimedia and Expo*, 2002, pp. 445–448.

- [87] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian, “Foreground object detection from videos containing complex background,” in *ACM International Conference on Multimedia*, 2003, pp. 2–10.
- [88] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. S. Davis, “Background modeling and subtraction by codebook construction,” in *IEEE International Conference on Image Processing*, 2004, pp. 3061–3064.
- [89] —, “Real-time foreground-background segmentation using codebook model,” *Real-Time Imaging*, vol. 11, no. 3, pp. 172–185, 2005.
- [90] M. Heikkilä and M. Pietikäinen, “A texture-based method for modeling the background and detecting moving objects,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 657–662, 2006.
- [91] Z. Zivkovic and F. van der Heijden, “Efficient adaptive density estimation per image pixel for the task of background subtraction,” *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, 2006.
- [92] L. Maddalena and A. Petrosino, “A self-organizing approach to background subtraction for visual surveillance applications,” *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1168–1177, 2008.
- [93] A. Ilyas, M. Scuturici, and S. Miguet, “Real time foreground-background segmentation using a modified codebook model,” in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2009, pp. 454–459.
- [94] O. Barnich and M. V. Droogenbroeck, “Vibe: A powerful random technique to estimate the background in video sequences,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2009, pp. 945–948.

- [95] V. Q. Pham, K. Takahashi, and T. Naemura, “Foreground-background segmentation using iterated distribution matching,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 2113–2120.
- [96] D. Farcas, C. Marghes, and T. Bouwmans, “Background subtraction via incremental maximum margin criterion: a discriminative subspace approach,” *Machine Vision and Applications*, vol. 23, no. 6, pp. 1083–1101, 2012.
- [97] S. Brutzer, B. Höferlin, and G. Heidemann, “Evaluation of background subtraction techniques for video surveillance,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1937–1944.
- [98] C. K. Eveland, K. Konolige, and R. C. Bolles, “Background modeling for segmentation of video-rate stereo sequences,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 1998, pp. 266–271.
- [99] G. G. Gordon, T. Darrell, M. Harville, and J. Woodfill, “Background estimation and removal based on range and color,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 1999, pp. 2459–2464.
- [100] Y. A. Ivanov, A. F. Bobick, and J. Liu, “Fast lighting independent background subtraction,” *International Journal of Computer Vision*, vol. 37, no. 2, pp. 199–207, 2000.
- [101] T. Darrell, D. Demirdjian, N. Checka, and P. F. Felzenszwalb, “Plan-view trajectory estimation with dense stereo background models,” in *IEEE International Conference on Computer Vision*, 2001, pp. 628–635.
- [102] M. Harville, G. G. Gordon, and J. Woodfill, “Foreground segmentation using adaptive mixture models in color and depth,” in *IEEE Workshop on Detection and Recognition of Events in Video*, 2001, pp. 3–11.

- [103] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, “Bi-layer segmentation of binocular stereo video,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 407–414.
- [104] S. Lim, A. Mittal, L. S. Davis, and N. Paragios, “Fast illumination-invariant background subtraction using two views: Error analysis, sensor placement and applications,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 1071–1078.
- [105] S. Bahadori, L. Iocchi, G. R. Leone, D. Nardi, and L. Scozzafava, “Real-time people localization and tracking through fixed stereo vision,” *Applied Intelligence*, vol. 26, no. 2, pp. 83–97, 2007.
- [106] R. Crabb, C. Tracey, A. Puranik, and J. Davis, “Real-time foreground segmentation via range and color imaging,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2008, pp. 1–5.
- [107] J. Zhu, M. Liao, R. Yang, and Z. Pan, “Joint depth and alpha matte optimization via fusion of stereo and time-of-flight sensor,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 453–460.
- [108] I. Schiller and R. Koch, “Improved video segmentation by adaptive combination of depth keying and mixture-of-gaussians,” in *Scandinavian Conference on Image Analysis*, 2011, pp. 59–68.
- [109] K. Greff, A. Brandão, S. Krauß, D. Stricker, and E. Clua, “A comparison between background subtraction algorithms using a consumer depth camera,” in *International Conference on Computer Vision Theory and Applications*, 2012, pp. 431–436.

- [110] E. J. Fernandez-Sanchez, L. Rubio, J. Díaz, and E. Ros, “Background subtraction model based on color and depth cues,” *Machine Vision and Applications*, vol. 25, no. 5, pp. 1211–1225, 2014.
- [111] J. Murgia, C. Meurie, and Y. Ruichek, “An improved colorimetric invariants and rgb-depth-based codebook model for background subtraction using kinect,” in *Mexican International Conference on Artificial Intelligence*, 2014, pp. 380–392.
- [112] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, “Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 3361–3368.
- [113] J. Nocedal, “Updating quasi-newton matrices with limited storage,” *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.
- [114] J. Ho, K.-C. Lee, M.-H. Yang, and D. J. Kriegman, “Visual tracking using learned linear subspaces,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2004, pp. 782–789.
- [115] A. Doucet, N. De Freitas, and N. Gordon, *An introduction to sequential Monte Carlo methods*. Springer, 2001.
- [116] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.
- [117] S. Baker and I. Matthews, “Lucas-kanade 20 years on: A unifying framework,” *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

- [118] S. Avidan, “Ensemble tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 494–501.
- [119] J. Yang, K. Yu, Y. Gong, and T. S. Huang, “Linear spatial pyramid matching using sparse coding for image classification,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1794–1801.
- [120] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai, “Minimum error bounded efficient ℓ_1 tracker with occlusion detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1257–1264.
- [121] B. Liu, L. Yang, J. Huang, P. Meer, L. Gong, and C. A. Kulikowski, “Robust and fast collaborative tracking with two stage sparse optimization,” in *European Conference on Computer Vision*, 2010, pp. 624–637.
- [122] M. Grabner, H. Grabner, and H. Bischof, “Learning features for tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- [123] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
- [124] S. M. A. Eslami, N. Heess, and J. M. Winn, “The shape boltzmann machine: A strong model of object shape,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 406–413.
- [125] A. Coates, A. Karpathy, and A. Y. Ng, “Emergence of object-selective features in unsupervised feature learning,” in *Annual Conference on Neural Information Processing Systems*, 2012, pp. 2690–2698.
- [126] A. Coates and A. Y. Ng, “Selecting receptive fields in deep networks,” in *Annual Conference on Neural Information Processing Systems*, 2011, pp. 2528–2536.

- [127] Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng, “Ica with reconstruction cost for efficient overcomplete feature learning,” in *Annual Conference on Neural Information Processing Systems*, 2011, pp. 1017–1025.
- [128] Q. V. Le, M. Ranzato, R. Monga, M. Devin, G. Corrado, K. Chen, J. Dean, and A. Y. Ng, “Building high-level features using large scale unsupervised learning,” in *International Conference on Machine Learning*, 2012.
- [129] N. Li and J. J. DiCarlo, “Unsupervised natural experience rapidly alters invariant object representation in visual cortex,” *Science*, vol. 321, no. 5895, pp. 1502–1507, 2008.
- [130] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual category models to new domains,” in *European Conference on Computer Vision*, 2010, pp. 213–226.
- [131] J. Yang, R. Yan, and A. G. Hauptmann, “Cross-domain video concept detection using adaptive svms,” in *ACM International Conference on Multimedia*, 2007, pp. 188–197.
- [132] L. Duan, D. Xu, I. W.-H. Tsang, and J. Luo, “Visual event recognition in videos by learning from web data,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1959–1966.
- [133] X. Glorot, A. Bordes, and Y. Bengio, “Domain adaptation for large-scale sentiment classification: A deep learning approach,” in *International Conference on Machine Learning*, 2011, pp. 513–520.
- [134] Q. Wang, F. Chen, J. Yang, W. Xu, and M.-H. Yang, “Transferring visual prior for online object tracking,” *IEEE Transactions on Image Processing*, vol. 21, no. 7, pp. 3296–3305, 2012.

-
- [135] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.
- [136] Y. Pang and H. Ling, “Finding the best from the second bests - inhibiting subjective bias in evaluation of visual tracking algorithms,” in *IEEE International Conference on Computer Vision*, 2013, pp. 2784–2791.
- [137] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418.
- [138] X. Li, C. Shen, A. R. Dick, and A. van den Hengel, “Learning compact binary codes for visual tracking,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2419–2426.
- [139] S. He, Q. Yang, R. W. H. Lau, J. Wang, and M. Yang, “Visual tracking via locality sensitive histograms,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2427–2434.
- [140] T. B. Dinh, N. Vo, and G. G. Medioni, “Context tracker: Exploring supporters and distracters in unconstrained environments,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1177–1184.
- [141] W. Zhong, H. Lu, and M. Yang, “Robust object tracking via sparse collaborative appearance model,” *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2356–2368, 2014.
- [142] J. Kwon and K. M. Lee, “Tracking by sampling trackers,” in *IEEE International Conference on Computer Vision*, 2011, pp. 1195–1202.

- [143] S. Kong and D. Wang, “A multi-task learning strategy for unsupervised clustering via explicitly separating the commonality,” in *International Conference on Pattern Recognition*, 2012, pp. 771–774.
- [144] H. Li, Y. Li, and F. Porikli, “Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking,” in *British Machine Vision Conference*, 2014.
- [145] L. Wang, T. Liu, G. Wang, K. L. Chan, and Q. Yang, “Video tracking using learned hierarchical features,” *IEEE Transactions on Image Processing*, vol. 24, no. 4, pp. 1424–1435, 2015.
- [146] S. Hong, T. You, S. Kwak, and B. Han, “Online tracking by learning discriminative saliency map with convolutional neural network,” in *International Conference on Machine Learning*, 2015, pp. 597–606.
- [147] L. Wang, W. Ouyang, X. Wang, and H. Lu, “Visual tracking with fully convolutional networks,” in *IEEE International Conference on Computer Vision*, 2015.
- [148] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, “Hierarchical convolutional features for visual tracking,” in *IEEE International Conference on Computer Vision*, 2015.
- [149] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, “Visual tracking: An experimental survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1442–1468, 2014.
- [150] X. Wang, G. Hua, and T. X. Han, “Discriminative tracking by metric learning,” in *European Conference on Computer Vision*, 2010, pp. 200–214.

- [151] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, “Semi-supervised recursive autoencoders for predicting sentiment distributions,” in *Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 151–161.
- [152] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, “Semantic compositionality through recursive matrix-vector spaces,” in *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012, pp. 1201–1211.
- [153] R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning, “Dynamic pooling and unfolding recursive autoencoders for paraphrase detection,” in *Annual Conference on Neural Information Processing Systems*, 2011, pp. 801–809.
- [154] R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning, “Parsing natural scenes and natural language with recursive neural networks,” in *International Conference on Machine Learning*, 2011, pp. 129–136.
- [155] R. Socher, B. Huval, B. P. Bath, C. D. Manning, and A. Y. Ng, “Convolutional-recursive deep learning for 3d object classification,” in *Annual Conference on Neural Information Processing Systems*, 2012, pp. 665–673.
- [156] C. Goller and A. Kuchler, “Learning task-dependent distributed representations by backpropagation through structure,” in *IEEE International Conference on Neural Networks*, vol. 1, 1996, pp. 347–352.
- [157] X. Ren and J. Malik, “Learning a classification model for segmentation,” in *IEEE International Conference on Computer Vision*, 2003, pp. 10–17.

- [158] P. A. Viola, M. J. Jones, and D. Snow, “Detecting pedestrians using patterns of motion and appearance,” in *IEEE International Conference on Computer Vision*, 2003, pp. 734–741.
- [159] B. Wu and R. Nevatia, “Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors,” in *IEEE International Conference on Computer Vision*, 2005, pp. 90–97.
- [160] Z. Lin, L. S. Davis, D. S. Doermann, and D. DeMenthon, “Hierarchical part-template matching for human detection and segmentation,” in *IEEE International Conference on Computer Vision*, 2007, pp. 1–8.
- [161] X. Wang, T. X. Han, and S. Yan, “An hog-lbp human detector with partial occlusion handling,” in *IEEE International Conference on Computer Vision*, 2009, pp. 32–39.
- [162] W. Ouyang and X. Wang, “A discriminative deep model for pedestrian detection with occlusion handling,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3258–3265.
- [163] ———, “Joint deep learning for pedestrian detection,” in *IEEE International Conference on Computer Vision*, 2013, pp. 2056–2063.
- [164] W. Ouyang, X. Zeng, and X. Wang, “Modeling mutual visibility relationship in pedestrian detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3222–3229.
- [165] F. Xu and K. Fujimura, “Human detection using depth and gray images,” in *IEEE Conference on Advanced Video and Signal Based Surveillance*, 2003, pp. 115–121.

- [166] D. M. Gavrila and S. Munder, “Multi-cue pedestrian detection and tracking from a moving vehicle,” *International Journal of Computer Vision*, vol. 73, no. 1, pp. 41–59, 2007.
- [167] S. Ikemura and H. Fujiyoshi, “Real-time human detection using relational depth similarity features,” in *Asian Conference on Computer Vision*, 2010, pp. 25–38.
- [168] L. Spinello and K. O. Arras, “People detection in rgb-d data,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3838–3843.
- [169] W. Choi, C. Pantofaru, and S. Savarese, “Detecting and tracking people using an rgb-d camera via multiple detector fusion,” in *IEEE International Conference on Computer Vision Workshops*, 2011, pp. 1076–1083.
- [170] T. Gill, J. M. Keller, D. T. Anderson, and R. H. L. III, “A system for change detection and human recognition in voxel space using the microsoft kinect sensor,” in *IEEE Applied Imagery Pattern Recognition Workshop*, 2011, pp. 1–8.
- [171] L. Xia, C. C. Chen, and J. K. Aggarwal, “Human detection using depth information by kinect,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2011, pp. 15–22.
- [172] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 2006.
- [173] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

-
- [174] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Annual Conference on Neural Information Processing Systems*, 2001, pp. 849–856.
- [175] F. R. K. Chung, “Spectral graph theory,” in *CBMS Regional Conference Series in Mathematics*, vol. 92, no. 92. American Mathematical Society, 1997, pp. 1–212.
- [176] C. Dicle, O. I. Camps, and M. Sznaiier, “The way they move: Tracking multiple targets with similar appearance,” in *IEEE International Conference on Computer Vision*, 2013, pp. 2304–2311.
- [177] M. Everingham, L. J. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.
- [178] L. Bo, X. Ren, and D. Fox, “Kernel descriptors for visual recognition,” in *Annual Conference on Neural Information Processing Systems*, 2010, pp. 244–252.
- [179] L. Bo, K. Lai, X. Ren, and D. Fox, “Object recognition with hierarchical kernel descriptors,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1729–1736.