Learning graphs for object tracking and counting

Shengkun Li

University at Albany, State University of New York, shengkunliluo@gmail.com

The University at Albany community has made this article openly available. Please share how this access benefits you.

Follow this and additional works at: https://scholarsarchive.library.albany.edu/legacy-etd

Part of the Artificial Intelligence and Robotics Commons

Recommended Citation

https://scholarsarchive.library.albany.edu/legacy-etd/2503

This Dissertation is brought to you for free and open access by the The Graduate School at Scholars Archive. It has been accepted for inclusion in Legacy Theses & Dissertations (2009 - 2024) by an authorized administrator of Scholars Archive.
Please see Terms of Use. For more information, please contact scholarsarchive@albany.edu.
LEARNING GRAPHS FOR OBJECT TRACKING AND COUNTING

by

Shengkun Li

A Dissertation
Submitted to the University at Albany, State University of New York
in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy

College of Engineering and Applied Sciences
Department of Computer Science
Winter 2021
To my beloved family
ABSTRACT

As important problems in computer vision, object tracking and counting attract increasing amounts of attention in recent years due to its wide range of applications, such as video surveillance, human-computer interaction, smart city. Despite much progress has been made in object tracking and counting with the arriving of deep neural networks (DNN), there still remains much room for improvement to satisfy the real-world applications.

Graph is an effective tool to describe the relations among objects. The nodes in the graph are generally used to represent the object and the edges are used to describe their relations. In this dissertation, I investigate to exploit the powerful graph model to solve object tracking and counting tasks in various scenarios.

First, I design a hybrid structure hypergraph to solve the visual object tracking, which use a non-uniform hypergraph to model the dependencies among object parts. Specifically, the tracking task is formulated as the dense structures extracting problem on the non-uniform hypergraph, solved by an approximate algorithm efficiently. In contrast to previous methods rely on sole degree of dependencies (e.g., pairwise or high-order dependencies), I integrate various degree of dependencies of different objects parts in consecutive frames, which is more robust in complex scenarios, especially for the deformable object tracking.

Second, I propose to use the non-uniform hypergraph to solve the multi-object tracking task. The non-uniform hypergraph is constructed to exploit various higher order dependencies among objects, which is more effective in complex scenarios compared to uniform hypergraph. The nodes in the hypergraph correspond to the tracklets and the hyperedges with different degrees encode various kinds of dependencies among them. Instead of setting the weights of hyperedges with different degrees empirically, they are learned automatically using the structural support vector machine (SSVM).

Third, I design a space-time graph convolutional network (STGNet) to handle counting, localization and tracking in crowded scenarios. STGNet is formed by the Siamese feature extraction subnetwork, followed by the density map estimation, localization, and association branches in parallel. The Siamese feature extraction subnetwork computes the correlations between multi-scale
features in two consecutive frames to exploit temporal coherence. The density map estimation branch is used to estimate the density of objects, and the localization branch is used to determine the accurate targets’ locations. Meanwhile, the association branch is introduced to predict the motion offsets of targets for tracking. To exploit the relations of neighboring targets, the graph convolution operations are integrated in the association branch.

Overall, this work is an active attempt to use graph model to handle object tracking and counting in various challenging real-world scenarios.
ACKNOWLEDGMENT

Throughout the writing of this thesis I have received a great deal of support and assistance. I would like to thank the following people, without whom I would not have made it through my degree.

I would like to express my deepest appreciation to my advisor: Professor Ming-Ching Chang for providing me with continuous support, encouragement and patience throughout the duration of the Ph.D study and research. You are so encouraging to me. My journey through Ph.D. program was amazing. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

Meanwhile, I would like to extend my deepest gratitude to my other advisor: Professor Siwei Lyu. I cannot fully express my appreciation for all the guidance and support you have shown me in the past five years. You are so dedicated and passionate about your research work, which is so encouraging to me. Thank you so much for your help and support in last five years.

In addition, I would like to sincerely thank to my other two committee: Professor Pradeep K. Atrey and Professor Yiming Ying, for your valuable advice and insightful suggestions. I also wish to thank Professor Michael Phipps and Professor Shaghayegh Sahebi for helping me build my communication and presentation skills when I am working as a teaching assistant.

I also had great pleasure of working with my lab mates: Dr.Yuezun Li and Lipeng Ke for the stimulating discussions and have been a great source of support in the last five years. I cannot leave UAlbany without mentioning Dr.Dawei Du, who for enlightening me the first glance of research. Thank you for the continuous support of my Ph.D study and related research, for your patience, motivation and extensive knowledge.

Finally, I am extremely grateful to my parents for your love, caring, and sacrifice throughout writing this thesis and my life in general. Last but not least, I am very much thankful to my husband: Longyin and my little boy: Ethan, for your love, understanding, and unconditional continuing support! You all kept me going, and this thesis would not have been possible without you.
# CONTENTS

<table>
<thead>
<tr>
<th>ABSTRACT</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENT</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 The Importance of Object Tracking and Counting</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Research Questions and Outline</td>
<td>2</td>
</tr>
<tr>
<td>1.2.1 Deformable Visual Tracking</td>
<td>2</td>
</tr>
<tr>
<td>1.2.2 Multi-Object Tracking</td>
<td>3</td>
</tr>
<tr>
<td>1.2.3 Crowd Counting, Localization, and Tracking</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Structure of Dissertation</td>
<td>6</td>
</tr>
<tr>
<td>2. Related Work</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Object Detection</td>
<td>8</td>
</tr>
<tr>
<td>2.1.1 Classical Object Detectors</td>
<td>8</td>
</tr>
<tr>
<td>2.1.2 Deep Neural Network based Object Detectors</td>
<td>9</td>
</tr>
<tr>
<td>2.1.3 Joint Detection and Segmentation</td>
<td>10</td>
</tr>
<tr>
<td>2.1.4 Detection for Counting</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Single Object Tracking</td>
<td>12</td>
</tr>
<tr>
<td>2.2.1 Metrics, Benchmarks, and Datasets</td>
<td>13</td>
</tr>
<tr>
<td>2.2.2 Classical SOT methods</td>
<td>14</td>
</tr>
<tr>
<td>2.2.3 Deep learning SOT methods</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Multi-Object Tracking</td>
<td>20</td>
</tr>
<tr>
<td>2.3.1 Metrics, Benchmarks and Datasets</td>
<td>20</td>
</tr>
<tr>
<td>2.3.2 Tracking-by-Detection</td>
<td>22</td>
</tr>
</tbody>
</table>

vi
4.3 Non-uniform Hypergraph

4.3.1 MOT Formulation

4.3.2 Hypergraph Construction

4.3.3 Near-online tracking

4.3.4 Inference

4.3.5 Learning

4.4 Experiments

4.4.1 Evaluation Metrics

4.4.2 Implementation Details

4.4.3 Ablation Study

4.4.4 Results on Multi-Pedestrian Tracking

4.4.5 Results on MOT2016 Benchmark

4.4.6 Multi-Face Tracking

4.4.7 Qualitative Tracking Results

4.4.8 Running Time

4.5 Conclusions

5. Space-Time Graph Convolutional Networks for Crowd Counting, Localization and Tracking

5.1 Motivation

5.2 Related Work

5.3 Space-Time Graph Convolutional Network

5.3.1 Network architecture

5.3.2 Loss Function

5.3.3 Ground-truth generation

5.3.4 Optimization

5.4 Experiment

5.4.1 Implementation details
LIST OF FIGURES

1.1 Tracking a deformable object in a video. The video frame is taken from the Deform-SOT dataset [66] ........................................................... 3

1.2 Multi-object tracking applied in surveillance scenarios. The video frame is taken from the PETS09 S2L3 sequence [72]. ........................................... 5

1.3 An example of crowd counting, localization and tracking in drone-captured scenarios. The video frame is taken from the DroneCrowd dataset [280] ............................................................ 6

3.1 An example to demonstrate the effectiveness of introducing the non-uniform hypergraph in tracking. We compare (a) DGT [33] that considers pairwise dependencies in a graph; (b) SAT [66] that considers fixed dependencies among target parts in a uniform hypergraph and (c) our method that exploits different degree of dependencies in a non-uniform hypergraph. ...................................................... 34

3.2 The success and precision plots over the dataset using OPE. The representative scores for each tracker are reported in the legends. .......................... 38

3.3 The Deform-SOT dataset. ........................................................................ 39

3.4 The tracking performance with different maximal degree of hypergraph. ............. 40

3.5 The success plots of OPE with different attributes over the dataset. The representative scores for each tracker are reported in the legends. .................. 41

3.6 The precision plots of OPE with different attributes over the dataset. The representa-tive scores for each tracker are reported in the legends. ................................ 42

3.7 Result comparison between our method and other state-of-the-art trackers. The video frames are taken from the Deform-SOT dataset [66]. ......................... 43

4.1 (a) Two previous methods using 3-uniform hypergraph H^2T [282] and FH^2T [281], often fails to describe the dependencies among tracklets, when occlusion or missed detection happen. (b) The proposed method uses the non-uniform hypergraph to encode different degrees of dependencies among tracklets effectively. ................................. 46

4.2 Tracking results of the proposed NT tracker in multi-pedestrian tracking dataset. The video frames are taken from the PETS09 S2L1, S2L2, S2L3, S1L1-1, S1L1-2 [72] and ParkingLot [312] sequences. ................................................. 65
4.3 Tracking results of the proposed NT tracker in multi-face tracking dataset. The video frames are taken from the SubwayFaces dataset [281].

4.4 Tracking results of the proposed NT tracker in the test set of MOT2016 benchmark. The video frames are taken from the MOT2016 benchmark [192].

5.1 The network architecture of the proposed STGNet method. The red circles denote the combined feature maps, i.e., from the stage 1 to stage 3, extracted from the VGG-16 network. The green and purple rectangles indicate the counting, localization and association subnets, respectively. Besides, “DConv” and “Deconv” indicate the deformable convolution and transpose convolution operations.

5.2 The counting and localization subnets of our STGNet. The blue rectangles denote the graph convolution operation.

5.3 The association subnet of our STGNet. The blue rectangles denote the graph convolution operation.

5.4 Qualitative results of our STGNet. The 1st column is the ground-truth tracking trajectories, the 2nd column is the predicted target points, and the 3rd column is the corresponding trajectories. The video frames are taken from the DroneCrowd dataset [280]. Best view in color version.
CHAPTER 1

Introduction

Computer vision aims to make computer to gain high-level understanding from digital images and videos, which is a hot research topic in recent years with various applications, such as intelligent agriculture, smart healthcare, and smart city. Specifically, computer vision has been widely used to support various developments of farming automation in agriculture, e.g., pig counting and weight estimation [209, 39] and cattle gait tacking [86]. For smart healthcare, computer vision is effective to help the diagnosis of lung cancer and skin cancer. The accuracy of the diagnosis results exceeds that of human doctors in some scenarios. In addition, computer vision also plays the critical role in smart city. For example, computer vision is used to monitor the traffic flows to optimize the schedules of traffic lights, and used to analyze image or video data captured by surveillance cameras for security and safety.

1.1 The Importance of Object Tracking and Counting

As the fundamental tasks in computer vision, object tracking and counting attracts much attention in both academy and industrial fields. Object tracking requires the algorithms to recover the trajectories of objects in video sequences, which is the basic step for various high-level applications, e.g., video understanding and editing. The research in the object tracking field can be roughly divided into two categories, i.e., visual tracking, and multi-object tracking. Visual tracking is also known as model-free tracking, which aims to locate the bounding boxes of target in subsequent video frames with the given initial bounding box of the target in the first frame. Meanwhile, the multi-object tracking aims to track multiple objects, i.e., localize object instances in each video frame with the target identities. In contrast to visual tracking, multi-object tracking methods are required to recover trajectories of the predefined categories of targets, such as pedestrian and vehicles in video sequences.

Object counting aims to estimate the number of objects, e.g., people or vehicle, in images or videos, with a wide range of applications, such as video surveillance [37], crowd analysis [327], public safety [195] and urban planning. From example, we should count the crowds in forbidden
areas in a manufacturing unit to enforce safety rules and minimize health risks. In recent years, many massive stampedes have taken place around the world that claimed many victims, further making the automatic density map estimation, counting and localization in crowds to be the urgent need in some scenarios. Although much progress has been achieved in object tracking and counting fields since the advent of deep learning, there still remains much room for improvement for object tracking and counting algorithms, especially in crowded scenes.

1.2 Research Questions and Outline

To advance the developments of object tracking and counting, we attempt to exploit the context information to improve the robustness. Graph is an effective tool to describe the relations among different objects. The nodes in the graph are generally used to represent the object and the edges are used to describe their relations. Thus, I investigate to use the powerful graph model to exploit the context information for object tracking and counting tasks in various scenarios.

1.2.1 Deformable Visual Tracking

Although significant progress has been achieved in recent years, tracking a deformable object remains a challenging task due to large changes of targets in appearance caused by deformation, occlusions, and clutter background (see Figure 1.1). Most of previous methods focus on modeling the appearance variations of targets in the bounding box, e.g., correlation filters \[57, 104\], sparse representation \[188, 116\], online boosting \[10\], deep neural network \[184, 197, 255\], which suffer from the drift problem due to considerable pixels of the background are encompassed in the bounding box, especially when large deformation and occlusion exist. To that end, Cai et al. \[33\] use the over-segmented superpixels to replace the bounding box based parts and formulate the deformable object tracking task as tracking the dynamic undirected geometric structure graph of the target. Du et al. \[66\] improve the method \[33\] by using hypergraph instead of graph to capture the high-order interactions among target parts. Specifically, the geometric hypergraph is constructed and learned to match the target parts and the candidate parts in two consecutive frames. However, the dependencies of different object parts in consecutive frames are different with each other, especially when large deformation and occlusion happen. Using the uniform hypergraph is less effective in handling deformation and occlusion challenges in unconstrained scenes.
To solve the deformable object tracking task, I design a hybrid structure hypergraph based tracker, which uses non-uniform hypergraph to describe the dependencies among target parts in consecutive frames. Specifically, the SLIC over-segmentation algorithm \cite{3} is used to generate superpixels in each video frame and apply the graph cut algorithm \cite{27} to produce the candidate parts. Then, a non-uniform hypergraph is constructed to capture the hybrid dependencies among candidate parts across multiple frames, where each node corresponds to a candidate part, and each edge encodes the dependencies of the parts, \textit{i.e.}, the consistencies in both appearance and motion. Inspired by \cite{283,66}, an approximate algorithm is proposed to extract the dense structures on the hypergraph to decide the parts belonging to the target. After that, the target state (\textit{i.e.}, center location in pixel and scale) is determined by analyzing the extracted parts belonging to the target. Several experiments are carried out on publicly available Deform-SOT dataset, to demonstrate the favorable performance of the proposed method against state-of-the-art trackers.

1.2.2 Multi-Object Tracking

Besides visual tracking, I also investigate to use hypergraph to solve the multi-object tracking task. Generally, an automatic MOT system usually employs a pretrained object detector to

\footnote{Here, the terminology “edge” is used to indicate the self-loop, conventional edge and hyperedge, and the terminology “hyperedge” is used to indicate the edge involving more than two nodes specifically.}
locate candidate object regions in each frame, then match the detections across frames to form target trajectories. Most existing methods only consider the pairwise dependencies of detections (e.g., 224, 61, 194, 73), and do not take full advantage of the high-order dependencies among multiple targets across frames. This strategy is less effective when nearby objects with similar appearance or motion patterns occlude each other in the video. Several recent methods [125, 45, 239, 125, 282, 281] attempt to use the high-order information to improve the tracking performance, such as dense structure search on hypergraph [282, 281], tensor power iterations [239], high-order motion constraints [45, 31], and multiple hypothesis tracking [125]. However, the aforementioned methods merely exploit fixed degrees of dependencies among objects, which limits the flexibility of the hypergraph model in complex environments, and calls for adaptive dependency patterns.

To that end, I propose a non-uniform hypergraph learning based tracker (NT), which has much stronger descriptive power to accommodate different tracking scenarios (see Figure 1.2) than the conventional graph [61] or uniform hypergraph [282, 281]. The nodes in the hypergraph correspond to the tracklets and the hyperedges with different degrees encode similarities among tracklets to assemble various kinds of appearance and motion patterns. The tracking problem is formulated as searching dense structures on the non-uniform hypergraph. Different from previous methods [282, 281], the degree of the hypergraph model is not fixed. The hyperedges of different degrees are mixed and the relative weights are learned automatically from the data using the structural support vector machine (SSVM) method [120]. An efficient approximation algorithm is designed to exploit the dense structures to form long object trajectories to complete the tracking task. In addition, to achieve both accuracy and efficiency, a near-online strategy is used for MOT, i.e., the dense structure searching is conducted on the non-uniform hypergraph to generate short tracklets in a temporal window, and then those short tracklets are associated to the tracked targets to get the final trajectories of targets at the current time stamp. This process is carried out repeatedly to complete the tracking task in a video.

2A hypergraph is a generalization of a conventional graph where an edge can join more than two nodes.
3The terminology “tracklet” indicates a fragment of target trajectory. Notably, the input detection responses in each frame can be treated as tracklets of length one.
1.2.3 Crowd Counting, Localization, and Tracking

In crowd scenes, heavy occlusions, large-scale variation, and perspective changes challenge the performance of existing object detection and tracking methods. In terms of crowd counting in videos, spatio-temporal context information is critical to improve the counting accuracy. Xiong et al. [295] design a convolutional LSTM model to fully capture both spatial and temporal dependencies for crowd counting. Zhang et al. [318] combine fully convolutional neural networks and LSTM by residual learning to perform vehicle counting.

In contrast to the aforementioned methods, as shown in Figure 1.3, I attempt to go a step further to solve the crowd counting, localization and tracking tasks jointly using a feed-forward neural network. That is, I design a space-time graph convolutional network (STGNet), which combines multi-scale feature maps in sequential frames and outputs the enhanced features by deformable convolution. The proposed method is effective in exploiting the temporal coherency across frames for better performance. The proposed STGNet method is formed by four modules, i.e., the Siamese feature extraction subnetwork, the density map estimation branch, the localization branch, and the
association branch. The Siamese feature extraction subnetwork first uses two-branch convolutional neural networks (CNNs) to extract multi-scale features, and then computes the correlations between the extracted features in consecutive two frames. The density map estimation branch is used to estimate the density of objects in video frames, and the localization branch is used to determine the accurate locations of targets. Meanwhile, I introduce the association branch to predict the motion offsets of targets for tracking. To exploit the relations of neighboring targets, the graph convolution operations are adopted instead of conventional convolution operations in the association branch. The whole network is trained in an end-to-end manner with the multi-task loss and Adam optimizer [126]. After that, the non-maximal suppression followed by the min-cost flow method [212] is used to generate long trajectories of targets. Extensive experiments are carried out on several datasets including Shanghaitech [321] and UCF-QNRF [114], and DroneCrowd [280] to demonstrate the effectiveness of the proposed method.

1.3 Structure of Dissertation

The remainder of this dissertation is summarized as follows. In Chapter 2, the related work in object detection, visual tracking, crowd counting and graph framework is reviewed. We also introduce the corresponding datasets and evaluation metrics. In Chapter 3, we present the hybrid structure hypergraph method for online deformable object tracking. In Chapter 4, the non-uniform
hypergraph framework is learned for multi-object tracking. In Chapter 5, we describe the space-time graph convolutional networks for crowd counting, localization and tracking. Finally, we summarize the works in this dissertation and the future works in Chapter 6.
CHAPTER 2

Related Work

In this chapter, I would like to first give a brief literature review in object detection, which is the foundation of other tasks such as single object tracking, multi-object tracking, and crowded counting. Then, we discuss the aforementioned topics in our paper in terms of algorithms and datasets. In addition, the graph based frameworks and the applications are introduced.

2.1 Object Detection

Object detection has attracted much attention in recent years because it is the foundation of many high-level computer vision tasks, such as visual tracking, object segmentation, and crowd counting. Given an image, object detection aims to predict a series of bounding boxes with the class labels. Early methods \cite{261, 317, 78} rely on hand-crafted features and classifier to scan the image using the sliding window strategy. With the arriving of deep neural network, object detection is quickly dominated by the deep convolution neural network based methods \cite{91, 221, 176}. In the following sections, I would like to discuss the classical object detection methods in Section 2.1.1 and the deep convolution neural network based methods including the anchor-based approach and the anchor-free approach in Section 2.1.2. Finally, the detection methods combined with segmentation and counting are discussed in Section 2.1.3 and 2.1.4 respectively.

2.1.1 Classical Object Detectors

As mentioned above, early object detection algorithms use the sliding-window mechanism with the hand-crafted features and classifiers applied on dense image grids to detect objects. Viola and Jones \cite{261} design the face detector using the Haar-like features and adaptive boosting to train a cascade classifier. Another popular hand-craft feature is local binary pattern (LBP) in object detection, which uses diverse local structures of images to represent the object. Zhang et al. \cite{317} design the discriminative multi-block local binary pattern (MB-LBP) to describe faces. Meanwhile, histogram of oriented gradients (HOG) \cite{50} with optical flows (HOF) \cite{51} is also effective in object detection, which uses gradient to describe the local dominant edge information for
pedestrian detection. DPM \cite{78} design the mixtures of multi-scale deformable part model using the structural SVM method to detect multi-class objects. Later in 2014, Yang et al.\cite{300} introduce a variant of channel features called aggregate channel features, extending the image channel to various types of information such as magnitude and oriented gradient histograms to encode rich information in a simple form. In \cite{65}, Dollar et al. propose to extract features from multiple channels, which achieves significant performance improvements.

### 2.1.2 Deep Neural Network based Object Detectors

In recent years, object detection has achieved significant improvement with the development of deep neural networks. Anchor-based approach and anchor-free approach are two popular pipelines for object detection, which are described in detail as follows.

**Anchor-based methods** In anchor-based algorithms, a set of rectangles with pre-defined sizes are placed first, and then these bounding boxes are regressed to the desired position according to ground-truth boxes. Generally speaking, anchor-based methods can be roughly grouped into the two-stage and one-stage methods.

Two-stage methods usually divide the object detection task into two steps: first extracting candidate object proposals, and then classifying and regressing these proposed regions. In R-CNN method \cite{91}, the selective search method is used for locating Region of Interests (RoIs). Then, a CNN-based region-wise classifier is applied to classify the RoIs independently. Later, SPPNet \cite{102} and Fast-RCNN \cite{90} extent R-CNN by using RoI pool operation on feature maps to obtain RoIs. Ren et al.\cite{221} introduce the Region Proposal Network (RPN) to generate proposals from features and design the Faster-RCNN detector. RoIs are generated by regressing the anchor boxes in RPN. Afterwords, the RPN is widely used in the two-stage object detection task. In order to generate high-level semantic feature maps at all scales, a top-down architecture with lateral connections designed in Feature Pyramid Network (FPN) \cite{170}. Mask-RCNN \cite{101} integrates a mask prediction branch into the Faster-RCNN framework, so that objects detection and masks prediction can be accomplished at the same time. Cascade R-CNN \cite{32} trains a set of detectors with increasing IoU thresholds for the purpose of dealing with over-fitting problem in the training process and quality mismatch at inference-time. Recently, EfficientDet \cite{252} designs a weighted bi-directional feature pyramid network (BiFPN) for easy and fast multi-scale feature fusion.

Compared to the two-stage methods, the one-stage approaches remove the RoI extraction
process and detect objects directly by regular and dense sampling over locations, scales and aspect ratios, resulting in better efficiency. SSD [176] spreads out anchor boxes of different scales densely to multiple convolutional layers, and each layer is enforced to focus on predicting objects with the certain scale. Improved from SSD [176], DSSD algorithm [82] merges a deconvolution module into SSD to integrate both low-level and high-level features for more accurate results. Following with SSD structure, DSOD [238] introduces an efficient framework to learn object detectors from scratch. YOLO [218] is another popular one-stage detector. In order to achieve high efficiency, the input image are divided into several grids. After that, several anchors are paved for regression and classification using a feed-forward convolutional network. Then, YOLOv2 [219] is designed to improve YOLO by integrating anchor boxes, adding batch normalization, using high resolution classifier, and applying convolution layers for prediction.

To combine the advantages of both one-stage and two-stage approaches, Zhang et al. [319] propose the anchor refinement module and the object detection module, where negative anchors can be filtered out in the first module and the locations and sizes of the anchors is refined twice in the whole detecting process.

**Anchor-free methods** In recent years, anchor-free detectors have attracted increasing interest from researchers because they represent the objects by eliminating the predefined set of anchor boxes. Thus the detectors can be transferred to different scenarios with different scales of objects. Corner-Net [144] is the pioneer work that uses a single convolution neural network to predict a heatmap for the top-left and bottom right corners of all instances of the same object with embedding vector for each detected corner. Moreover, Duan et al. [70] use a triplet instead of a pair of keypoints to represent each object, i.e., top-left corners, bottom-right corners and center points. Besides, Tian et al. [257] propose a fully convolutional one-stage object detector to perform object detection in a per-pixel prediction fashion.

### 2.1.3 Joint Detection and Segmentation

Object detection frameworks can be naturally extended to semantic/instance segmentation by adding segmentation heads. Therefore, many researchers devote their efforts to completing object detection and segmentation jointly. Yao et al. [305] apply a graphical model to present a holistic approach to scene understanding which reasons jointly about object detection, classification and segmentation. Pham et al. [210] propose a framework (BiSeg) for simultaneous semantic
segmentation and instance segmentation with Fully Convolutional Networks, where sub-networks including region proposal, bounding box regression, semantic segmentation and instance segmentation heads share convolutional features. Similarly, Multinet [256] proposed an unified network for joint classification, detection and semantic segmentation by sharing the encoder sub-network. Sistu et al. [246] also use a shared network to capture image cues in the encoding step and different sub-networks for prediction of object bounding boxes and segmentation maps in the decoding step. BlitzNet [71] is introduced for completing object detection and semantic segmentation task jointly, allowing real-time computations. Later, TripleNet [34] is proposed to deeply join the object detection task and segmentation task at different scales, where triple supervisions including detection-oriented supervision and class-aware/agnostic segmentation supervisions are imposed on each layer of the decoder.

Recently, panoptic segmentation [128] is further proposed to unify the typically distinct tasks of semantic segmentation and instance segmentation. Li et al. [160] propose a novel pipeline using segmentation and localisation cues to predict a coherent panoptic segmentation in an end-to-end manner. Different from a sliding window detector that densely enumerates object proposals, Wang et al. [266] propose a Pixel Consensus Voting (PCV) approach to detect instances as a result of the consensus among pixel-wise votes.

2.1.4 Detection for Counting

Object detection methods can also be applied in object counting, especially for small objects in crowded scenes. This task is quite difficult for object detection algorithms due to the limited and distorted information that small region of interests contain. Since small objects do not contain detailed information, many algorithms feed high-resolution images with more details into a network in order to deal with this issue. Haris et al. [99] introduce an end-to-end model that jointly trains super-resolution and detection models in a deep neural network, where the super-resolution sub-network explicitly incorporates a detection loss in its training objective. Instead of super resolving the whole images, SOD-MTGAN [13] pools RoIs first and then train the super-resolution model using those pooled RoIs.

Moreover, some researchers focus on generating high-resolution features in order to improve detection performance. Liu et al. [179] design a High-resolution Detection Network (HRD-Net) including Multi-Depth Image Pyramid Network (MD-IPN) and Multi-Scale Feature Pyra-
mid Network (MS-FPN), which is used to balance the performance among objects with different scales. Noh et al. [200] propose a feature-level super-resolution approach by utilizing proper high-resolution target features as supervision signals for training of a super-resolution model and employing atrous convolution layers to match the relative receptive fields between high and low-resolution features. Perceptual GAN [156] generates super-resolved features and stacked them into feature maps of small objects to enhance the feature representations.

Many studies have empirically proved that the context information also helps detect small objects. Singh et al. [245] process context regions around ground-truth instances at the appropriate scale in order to decrease the influence of background. Hu and Ramanan [107] make use of multi-layer convolutional deep to capture both high-resolution detail and coarse low-resolution cues across large receptive field by extracting surrounding regions along with RoIs to detect human faces. In the works of [170, 292], context features are extracted from multiple layers. Lastly, some studies [96, 309, 323] employ atrous convolution layers for better segmenting small objects due to larger receptive fields without losing resolution.

### 2.2 Single Object Tracking

Single Object Tracking (SOT) plays an important role with various applications including video understanding, automatic driving, surveillance system, human-computer interaction, and robot sensing. In this task, a target within a bounding box is given in the first video frame, then the goal is to find the same target in following frames in order to generate object trajectory. It is very challenging due to several factors such as complex background, object deformations, motions and occlusion, appearance variations, and real-time requirement. Notably, this section focuses on single object tracking methods in general cases and not on trackers designed for special cases, for example, 3D tracking [159, 241, 158], and RGB-T tracking [263, 142, 333].

In general, SOT task could be formed into four basic steps, 1) generate multiple proposal regions in the next frame when target is given in current frame; 2) extract features from target and proposal regions; 3) evaluate proposal regions, by comparing to target or previous tracking output; 4) output the best candidate region as final result and move to the next frame. Numerous methods have been proposed in object tracking. Different methods focus on different questions according to each step as described above: How to obtain proposal regions? Which features should be used to represent target? How to extract features? How to evaluate proposal regions? When to update
target template or algorithms? How to select or generate the best candidate region? A large number of proposed object tracking algorithms aim to look into these questions in different scenarios.

There are multiple ways to categorize object tracking methods. Traditional tracking algorithms use hand-craft features to represent object and apply different observation model to evaluate candidate regions. In recent years, deep learning models can generate much more discriminative representation. The remaining sections are organized in this manner. We first describe the most commonly used metrics and datasets in SOT in Section 2.2.1. Then traditional SOT methods and deep learning SOT methods are reviewed in Section 2.2.2 and 2.2.3 respectively.

2.2.1 Metrics, Benchmarks, and Datasets

In the past few years, a number of SOT benchmarks have been published, such as OTB50 [290], OTB100 [291], and VOT2015 [133].

**OTB** OTB50 [290] contains 50 challenging videos with substantial variations. OTB100 [291] expands OTB50 and has 100 fully annotated video sequences with 590 frames average length. In particular, OTB100 contains quite humans related sequences (36 body and 26 face/head videos) and contains twenty-five percents of gray scale sequences. The test sequences are manually tagged with eleven attributes to represent the challenging aspects, including illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutters and low resolution.

In these two benchmarks, a precision score and an area under curve (AUC) of success plot are used as two evaluation metrics. The precision score is the percentage of frames in which the estimated locations are within a given threshold distance of the ground-truth positions. The success plot shows the ratios of successfully tracked frames with sampled overlap thresholds.

**VOT** VOT is one of the most commonly used benchmarks in visual object tracking field. Each frame in the videos is labeled with five major visual attributes including occlusion, illumination change, motion change, size change, and camera motion. VOT2013 [135] contains 16 short sequences showing various objects in challenging backgrounds, where axis-aligned bounding boxes are used to annotate target in sequences. VOT2014 [134] has 25 short sequences including 8 sequences from VOT2013 and labelled with rotating bounding boxes. VOT2015 [133] contains 60 video sequences. VOT2016 [132] keeps all 60 sequences from VOT2015 but objects are re-

Different from OTB, Accuracy (A) and robustness (R) are the two basic measures. The accuracy value is defined as the overlap between the estimated bounding box and the ground truth bounding box. The robustness is measured by the failure rate measure, which counts the number of times the tracker drifted from the target. Besides, expected average overlap (EAO) is introduced to estimate the average overlap a tracker is expected to attain on a large collection of short-term sequences with the same visual properties.

**GOT-10K** GOT-10K [110] is a recently released large-scale and high-diversity benchmark for generic object tracking in real world. It contains more than 10,000 video segments and populates 563 object classes and 87 motion classes in total. Each video is attached with two semantic labels: object and motion classes. There are five object classes to cover the majority of both natural and artificial moving objects including animal, vehicle, person, passive motion object and object part.

**LaSOT** The LaSOT dataset [74] contains more than 3.52 million manually annotated frames and 1,400 video sequences. It contains 70 classes and each class includes 20 tracking sequences. The average sequence length is 2,500 frames, demonstrating long-term performance of the evaluated trackers. Note that normalized precision plots, precision plots and success plots in one-pass evaluation (OPE) are considered as primary evaluation metrics.

**Other datasets** Besides the above mentioned datasets, there are some less frequently used ones. For example, UAV123 and UAV20L [196] are utilized for unmanned aerial vehicle (UAV) tracking, containing 123 short and 20 long video sequences, respectively. NfS [83] consists of 100 sequences with a high frame rate of 240 fps, for the purpose of analyzing appearance variations effects on tracking performance. UAVDT [68, 310] includes 50 sequences with the vehicle targets in various complex drone-captured scenarios.

### 2.2.2 Classical SOT methods

In a video, the appearance of the target usually changes dramatically in the movement. On one hand, in order to obtain favourable tracking results, many researchers pay close attention to
how trackers can learn target appearance variation and severe deformation quickly and be updated when needed in the tracking process. On the other hand, trackers also need to avoid incorrect on-line updates caused by misleading information in purpose to reduce tracking failure. The tracking algorithms can be generally classified into generative model and discriminative model. Generative model formulates tracking problem as a searching question. Generative methods usually use appearance model to find the best candidate region which has the largest similarity with target template. Discriminative model formulates tracking problem as a classification problem. Discriminative methods firstly train a classifier to distinguish target from background and select a proposal region with highest confidence score as final result.

**Generative methods.** Black and Jepson [25] use a pre-trained subspace model to describe appearance information. However, the target appearance model will not be updated in the tracking process, resulting in failures in handling severe deformation and appearance variation. After that, Ross et al. [228] propose an incremental subspace method. Based on this model, many researchers came up some ideas to improve tracking results [213, 190, 163, 108]. However, when partial occlusion happens, these trackers would easily drift to similar objects in the background. To deal with occlusion, some other methods [188, 189, 320, 116, 264, 325] use sparse representation to model target’s appearance variation. For example, Mei and Ling [188] use target templates from different frames and trivial templates as dictionary to represent appearance information and predict target location by solving an $\ell_1$-regularized least squares problem. For the purpose of handling occlusion and accelerating tracking speed, local sparse and sparse collaborative mode [16, 171, 320, 325, 116] has been introduced to object tracking field. Bao et al. [16] apply accelerated proximal gradient method to solve the $\ell_1$-regularized least squares problem more efficiently. Liu et al. [171] use local sparse representation model to describe target appearance information and locate target location by using mean shift algorithm. Zhang et al. [320] formulate object tracking in a particle filter framework as a multi-task sparse learning problem under the assumption that each particle can be represented by dictionary templates. Zhong et al. [325] propose a collaborative tracking method by joining sparse generative model with sparse discriminative model. In order to handle object appearance variation Jia et al. [116] apply alignment pooling method to obtain sparse code which is used to model target appearance model. Besides sparse representation, FragTrack algorithm [4] uses multiple histograms of multiple rectangular sub target regions to describe target appearance information, resulting in more robustness for occlusion. However, target integral histogram will not be updated in the tracking process, which would lead to target drifting when object
severe deformation happens. For the purpose of tracking accurately under drastic object appearance changing scenarios, Kwon and Lee [140] develop a local patch-based appearance model and provide an efficient scheme to evolve the topology between local patches by online update.

**Discriminative methods** In terms of discriminative methods, Avidan [8] fuse together an optic-flow-based tracker and an Support Vector Machine (SVM) classifier. Then, he proposes an Ensemble Tracking method [9] by formulating tracking task as a two-class classification problem, and uses Adaboost algorithm to integrate multiple weak classifiers into a strong classifier to distinguish target from background. In 2006, Grabner and Bischof [93] introduced an online boosting based feature selection framework instead of offline training in traditional boosting approaches. Because tracker’s updates are based on tracking result in the current frame and more and more noise information will be brought in in the tracking process, object drift would become a side effect. To solve this issue, in 2008, Grabner et al. [94, 248] improved boosting updating methods and gained more robustness by using semi-supervised learning method to restrict the negative effect from background noise information. Kalal et al. [1] apply positive and negative structural constraints in training classifier in order to improve tracker’s discriminability. Hare et al. [98] use Structured SVM Model to complete tracking task.

On the other hand, some trackers use different features to ensure good tracking results. Comaniciu et al. [46] propose mean shift algorithm to perform optimization based on feature histogram target representation. Collins et al. [44] apply mean shift method to take care of object scale changes. In [208], the color histogram is introduced in the particle-filter-based tracking framework [115]. Birchfield and Rangarajan [24] use Spatiogram to capture the values of the pixels and their spatial relationships in order to improve tracking results. Besides, a locality sensitive histogram algorithm [103] is used to taking into account contributions from every pixel adaptively and could describe target appearance information more accurately. In order to obtain edge direction information of the target, Histograms of Oriented Gradients [50] and Integral Histogram [214] are used to describe object appearance. For the purpose of fusing multiple types of features, Tuzel et al. [258] use Covariance Region Descriptors on target description, where spatial information, statistical information and spatial relations could be represented in this framework. Moreover, binary pattern [202] and Haar-like [260] feature are also applied in tracking methods.

**Correlation filtering** In recent years, correlation filter (CF) has been drawn much attention in object tracking field because of its high accuracy and efficiency. MOSSE tracker [26] uses adaptive
correlation filters to model target appearance and finds a filter that minimizes the sum of squared error. Since then, a number of tracking algorithms are proposed on the basis of MOSSE. In order to solve redundant training data issue, Henriques et al. [105] use dense sampling and derive closed-form solutions for training and detection with several types of kernels. Then, they improve the previous work by adding a new variant of the kernelized correlation filter tracker based on HOG features instead of raw pixels in KCF tracker [104]. Beside HOG features, Danelljan et al. [58] propose a low-dimensional color name feature to achieve computationally efficiency.

Since fixed template size in kernel may result in tracking drifting when target size expands, the previous CF methods are not able to handle large target scale variation effectively. To solve this problem, Li and Zhu [164] propose a scale adaptive scheme by sampling the target with different scales and resizing the samples into a fixed size. DSST [55] separates correlation filters for translation and scale estimation. To avoid unwanted boundary effects, SRDCF tracker [57] introduces a spatial regularization component in the learning to penalize correlation filter coefficients depending on their spatial location. Besides, Galoogahi et al. [85, 84] use ADMM optimization process and reorganize previous correlation filter objective used in CSK method [105]. A masking matrix is introduced to extract object region in order to increase in the proportion of examples unaffected by boundary effects. Wang et al. [269] verify tracking confidence to avoid the model corruption problem and apply a multi-modal target detection technique to improve the target localization precision. Recently, more researchers [84, 112] put effort into reducing boundary effects by imposing restrictions in learning process or mitigating filter degradation [269]. Galoogahi et al. [84] learn trackers from negative background patches, instead of shifted foreground patches. Huang et al. [112] use response maps generated in the detection phase to form the restrictions in learning.

2.2.3 Deep learning SOT methods

With the breakthrough of deep neural network, object tracking field has achieved significant progresses over the past few years. Convolutional neural network (CNN) provides a powerful tool with the ability to learn strong deep representations, and recent work attempted to incorporate the correlation filter framework with such features learning capability. Moreover, some tracking algorithms learn complex deep trackers or introduce Siamese network to completing tracking task.
Deep correlation filtering Deep neural network’s strong representational power makes them good at extracting meaningful high-level features. Researchers apply CNNs features into correlation filter framework in order to improve tracking quality. HCF tracker [184] and DeepSRDCF tracker [56] first apply features extracted from deep CNN in discriminative correlation filter based tracking frameworks for the purpose of improving tracking accuracy. Then, C-COT tracker [59] learns a set of convolution filters to produce a continuous-domain confidence map of the target, which integrates multi-resolution feature maps in a joint learning formulation and enables accurate sub-pixel localization. ECO tracker [52] could be regarded as an advanced version of C-COT, which introduces a factorized convolution operator in order to reducing the number of parameters from D to c dimensions in the DCF model, bring in a compact generative model of the training sample space to reduce sample amounts. To alleviate boundary effect or overfitting issue on DCF, Li et al. [153] and Dai et al. [49] incorporate both spatial and temporal regularization into correlation filters. Sun et al. [249] enforce additional constraints on the learned filter weights to form the ROI pooled correlation filter (RPCF) algorithm. Xu et al. [297] introduce a temporal smoothness regularisation term by applying an efficient low-rank approximation to adaptively integrate historical information, and reduce the dimensionality across both spatial and channel dimensions by the use of group feature selection method. While aforementioned algorithms only use deep neural network in partial tracking process, a series of trackers [53, 23, 54] extend existing online discriminative framework with deep networks for end-to-end learning. ATOM [53] method consists of dedicated target estimation and classification component. In DiMP algorithm [23], the object is coarsely localized for distinguishing the target from background in the first stage, and a separate network branch is employed for regressing the target bounding box in the second stage. Then, PrDiMP [54] improves DiMP by minimizing KL-divergence loss to calculate conditional density instead of calculating confidence score directly, which is capable of modeling label noise stemming from inaccurate annotations and ambiguities in tracking tasks.

Siamese neural network Some trackers formulate visual tracking task as a cross-correlation problem and learn a tracking similarity map from a Siamese network structure, where one branch for learning the feature presentation of the target, and the other one for the search area. SiamFC tracker [21] first introduces Siamese networks for visual object tracking. This tracker is quite efficient because tracking model and target template is fixed during the tracking process. Guo et al. [95] propose to learn a dynamic Siamese network, which enables effective online learning of target appearance variation and background suppression from previous frames with acceptable
trade off of speed. Motivated by object detection frameworks, SiamRPN tracker \cite{152} introduces a regional proposal network after the Siamese network and combines classification and regression for tracking. Zhu et al. \cite{334} extend the SiamRPN by developing distractor-aware training in DaSiamRPN tracker. It increases the hard negative training data and the varieties of positive training data during the training phase in order to improve tracking quality. Wang et al. \cite{265} propose a series-parallel matching framework to improve the robustness and discrimination power of SiamRPN. C-RPN method \cite{75} is developed to progressively refine the location of target with a sequence of RPNs (region proposal networks) cascaded from deep high-level to shallow low-level layers in a Siamese network for the purpose of handling similar distraction and large scale variation. The SiamRPN++ algorithm \cite{151} adopts a depth-wise cross correlation to replace the up-channel cross correlation and aggregates multi-layer features to predict the target more accurately. SiamRPN++ allows Siamese-based trackers to explore deeper network, reduces the number of parameters and accelerates the training process. Similarly, Zhang and Peng \cite{322} design new residual modules and architectures that allow deeper and wider backbone networks to unleash power in Siamese trackers.

On the other hand, Wang et al. \cite{271} develop a SiamMask that incorporates instance segmentation into tracking. Since one similarly map is not able to provide sufficient information, He et al. \cite{100} build a two-fold Siamese network with a semantic branch and an appearance branch. The two similarity learning Siamese network branches are trained separately to keep the heterogeneity of features but combined at the testing time to improve the tracking accuracy. FlowTrack \cite{335} exploits motion information by using flow information in consecutive frames in Siamese networks to improve the feature representation. Besides, some tracking methods \cite{311, 69} introduce the attention mechanism into Siamese network to enhance the feature learning capability of Siamese-based trackers. Yu et al. \cite{311} compute deformable self-attention and cross-attention jointly, then compute depth-wise cross correlations between the attention features in order to improve tracker’s discriminability. Du et al. \cite{69} uses a pixel-wise correlation-guided spatial attention module and a channel-wise correlation-guided channel attention module to exploit the relationship between the template and the RoI (region of interest) for improved accuracy in corner detection.

**Convolutional neural network** Besides CNN-based correlation filter tracker and Siamese-based tracker, other researchers propose complex deep trackers directly. Nam and Han \cite{197} design the Multi-Domain Network (MDNet) to learn the shared representation of targets from multiple anno-
tated video sequences for tracking, so that domain-independent information can be separated from domain-specific one. Jung et al. [121] accelerate MDNet method by applying RoIAlign technique to extract more accurate representations of targets and candidates from a feature map and improve target localization. Instead of using CNN as a feature extractor, Wang et al. [267] design a fully CNN to generate object tracking. Specifically, top convolutional layers and lower convolutional layers are jointly used to describe semantic feature and discriminative information respectively within a switch mechanism during tracking process.

2.3 Multi-Object Tracking

Multi-object tracking (MOT) plays an important role in computer vision field with many applications, such as surveillance, autonomous driving, and crowd behaviour analysis. MOT task aims to identify and track all objects in a video sequence. Except dealing with targets with scale variation, deformation and illumination changes in SOT, MOT focuses on determining the number of objects and maintaining object’s ID under frequent occlusions and interactions. To this end, early multi-object tracking algorithms formulate MOT task as a state estimation problem by applying Kalman filters [149, 207] and particle filters [115, 124, 191]. These trackers typically estimate objects states in a short duration effectively but are not able to perform well in complex scenarios.

In recent years, tracking-by-detection framework and joint tracking with detection framework are the two most popular paradigms. Specifically, MOT can be regarded as a combination of detection step to obtain frame-by-frame object detection results and tracking step to associate detections into trajectories. Among joint tracking with detection methods, some researchers apply single object trackers in MOT task by integrating SOT based motion model with affinity estimation, while the others use a multi-task objective to train detection and tracking simultaneously. The remaining sections are organized in this manner. In Section 2.3.1 we list the most commonly used metrics and datasets in multi object tracking field. Then tracking-by-detection and joint-tracking-and-detection methods are introduced in Section 2.3.2 and 2.3.3 respectively.

2.3.1 Metrics, Benchmarks and Datasets

Quite a few MOT datasets have been proposed to evaluate MOT algorithms. Here we introduce the metrics and corresponding MOT datasets including MOTChallenge benchmark [146,
For MOT, there are several metrics considered to evaluate the performance of algorithms, such as the two CLEAR MOT metrics \[18\], \textit{multi-object tracking accuracy} (MOTA) and \textit{multi-object tracking precision} (MOTP), \textit{mostly tracked} (MT), \textit{mostly lost} (ML), \textit{identity switches} (IDS), \textit{fragmentations of target trajectories} (FM), \textit{false positives} (FP), and \textit{false negatives} (FN). The FP is the number of false alarms in the whole video, and FN describes the number of targets missed by any tracked trajectories in each frame within the whole video. The IDS metric is the total number of times that the matched identity of a tracked trajectory changes, while FM is the number of times that trajectories are disconnected. Both IDS and FM metrics reflect the accuracy of tracked trajectories. The ML and MT metrics indicate the percentage of tracked trajectories less than 20\% and more than 80\% of the time span compared to the ground truth respectively. The MOTA metric is defined as following,

\[
\text{MOTA} = 100 \cdot \left(1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDS}_t)}{\sum_t \text{GT}_t}\right)[\%],
\]

where FN\(_t\) denotes false negatives, and FP\(_t\) denotes false positives at time index \(t\) in the sequence, with the hit/miss threshold set to be 0.7. In addition, IDS\(_t\) is identity switches of a trajectory, and GT\(_t\) is the number of ground truth objects. The MOTP metric computes the average dissimilarity between all true positives and the ground truth targets, as the average overlap between correctly matched hypotheses and respective objects.

\textbf{MOTChallenge} MOTChallenge is one of the most widely used benchmarks in MOT field. MOT15 \[146\] contains 22 video sequences (11 for training and 11 for testing) mostly captured by surveillance cameras where the targets of interest are pedestrians. It provides a platform where new datasets and multi-object tracking methods can be incorporated in a plug-and-play manner. MOT16 \[192\] extends the MOT15 dataset by adding more challenging sequences and thorough annotations. It has a higher pedestrian density and includes 14 video sequences (7 for training and 7 for testing). The MOT17 dataset contains the same videos as MOT16 dataset, but with more accurate ground truth and with three sets of detections for each video: Faster R-CNN \[221\], Deformable Part-based Model (DPM) \[78\], and Scale-Dependent Pooling detector (SDP) \[303\]. MOT19 \[63\] dataset is formed by 8 videos including 4 for training and 4 for testing with extremely high pedestrian density, reaching up to 245 pedestrians per frame on average.
KITTI The KITTI dataset [88] is collected for object tracking and detection, which is acquired from a moving vehicle with viewpoint of the driver. It consists of 50 video sequences (21 training and 29 testing), with a total of 19,000 frames. This dataset includes detections obtained two 2D detectors: DPM [78] and RegionLets [275].

UA-DETRAC The UA-DETRAC [278] benchmark dataset consists of 100 challenging videos captured from real-world traffic scenes. It is designed for vehicle surveillance scenarios with significantly more video frames (over 140,000 frames), annotated bounding boxes, vehicle type and attributes. The vehicles in the videos are acquired at different view angles and frequently occluded. Meanwhile, UA-DETRAC is designed for performance evaluation of both object detection and MOT. This benchmark considers the effect of detection performance on MOT evaluation and introduce a series of revised metrics.

Other datasets There are some older, and now less frequently used datasets. For example, the PETS09 dataset [80] and PETS16 datasets focus on multi-pedestrian detection, tracking as well as counting. Recently, UA VDT [68] is proposed to advance object tracking algorithms applied in drone based scenes.

2.3.2 Tracking-by-Detection

As discussed above, many MOT methods are proposed based on the tracking-by-detection framework, where objects are detected in each frame first and then detection results are associated in consecutive frames.

Classical methods. In classical methods, appearance and motion information are used to compute the similarities between objects in data association stage. In the JPDA method [81], all possible matches between the tracked objects and detection results in each frame are considered to calculate the joint probabilistic score in order to obtain tracking results. However, trackers’ computational complexity increases dramatically when there is an increasing number of objects in video sequences. Then, Rezatofighi et al. [224] introduce a computationally tractable approximation to the JPDA algorithm to find the m-best solutions of the integer linear program. In MHT algorithm [220], a tree of potential track hypotheses for each candidate target is built, and the likelihoods of the hypothesized matches over several frames are evaluated. In order to further improve MHT in exploring higher-order information, Kim et al. [125] propose an online appearance model for each track hypothesis. In addition, some other techniques have been used in solving the data as-
sociation problem including Hungarian algorithm [11, 22, 109], network flows [62, 312, 315, 212], tensor power iterations [239], linear programs [117], and generalized linear assignment optimization [64]. Moreover, Andriyenko and Schindler [5] tackle MOT task as an energy minimization problem by applying physical constraints such as target dynamics, mutual exclusion, and track persistence. Yamaguchi et al. [298] predict human behavior on the basis of an energy minimization problem and an agent-based behavioral model of pedestrians is designed in for the purpose of improve the quality of tracking results. Andriyenko et al. [6] formulate MOT problem as a discrete-continuous optimization problem, where data association and trajectory estimation are integrated together in an energy function.

**Deep learning based methods** Recently, deep learning is explored with increasing popularity in MOT with great success. Tracking-by-detection paradigm benefits from the rapid progress in the field of object detection [78, 221, 303, 48]. Many MOT trackers [12, 145, 247] rely on deep neural network as a powerful discriminative technique for feature extraction by using appearance models or re-ID methods to improve multi-object tracking results. Kuo et al. [139] adapts person re-ID method to “re-identify” targets in tracking task. Yang et al. [301] design a CRF model for the purpose of better distinguishing pedestrians with similar appearance. Leal-Taixé et al. [145] combine both appearance and short-term motion information in the form of optical flow, which can be used as input to a Siamese neural network in order to determine whether two proposed bounding box belong to the same object. Sadeghian et al. [230] adapt recurrent neural networks (RNN) and long short-term memory (LSTM) to describe the higher-order discriminative information. Deep Sort [286] algorithm takes offline detection results produced by Faster R-CNN and links detections according to an offline trained deep Re-ID model and Kalman filter motion model. In [253], a DNN based Re-ID techniques are used for affinity estimations, which exploits lift edges connecting two candidates across multiple frames in order to capture the long-term affinity. Xu et al. [296] extend relation network and design a unified framework in spatial-temporal domain for similarity measurement by integrating multiple information in an end-to-end manner.

### 2.3.3 Joint Tracking and Detection

Since temporal information is not explored in the detection phase, the quality of detections is often limited, further decreasing the accuracy of MOT. This calls for better capturing objects motion information in MOT task. To this end, many researchers adopted single object trackers in
MOT problems. These trackers attempt to utilize SOT methods to bring in objects’ temporal information and recover missing candidate detection results. Initially, SOT algorithms are used to solve a small sub-problem due to the concern that SOT tracker might learn easily from noisy samples, which will lead to drifting problem. Xing et al. [294] employ an optical flow-based SOT model to generate initial tracklets, while Xiang et al. [293] apply single object trackers to track objects in tracked state when target’s state space is divided into four sub-spaces. TLD [123] decomposes the long-term tracking task into three sub-tasks: tracking, learning and detection, where SOT tracker is used for tracking individual object in the tracking stage.

Some work has been done to integrate SOT methods into whole MOT tracking procedure. In the work of [29], target-specific classifiers are used to calculate the similarity in data association step under a particle filtering framework. Yan et al. [299] hold on to both the tracking results from SOT methods and the target detection results and select the best candidate using an ensemble framework. Yang and Nevatia [302] introduce an online learned discriminative part-based appearance model, where each tracklet is online learned to be distinguished from other tracklets and background. Then each tracklet is further tracked by SOT methods to partially overcome the missing detection results as well as to reduce difficulties in data associations when long gaps happen. Sadeghian et al. [230] use a multi-LSTM network for integrating different long-term information. Zhu et al. [330] combine the ECO track and attention-based network for data association, in a unified framework for the purpose of handling occlusion issue.

Above trackers’ promising results show the advantages of integrating SOT trackers as motion model. However, an additional affinity module is usually required in these approaches to handle occlusion condition. Besides, features learning for SOT trackers and affinity module are trained separately, which would increase the computation cost and memory demand. Qi et al. [43] apply online learned CNN-based SOT methods into online MOT task and focus on tackling the drift problem caused by occlusion among multiple objects. Yin et al. [307] combine Siamese SOT methods and metric learning into a unified triplet architecture by means of multi-task learning, where object motion and affinity model are unified into one single network.

Besides, some recent works attempt to tentatively tackle MOT task in an entirely end-to-end framework. Feichtenhofer et al. [77] run the R-FCN method [47] base detection architecture and simultaneously calculate correlation maps between high level feature maps of consecutive frames. Then the correlation map is passed to a secondary prediction tower for the purpose of predicting
frame-to-frame instance motion state. Ondruska and Posner [203] bring the RNN in the state estimation task and show the efficacy of applying RNN as an end-to-end solution. Milan et al. [193] design an RNN-LSTM based online framework, where both motion affinity estimation and data association are integrated into one deep learning network. In this method, LSTM is applied for data association purpose where constrains are learned from the training data. Chu and Ling [42] introduce an end-to-end model, where feature extraction, affinity estimation and multi-dimensional assignment are refined in a single network. RetinaTrack [180] trains both detection and tracking tasks jointly, which modifies popular single shot detection architectures that allow for extracting instance level features.

2.4 Crowd Counting

Object counting aims to estimate the number of objects, e.g., people or vehicle, in images or videos. In all these domains, crowd counting is of paramount importance, and it is crucial to building a more high-level cognitive ability in some crowd scenarios, such as video surveillance [37], crowd analysis [327], public safety [195] and urban planning. With the development of urbanization and the increase of the population in the world, crowd gathering such as parades, concerts and stadiums has become an important event. To this end, most early works estimate crowd count via pedestrian detection, regression-based methods or head detection. In recent years, with the excellent representation learning ability of CNNs, CNN-based methods have shown great progress for crowd counting. The remaining sections are organized in this manner. The most commonly used metrics and datasets in crowd counting field are given in Section 2.4.1. After that, traditional approaches and CNN-based approaches are reviewed in Section 2.4.2 and 2.4.3.

2.4.1 Counting: Metrics, Benchmarks and Datasets

In recent years, many datasets have been proposed to advance crowd counting methods, such as UCF CC 50 [113], ShanghaiTech [321], UCSD [37], WorldExpo’10 [314], and other less commonly used ones.
Metrics According to [321], two most common used metrics are Mean Absolute Error (MAE) and Mean Square Error (MSE), computed as

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |z_i - \hat{z}_i|, $$

$$\text{MSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (z_i - \hat{z}_i)^2}, \quad (2.2)$$

where $N$ is the total number of images, and $z_i$ and $\hat{z}_i$ indicate the ground-truth and predicted number of object in the $i$-th image respectively.

UCF_CC_50 The UCF_CC_50 dataset [113] contains 50 annotated crowd images created from publicly available Web images. It has a large variance of object counts, i.e., the number of objects in each image varies from 94 to 4543 and the average number of objects in each image is 1280. Notably, this dataset is quite challenging for deep learning algorithms due to the small scale data volume and large variance in crowd count.

ShanghaiTech The ShanghaiTech dataset [321] is formed by two parts, i.e., Part A and Part B, with the total number of 330, 165 annotated people. Part A includes 482 images randomly crawled from the Internet, and Part B includes 716 images captured from the streets in Shanghai. Both Part A and Part B are split into two parts for training and testing, i.e., 300 images for training and 182 images for testing in Part A, and 400 images for training and 316 images for testing in Part B.

UCSD The UCSD dataset [37] contains 2,000 frames with resolution of $158 \times 238$ collected from cameras on the sidewalk, with a total of 49,885 pedestrian instances collected from a single scene. This dataset has relatively low density with the number of people ranging from 11 to 46 per frame. The training set contains 800 frames and the remaining 1,200 frames held out for testing.

WorldExpo’10 The WorldExpo’10 dataset [314] includes 3,980 frames in total with average number of 50 people per frame, extracted from 1,132 video sequences captured with 108 surveillance cameras from Shanghai 2010 WorldExpo’10. The crowd density is relatively sparse in comparison to other datasets. 5 different video sequence with 120 frames per sequence are held out for testing.

Other Datasets There are some recent and less frequently used datasets. For example, Mall [40], UCF-QNRF [114], and ShanghaiTechRGBD [169] datasets focus on real-world crowd counting. Besides, DroneCrowd [280] and VisDrone2019 [331] are proposed to accomplish crowd counting
in drone based scenes.

### 2.4.2 Classical Methods

**Detection-based methods.** Early counting algorithms [87, 157, 147] usually apply a pedestrian detector or head detector via a sliding window in a frame. Specifically, body or part-based detector are used to locate people in the crowd image and sum them up. On the other hand, head detection is also used for crowd counting. Viola et al. [262] and Wu et al. [288] use motion information and appearance cues to train head detectors. However, these detection-based methods present unsatisfactory results when occlusions and background clutters affect dense crowd scenes.

**Regression-based methods.** To improve the counting results in occlusion and background cluttering situation, some researchers propose regression-based methods [38, 229, 40] that learn the mapping directly from an image to the count of the region. Regression-based algorithms usually first extract global (e.g., texture, gradient, edge features) or local information and then use some regression techniques such as linear Regression and Gaussian mixture regression to learn a mapping function. These methods perform reasonable results in tackling with the occlusion and background clutter problems. However, the spatial information is always ignored in these algorithms.

**Density map based methods.** In order to make use of spatial information, Lemptisky et al. [148] first propose a density map based method, which attempts to learn a linear mapping between features and object density maps in local regions. Pham et al. [211] observe that it is difficult to learn a linear mapping. To that end, the non-linear mapping and random forest regression are proposed, which introduces a crowdedness prior to train two different forests.

### 2.4.3 CNN-based Methods

Recently, benefiting from the excellent representation learning ability of CNNs, CNN-based methods have largely outperformed traditional crowd counting approaches. Zhang et al. [321] introduce multi-column network (MCNN) with three branches and use different kernel sizes (large, medium, small) to deal with multi-scale problems in crowd scenes. Then, inspired by MCNN, Switch-CNN [232] trains several independent density regressors for particular input patches and a switch classifier is trained to select the optimal regressor from multiple regressors. Hydra-CNN [204] designs a pyramid of image patches extracted from different scales to learn a multi-scale
non-linear regression model to estimate the final density map. Sindagi et al. [244] design a novel Contextual Pyramid CNN (CP-CNN) to encode local and global context into the density estimation process for generating high quality density maps. Different from above multi-column network architecture, some researchers employ single and deeper CNNs without increasing the complexity of the network. Li et al. [166] propose CSRNet by using the VGG16 backbone for feature extraction and adding dilated convolution layers to expand the receptive field, for the purpose of aggregating multi-scale context cues. Cao et al. [35] introduce encoder-decoder based SANet to perform accurate and efficient crowd counting. In SANet, the scale aggregation is used to extract multi-scale features in the encoder, and transposed convolutions are applied to generate high-resolution density maps in the decoder.

Moreover, some research work apply other techniques to solve crowd counting. Ranjan et al. [216] develop a two-branch network, where the first branch is designed to generate low-resolution density map and the second branch is used to incorporate the low-resolution density maps and the extracted feature maps to produce high resolution density maps. Cross-scene method [314] is trained alternatively by two different objectives, crowd density and crowd count. The pre-trained CNN model is fine-tuned using training samples similar to each target scene when a new scene is deployed. In contrast to the patch-based estimation methods, Shang et al. [236] propose to input the image into a network and estimate local and global counts simultaneously. Ma et al. [187] introduce the density contribution probability model based on the point annotations using the Bayesian loss function, and train loss on the count expectation at each annotated point rather than constraining the value at each pixel in density map.

### 2.5 Graph Neural Network

Graph has been attracting much attention because of its effective representation. It is formed by several nodes and edges/hyperedges. Specifically, the nodes are generally used to represent the targets or the target parts while the edges/hyperedges are used to describe the relations among them. In this section, we first briefly introduce some background on graph theory before deep learning step in, and outline some Graph Neural Networks (GNN) methods in Section 2.5.1. Then two applications including graph matching and tracking are described in Section 2.5.2 and 2.5.3 respectively.
2.5.1 Graph based Frameworks

Graph theory. In computer science field, graphs could be used to describe communication networks, data structure, computational devices and so on. Especially, in computer vision field, graphs are applied in tracking, data association, graph matching, and segmentation task. In graph theory and computer science, a finite graph could be represented by an adjacency matrix. The graph Laplacian is another useful matrix representation of a graph, which could indicate many useful properties of a graph. The properties of graphs via eigenvalues and eigenvectors of their associated matrices: the adjacency matrix and the graph Laplacian are studied in the spectral graph theory. Both matrices have been extremely well studied from an algebraic point of view. The Laplacian allows a natural association between continuous representation (e.g., vector spaces and manifolds), and discrete representations (e.g., graphs). The two most important applications of the Laplacian are spectral clustering and spectral matching, which could be used to solve the graph partitioning problem and the graph matching problem respectively.

Graph neural networks. Deep learning has been shown to be successful in many tasks such as object detection [218], speech recognition [2], machine translation [183]. Recently, there is increasing interest in applying deep learning methods to graphs, resulting in beneficial results in graph analysis. CNNs have strong power in extracting multi-scale spatial features and integrate those features in constructing highly expressive representations, leading to breakthroughs in many machine learning areas. When we look into CNNs and graphs, we could find some common important keys including local connection, shared weights and the use of multi-layer architecture. The notion of GNN was initially outlined in [92] and further elaborated in [235]. In 2009, Scarselli et al. [235] first proposed the concept of GNN. In this work, they extend existing neural networks to process the data represented in graphs. It learns the representations of object nodes by propagating neighbor information in an iterative manner until a stable state is reached.

After that, a large number of algorithms are proposed, which redefine the notion of convolution for graph. Spectral-based approaches and spatial-based (non-spectral-based) approaches are the two main categorises in this direction. Spectral-based approaches [161, 199] deal with a spectral representation of graphs. For example, Bruna et al. [30] design the spectral network. The graph convolution operation is defined in the Fourier domain by calculating the eigen decomposition of the graph Laplacian. Adaptive Graph Convolution Network [161] learns a “residual” graph Laplacian to contain relations between different nodes, then the “residual” graph Lapla-
cation is added to the original Laplacian matrix for future computation. Ng et al. [199] propose a Gaussian process-based Bayesian method to tackle the semi-supervised learning problems. On the other hand, spatial-based approaches define convolutions operation on the graph directly, with differently sized spatially close neighbors while maintaining CNN’s local invariance. Atwood and Towsley [7] introduce the diffusion-convolutional neural networks, where transition matrices are used to describe neighborhood nodes. In GraphSAGE framework [97], embeddings are generated by sampling and aggregating multiple features from the local neighborhood of a node. It contains three aggregate functions including mean aggregator, LSTM aggregator and Pooling aggregator, to enhance the expressive capability of the model. MPNN [89] outlines a general spatial-based framework, where the graph convolutions are operated as the message passing process. Specifically, the information in the graph can be passed from one node to another along edges directly.

In addition, some researchers attempt to add gate mechanism or attention strategy in GNNs in order to decrease the restrictions in previous GNN models and enhance the long-term information propagation. Li et al. [168] introduce the gated GNN, where the Gate Recurrent Units are applied in the propagation step. For the purpose of computing gradient, this method expands the recurrence for a fixed number of steps and applies back propagation through time. Tai et al. [251] propose two extensions: the Child-Sum and the N-ary Tree-LSTM, which are merged into the basic LSTM framework. Velickovic et al. [259] integrate the attention mechanism into the propagation step, by designing a single graph attention layer. This algorithm follows a self-attention strategy to calculate the hidden states of each node by attending over its neighbors. Over its development in the past decade, GNN framework have been applied across domains including supervised [217], semi-supervised [127], and few-shot [234].

2.5.2 GNN in Graph Similarity Calculation

The computation of distance/similarity between two graphs is one of the core operations in various graph-based applications, such as graph similarity search, graph database analysis, graph clustering.

Recently, several data-driven approaches based on neural networks have been proposed to tackle this task. SIAMESE MPNN [225] is an early work that employs a message passing neural network to capture the graph structure and learns a metric with a Siamese network approach, where the similarity is modeled as a summation of certain node-to-node similarity values. Ktena
et al.\textsuperscript{[137]} propose a GNN architecture in order to generate graph-level embeddings for calculate
the similarity. Besides graph-level embeddings, node-level information is also considered in some
algorithms. SimFNN method \textsuperscript{[14]} first designs a learnable embedding function to map every graph
into an embedding vector, and selects the important nodes out of an entire graph by an attention
mechanism. Then a pairwise node comparison method is used to supplement the graph-level em-
beddings with fine-grained node-level information. Li \textit{et al.}\textsuperscript{[167]} introduce node-node similarity
information into graph-level embeddings through a cross-graph attention mechanism. Wang \textit{et
al.}\textsuperscript{[272]} employ a GNN and additional layers to implement the node-to-node cross-graph affinity
function in order to learn the node-wise feature and the implicit structure information. Later, Bai
\textit{et al.}\textsuperscript{[15]} directly perform neural operations on the two sets of node embeddings without using
fixed-dimensional vectors to represent whole graph, in order to fully capture graphs in varying
sizes and link structures.

\subsection*{2.5.3 GNN for multi-object tracking
}

Since the GNN models have strong power to infer the relations among objects and tackle
non-Euclidean graph data, some researchers recently apply GNNs in multi-object tracking task.

Some methods used GNNs as optimization module in data association step \textsuperscript{[118, 185, 284, 285]}. Jiang \textit{et al.}\textsuperscript{[118]} formulate the frame-by-frame data association problem as maximum
weighted bipartite matching problem. This algorithm contains two modules, where the affinity
learning module is used to extract the appearance and motion information of objects, and the GNN-
based optimization module is designed for resolving the matching problem on a bipartite graph.
Similar to previous methods, GNN model \textsuperscript{[185]} is also used for optimizing graph structure to asso-
ciate the same target among different frames. Weng \textit{et al.}\textsuperscript{[284]} introduce a joint feature extractor
to learn the appearance and motion features in 2D and 3D spaces simultaneously and design a
feature interaction mechanism by introducing the GNN model. A graph is constructed with each
node being the object feature. Then, each node can update its feature by aggregating features from
other nodes at every layer of the GNNs.

Besides data association, some researchers use the GNN models for other purposes. Li \textit{et
al.}\textsuperscript{[155]} introduce an appearance graph network and a motion graph network to capture the appearance
and the motion similarity separately. Since GNN is able to indicate the global information
of all nodes and edges in the updating process, another graph network is used for future updating
mechanism, so that nodes, edges and the global variable can all be updated. Exploited by the classical network flow formulation of MOT, Brasó and Leal-Taixé [28] design a fully differentiable framework based on Message Passing Networks. In this algorithm, a node indicates an object detection, an edge indicates the connection between two nodes, and an active edge shows the two detection results belong to the same object.
CHAPTER 3

Hybrid Structure Hypergraph for Online Deformable Object Tracking

In this chapter, I attempt to use the hybrid structure hypergraph to encode the dependencies among target parts in consecutive frames to solve the deformable object tracking.

3.1 Motivation

Although significant progress has been achieved in recent years, tracking a deformable object remains a challenging task due to large changes of targets in appearance caused by deformation, occlusions, and clutter background. Most of previous methods focus on modeling the appearance variations of targets in the bounding box, e.g., correlation filters [57, 104], sparse representation [188, 116, 320], online boosting [10, 277], deep learning [184, 197, 255], etc., which suffer from the drift problem due to considerable pixels of the background are encompassed in the bounding box, especially when large deformation and occlusion exist.

To this end, some recent methods use the part-based model to encode the appearance variations of target accurately, e.g., [140, 306, 165, 181]. However, these methods use the bounding box based parts to describe the target appearance, which include background pixels affecting the tracking performance. Another related work [198] design a novel dissimilarity measure for the clustering of correspondences between keypoints between consecutive frames to complete the deformable object tracking task. However, the interest point generating strategy makes it sensitive to local noise in severe occlusion or clutter backgrounds.

Besides, some recent works [33, 67, 66] use the over-segmented superpixels to replace the bounding box based parts and construct different kinds of graphs/hypergraph model to represent the target. In contrast to bounding box, superpixel is able to describe the appearance of targets more accurately, which contains less background regions. However, the dependencies of different object parts in consecutive frames are different with each other, especially when large deformation and occlusion happen. Using the graph or uniform hypergraph is less effective in handling deformation and occlusion challenges in unconstrained scenes.
In this work, we proposed a hybrid structure hypergraph based tracker, which uses non-uniform hypergraph to describe the dependencies among target parts in consecutive frames, as illustrated in Figure 3.1(c). Compared with the previous tracker DGT \cite{33} and SAT \cite{66}, our method is more effective in handling large deformation and severe occlusion challenges. Specifically, we first use the SLIC over-segmentation algorithm \cite{3} to generate the superpixels in each video frame and apply the graph cut algorithm \cite{27} to produce the candidate parts. Then, we construct a non-uniform hypergraph to capture the hybrid dependencies among candidate parts across multiple frames, where each node corresponds to a candidate part, and each edge\textsuperscript{1} encodes the dependen-

\textsuperscript{1}In this work, we use the terminology “edge” to indicate the self-loop, conventional edge and hyperedge, and use the terminology “hyperedge” to indicate the edge involving more than two nodes specifically.
cies of the parts, i.e., the consistencies in both appearance and motion. Inspired by \cite{283,66}, we propose an approximate algorithm to extract the dense structures on the hypergraph to decide the parts belonging to the target. After that, the target state (i.e., center location in pixel and scale) is determined by analyzing the extracted parts belonging to the target. Several experiments are carried out on publicly available Deform-SOT dataset, to demonstrate the favorable performance of the proposed method against state-of-the-art trackers.

Results of this work is published in \cite{162}. My contribution in this joint work is mainly on algorithm design, implementation, and conducting experiments.

### 3.2 Related Work

#### 3.2.1 Part based Trackers

To perform deformable object tracking, part-based trackers are more flexible than bounding box based method to encode local appearance variations of the target. For example, Kwon and Lee \cite{140} propose a local patch-based appearance model and introduce the Basin Hopping Monte Carlo (BHMC) sampling method for visual tracking. Yao et al. \cite{306} develop an online latent structural learning strategy to model the unknown parts using latent variables. Li et al. \cite{165} propose the reliable patch trackers, which can identify and exploit the reliable patches that can be tracked effectively. Lukezic et al. \cite{181} propose the constellation model with correlation filters such that the geometric and visual constraints.

On the other hand, Cai et al. \cite{33} formulate the deformable object tracking task as tracking the dynamic undirected geometric structure graph of the target. Du et al. \cite{67} improve the method \cite{33} by using hypergraph instead of graph to capture the high-order interactions among target parts. Specifically, the geometric hypergraph is constructed and learned to match the target parts and the candidate parts in two consecutive frames. Furthermore, Du et al. \cite{66} exploit the high-order constraints of target parts in multiple consecutive frames rather than only two frames by constructing a uniform hypergraph.

#### 3.2.2 Hypergraph representation

A hypergraph is extended by a graph where an edge can connect any number of given nodes in the graph, achieving a better representation. Liu et al. \cite{172} formulate the clustering problem
as ensembles of k-ary affinity relations and clusters correspond to subsets of objects with maximal average affinity relations. To improve the efficiency, Liu and Yan \[173\] randomly sample minimal size samples and generate hypotheses to detect the underlying structures in data. To deal with large hyperedges, Purkait et al. \[215\] propose a novel guided sampling strategy based on the concept of random cluster models and demonstrate its effectiveness in both a theoretical and an empirical standpoint. Recently, Feng et al. \[79\] develop a deep learning based hypergraph neural networks for data representation learning, which encodes high-order data correlation in a hypergraph structure.

3.3 Hypergraph based tracker

3.3.1 Constructing hypergraph

As discussed above, we process multiple consecutive frames at a time. Similar to \[66\], we construct a frame buffer $\Gamma$ of size $M$ to store the frames to be processed. Notably, when the latest frame index $m \leq M$, we make $M - m + 1$ copies of the previous frames of $\Gamma$. When receiving a new frame, we will remove the earliest frame in $\Gamma$ and add the new receiving frame. In this way, we can process the video online. Then, we use the SLIC over-segmentation method \[3\] to generate several superpixels of each frame in the frame buffer $\Gamma$, i.e., $\mathcal{P} = \{p_1, \cdots, p_n\}$, where $p_i$ is the $i$-th generated superpixel, and $n$ is the total number of superpixels. After that, similar to \[66\], we form the energy function and use the graph cut algorithm \[27\] to get coarse labeling of each superpixel as belonging to the target or background.

Given the candidate parts of the target, we construct a non-uniform hypergraph to describe the hybrid dependencies among parts in consecutive frames, denoted as $\mathcal{G} = (V, E, A)$, where $V$ is the node set of the hypergraph, $E$ is the edge set, and $A$ is the weight set corresponding to the edges. A hypergraph is an extension of a conventional graph, where an edge can connect more than two nodes. In other words, an edge is a subset of nodes. If all edges are constructed by the same number of nodes, the hypergraph is called uniform hypergraph. Otherwise, we call it non-uniform hypergraph.

To ensure the running efficiency of the tracker, we design a simple approximate sampling strategy to focus on the edges including the nodes in the current frame. We enumerate degree $d$, (i.e., $1 \leq d \leq D$) to generate several edges with different degrees. Note that the edges with degree
$d = 1$ indicate the self-loops of the graph. For the $d$-th degree edges, we first collect several node arrays from the first $M - 1$ frames in $\Gamma$. Each node array consists of $d - 1$ nodes. Specifically, any two nodes in each node array should not belong to the same frame. Then, we construct the edges with $d$-degree by combining the collected node arrays with the nodes of the $M$-th frame in $\Gamma$ (i.e., the current frame). In contrast to the previous method [66] constructing the edges by enumerating all combinations of the nodes in $\Gamma$, the proposed method focuses on the edges including the nodes in the current frame to control the scale of the hypergraph for running efficiency.

The weight of self-loop $A(v_i)$ in the hypergraph indicates the probability of a node belonging to the target. Since we do not have accurate prior information of the nodes, thus we set $A(v_i) = 1.0$ for all nodes (i.e., superpixels) coarsely labeled as the target by the graph cut algorithm. Meanwhile, the pairwise dependency $A(v_i, v_j)$ encodes the appearance similarity between two nodes $v_i$ and $v_j$, which is calculated as $A(v_i, v_j) = e^{-\lambda_1 \chi(U(v_i), U(v_j))}$, where $U(v_i)$ and $U(v_j)$ are the features corresponding to the node $v_i$ and $v_j$, respectively, which is constructed by catenating the HSV color histogram and Local Binary Pattern (LBP) texture histogram, $\chi(\cdot, \cdot)$ calculates the chi-squared distance between two features $U(v_i)$ and $U(v_j)$, and $\lambda_1$ is the parameter controlling the sensitive of the distance to the pairwise dependency.

Furthermore, the high-order dependency $A(v_{1:d})$ (i.e., $d \geq 1$) encodes the motion consistency among the nodes $v_{1:d} = \{v_1, \ldots, v_d\}$. Intuitively, the target parts move smoothly in a short time interval. We compute $A(v_{1:d})$ based on fitting the motions of nodes $v_{1:d}$ using quadratic spline interpolation. Specifically, we fit a piecewise parametric curve to a subset of the nodes $S_v \subset v_{1:d}$ with equally interval in temporal domain. Then, we calculate the $\ell_2$ distance of the center locations of the remaining nodes with their predictions based on the fitted curve to get the high-order dependency, i.e., $A(v_{1:d}) = e^{-\lambda_2 \sum_v s_v \|C(v) - Q(m_v)\|^2}$, where $C(v)$ is the center location of node $v$, $m_v$ is the frame index of node $v$, $Q(m_v)$ is the predicted location of node $v$, and $\lambda_2$ is the parameter controlling the sensitive of the predicted deviation to high-order dependency.

### 3.3.2 Extracting dense structures

Inspired by [66], we set each node as a starting point, and search the corresponding dense structure. In this way, we can extract all dense structures on the hypergraph to determine the state of target parts in the current frame. Specifically, for the node $v_p$, we define the corresponding structure (sub-hypergraph) “dense”, if the nodes in the structure are interconnected by a large number of
different degree edges with large weights. We assume there exits $\alpha$ nodes in the structure. Let $y = (y_1, \cdots, y_n)$ to be the indicator variable corresponding to node $v_s$. That is, $y_i = \frac{1}{\alpha}$, if the node $v_i$ belongs to the dense structure; otherwise $y_i = 0$. Thus, we have $\sum_{i=1}^{n} y_i = 1$. To treat each node equally at the beginning, for a starting node $v_s$, we initialize the indicator variable with the score $\frac{1}{|N(v_s)|}$, $i = 1, \cdots, |N(v_s)|$, where $|N(v_s)|$ is the number of nodes in $v_s$’s neighborhood. To extract the dense structure of node $v_s$, we maximize the weight summation of the edges included in the structure, i.e.,

$$
\mathbf{y}^* = \argmax_{\mathbf{y}} \sum_{d=1}^{D} \lambda_d \sum_{v_1:d \in N(v_s)} A(v_1:d) \prod_{y_d}^{d} y_d
$$

\text{s.t.} \sum_{i=1}^{N(v_s)} y_i = 1, \quad y_s = \frac{1}{\alpha}, \quad \forall i, \quad y_i \in \{0, \frac{1}{\alpha}\},

(3.1)

where $D$ is the maximal degree of the edges in $G$, and $\lambda_d$ is the preset influence factors of edges with different degree, $N(v_s)$ is the neighborhood of node $v_s$. Notably, we would like to highlight that the objective function for tracking in [66] is a specific case of (3.1). That is, if we set $\lambda_{d^*} \neq 0$ for a specific $d^* \geq 3$, and $\lambda_d = 0, \forall d \neq d^*$, the non-uniform hypergraph $G$ will reduce to a $d^*$-uniform hypergraph, similar to that in [66].

The optimization problem in (3.1) is a combinational optimization problem, since we do not know any prior information about the number of nodes $\alpha$ in the dense structure. To reduce the computational complexity, we convert (3.1) to an approximate optimization problem, i.e., relax the

Figure 3.2: The success and precision plots over the dataset using OPE. The representative scores for each tracker are reported in the legends.
constraint $y_i \in \{0, \frac{1}{\bar{\alpha}}\}$ to $y_i \in [0, \frac{1}{\bar{\alpha}}]$. Meanwhile, to avoid the degeneration problem, we require that each extracted dense structure include at least $\bar{\alpha}$ nodes, i.e., $\alpha \geq \bar{\alpha}$. Then, the constraint $y_i \in [0, \frac{1}{\bar{\alpha}}]$ is converted to $y_i \in [0, \bar{\alpha}]$. We extend the greedy pairwise updating proposed in [173] to extract the dense structure on non-uniform hypergraph efficiently. Specifically, we can increase one component $y_p$ and decrease another one $y_q$ appropriately, to increase the objective, i.e.,

$$\lambda(y) = \sum_{d=1}^{D} \lambda_d \sum_{v_1:d-2\neq p,q} A(v_{1:d-2},p,q) \prod_{j=1}^{d-2} y_{v_j},$$

while satisfying the constraint $\sum_{i=1}^{n} y_i = 1$, i.e., $y'_l = y_l$, $\forall l \neq p, l \neq q$; $y'_l = y_l + \eta$, $l = p$; $y'_l = y_l - \eta$, $l = q$. The different of objective $\Theta(y')$ after updating is $\Theta(y') - \Theta(y) = \varphi_{p,q}(y)\eta^2 + (\phi_p(y) - \phi_q(y))\eta$, where $y' = (y'_1, \cdots, y'_n)$, and

$$\varphi_{p,q}(y) = -\lambda_2 \cdot A(p,q) - \sum_{d=3}^{D} \lambda_d \sum_{v_1:d-2\neq p,q} A(v_{1:d-2},p,q) \prod_{j=1}^{d-2} y_{v_j},$$

$$\phi_p(y) = \sum_{d=2}^{D} \lambda_d \sum_{v_1:d-1\in N(v_s)} A(v_{1:d-1},p) \prod_{j=1}^{d-1} y_{v_j}. \quad (3.2)$$

Thus, to maximize the difference to increase the objective $\Theta(y)$, we can select the updating step $\eta$ as follows.

$$\eta = \begin{cases} 
\min(y_q, \frac{1}{\bar{\alpha}} - y_p), & \text{if } \varphi_{p,q}(y) \geq 0; \\
\min \left( y_q, \frac{1}{\bar{\alpha}} - y_p, \frac{\phi_q(y) - \phi_p(y)}{2\varphi_{p,q}(y)} \right), & \text{if } \varphi_{p,q}(y) < 0; \\
\min(y_q, \frac{1}{\bar{\alpha}} - y_p), & \text{if } \phi_p(y) = \phi_q(y), \varphi_{p,q}(y) > 0. 
\end{cases} \quad (3.3)$$

39
Here we can assume $\phi_k(y) > \phi_j(y)$. If $\phi_k(y) < \phi_j(y)$, we exchange indexes $k$ and $j$ to maximize the difference $\lambda(y') - \lambda(y)$. In this way, we can use the similar heuristic strategy in [174] to compute the local maximizer $y^*$ for the dense structure extraction. After extracting the dense structures on the non-uniform hypergraph, we use the similar strategy proposed in [66] to estimate the target state and update the appearance model online. In this way, we can complete the object tracking task effectively.

### 3.4 Experiment

We implement the proposed tracker in with Matlab and C++, which can be run at 1.0 frames per second (fps) on a machine with a 3.4 GHz Intel i7 processor and 8 GB memory.

![Figure 3.4: The tracking performance with different maximal degree of hypergraph.](image)

#### 3.4.1 Deform-SOT Dataset

We carry out several experiments on the publicly available online deformable object tracking dataset, i.e., Deform-SOT dataset presented in [66], to demonstrate the performance of the proposed method in unconstrained environments. The Deform-SOT dataset consists of 50 challenging sequences, which are grouped into 6 visual attributes such as large deformation, severe occlusion, and abnormal movement. There are 20 of the 50 sequences used in the previous works (e.g., avatar,

---

2The code is available at [www.cbsr.ia.ac.cn/users/lywen](http://www.cbsr.ia.ac.cn/users/lywen)
Figure 3.5: The success plots of OPE with different attributes over the dataset. The representative scores for each tracker are reported in the legends.

carscale and waterski), and the rest 30 sequences collected from the Internet (e.g., bike, lola and uneven-bars). As shown in Figure 3.5, we list all the sequences with the corresponding visual attributes.
Figure 3.6: The precision plots of OPE with different attributes over the dataset. The representative scores for each tracker are reported in the legends.

3.4.2 Evaluation Metrics

To evaluate the performance of trackers, we run One-Pass Evaluation (OPE) [290], which uses the ground truth in the first frame to initialize the trackers. Two measures, namely success plot and precision plot, are used to compare performances. The Area Under Curve (AUC) of each success plot is used to represent the performance of the tracking algorithms. The precision score of the center location error for the threshold equals to 20 pixels is also reported to indicate the
tracking performance comprehensively.

3.4.3 Parameter analysis

We study the effect of a critical parameter, i.e., the maximal degree $D$ of the non-uniform hypergraph $G$ to the tracking performance. To examine the effect of $D$, we report the performance of the proposed method on 15 sequences from Deform-SOT dataset with different maximal degree, i.e., Ours-$D$ ($D = 2, 3, 4$) in Figure 3.4. As shown in Figure 3.4, we find that Ours-2 performs worse than Ours-3, since the high-order dependencies among parts (i.e., the information of motion consistency) are considered to improve the performance. Meanwhile, Ours-3 performs better than Ours-4, because the hybrid structure hypergraph with the degree that is too high fails to adapt the large deformations and occlusions of target parts effectively. Thus, we choose the maximal degree $D = 3$ in our experiments. Other parameters in our algorithm are chose empirically using the grid search strategy, i.e., $\lambda_1 = 1.0$ in pairwise dependency, $\lambda_2 = 0.0625$ in high-order dependencies, and $\lambda_1 = 1.0, \lambda_2 = 3.0$, and $\lambda_d = 9.0 \ (d \geq 3)$ in \((3.1)\). The minimal size of the searched dense
structures $\hat{\alpha} = 2$.

### 3.4.4 Result Analysis

To demonstrate the favorable performance of the proposed method, we evaluate the proposed method against 15 trackers, \textit{i.e.,} GGT [67], SAT [66], DGT [33], Staple [20], LGT [36], KCF [104], DSST [55], CN [58], SPT [274], ASLA [116], Struck [98], LOT [205], LSL [306], SCM [324], and TLD [122]) with the top performance on the Deform-SOT dataset. Compared with aforementioned trackers, some visual tracking results on several sequences (\textit{e.g.,} neymar, run, and waterski) are shown in Figure 3.7. The qualitative results indicate that our method performs more robust than other methods.

Furthermore, as shown in Figure 3.2, the success and precision plots of the overall dataset indicate that our method performs favorable against 15 state-of-the-art trackers, which demonstrates the effectiveness of using non-uniform hypergraph in tracking task. Our method achieves the best performance in 5 challenging scenarios out of 6 according to the success plot, especially when severe occlusion and large deformation challenges happen, as presented in Figure 3.5 and Figure 3.6. In summary, hybrid structure hypergraph considers several degrees of dependencies among target parts jointly, which is more accurate to describe the dependencies among different parts to improve the tracking performance, comparing to the uniform hypergraph in [66].

### 3.5 Conclusion

In this work, we propose a hybrid structure hypergraph to complete the online deformable object tracking task, which formulates the tracking task as the dense structure extracting problem on a non-uniform hypergraph, solved by an approximate algorithm efficiently. Experimental results on the public available online deformable object tracking dataset, \textit{i.e.,} Deform-SOT dataset, demonstrate that our method performs favorable against state-of-the-art methods.
CHAPTER 4
Non-Uniform Hypergraph for Multi-Object Tracking

In this chapter, the hypergraph model is used to solve the multi-object tracking task, which aims to exploit adaptive degrees of dependencies among objects to improve the performance of the tracker in various scenarios.

4.1 Motivation

Multi-object tracking (MOT) is an interesting but critical problem in computer vision with many applications, such as surveillance, behavior analysis, and sport video analysis. Although the performance of MOT has been significantly improved in recent years [41, 125, 281, 254], it is still a challenging problem due to several factors such as missed detections, identification switches, and fragmentation of the tracked objects.

Tracking-by-detection is a popular framework in the MOT field, which employs a pre-trained object detector to locate object regions in individual frames, and then associates detections across frames to generate target trajectories. Most existing methods only consider the pairwise dependencies between detections (e.g., [224, 61, 194, 73]), and fail to take full advantage of the high-order dependencies among multiple targets across frames. This strategy is less effective when nearby objects with similar appearance or motion patterns occlude each other in video sequences. To that end, some recent methods [125, 45, 239, 125, 282, 281] attempt to exploit the high-order information to improve the tracking performance, such as tensor power iterations [239], high-order motion constraints [45, 81], dense structure search on hypergraph [282, 281], and multiple hypothesis tracking [125]. However, the aforementioned methods merely exploit fixed degrees of dependencies among objects, which limits the flexibility of the hypergraph model in complex environments, and calls for adaptive dependency patterns. For example, as shown in Figure 4.1, 3-uniform hypergraph is unable to describe the dependencies between two tracklets of target 1 and 4 correctly. In contrast to the uniform hypergraph, non-uniform hypergraph better adapts to different degrees of dependencies among tracklets, and produces more accurate results.

A hypergraph is a generalization of a conventional graph where an edge can join more than two nodes.
Figure 4.1: (a) Two previous methods using 3-uniform hypergraph $H^2_T$ [282] and $FH^2_T$ [281], often fails to describe the dependencies among tracklets, when occlusion or missed detection happen. (b) The proposed method uses the non-uniform hypergraph to encode different degrees of dependencies among tracklets effectively.

In this work, we design a new non-uniform hypergraph learning based tracker (NT) to solve the multi-object tracking task. The proposed method has much stronger descriptive power to accommodate different challenging tracking scenarios than the conventional graph [61] or uniform hypergraph [282, 281]. The nodes in the hypergraph are used to represent the tracklets\(^2\) and the hyperedges with different degrees encode similarities among tracklets to assemble various kinds of appearance and motion patterns. The MOT problem is formulated as searching dense structures on the non-uniform hypergraph. In contrast to previous methods [282, 281] using the sole degree hypergraph model, we mix hyperedges of different degrees and learn their relative weights automatically from the data using the structural support vector machine (SSVM) method [120]. An efficient approximation algorithm is designed to exploit the dense structures to generate long trajectories of objects to complete the tracking task. In addition, we use a near-online strategy for MOT, \(i.e.,\) the dense structure searching is performed on the non-uniform hypergraph to generate

\(^2\)The terminology “tracklet” indicates a fragment of target trajectory. Notably, the input detections in individual frames can be treated as tracklets of length one.
short tracklets in a local temporal window, and then associate those short tracklets to the tracked targets to get the final trajectories of targets at the current time stamp. This process is conducted repeatedly to complete MOT in video sequences. We conduct several experiments on various challenging datasets, \textit{i.e.}, PETS09, ParkingLot sequence, SubwayFace, and MOT16 benchmark, to demonstrate the effectiveness of the proposed method compared to the state-of-the-art MOT methods.

Results of this work is published in [279]. My contribution in this joint work is on brainstorming method design and the implementation of affinity score computation.

### 4.2 Related Work

MOT methods can be roughly classified into three categories, 1) online strategy, 2) off-line processing strategy, and 3) near-online strategy. If there occurs an error in tracking, it is hard for online strategy (\textit{e.g.}, [304, 293, 308]) to recover from due to imprecise appearance or motion measurements. Thus, many algorithms focus on off-line strategy (\textit{e.g.}, [17, 254, 194]). To make the association step efficient, [17] formulate the association as a constrained flow optimization problem, solved by the k-shortest paths algorithm. Tang \textit{et al.} [254] present a graph-based formulation that links and clusters person hypotheses over time by solving an instance of a minimum cost lifted multicut problem. In addition, Milan \textit{et al.} [194] pose MOT as minimization of a unified discrete-continuous energy function using the L-BFGS and QPBO algorithms. However, as only association between pairs of detections in local temporal domain are considered, the aforementioned methods do not perform well when multiple similar objects appear in proximity with clutter backgrounds.

To alleviate this problem, Dehghan \textit{et al.} [61] use a graph to integrate all the relations among objects in a batch of frames and formulate the MOT problem as a Generalized Maximum Multi Clique problem on the graph. Wen \textit{et al.} [282] exploit the motion information to help tracking and formulate MOT as the dense structure searching on a uniform hypergraph, in which the nodes correspond to tracklets and the edges encode the high-order dependencies among tracklets. To further improve the efficiency, an approximate RANSAC-style approach is proposed in [281] to complete the dense structure searching.

Besides, Choi \textit{et al.} [41] designs a near-online strategy, which inherits the advantages of
both online and offline approaches. The tracking problem is formulated as a data-association between targets and detections in a temporal window, that is performed repeatedly at every frame. In this way, the algorithm is able to fix any association error made in the past when more detections are provided. Wang and Fowlkes [273] present an end-to-end framework to learn parameters of min-cost flow for MOT problem using a tracking-specific loss function in the SSVM framework. Nevertheless our approach uses the non-uniform hypergraph to describe the high-order dependencies among tracklets, and uses SSVM framework to learn the weights of the hyperedges with different degrees.

4.3 Non-uniform Hypergraph

A hypergraph is a generalization of a conventional graph, where an edge can join more than two nodes. We use \( G(V, E, A) \) to denote a (weighted) hypergraph, where \( V = \{ v_1, \cdots, v_n \} \) is the node set, \( v_i \) is the \( i \)-th node and \( n \) is the total number of nodes, \( E \) is the set of hyperedges, and \( A \) is the affinity set corresponding to the edges/hyperedges. Specifically, we define \( E = E_1 \cup \cdots \cup E_D \), where \( E_1 = \{ (v_1), \cdots, (v_n) \} \) is the set of self-loops, \( E_2 \subseteq V \times V \) is the set of conventional graph edges, \( E_d \subseteq V^d \) is the set of hyperedges with degree \( d, d = 3, \cdots, D \), and \( D \) is the maximal degree of hyperedges. If all hyperedges in \( G \) have the same cardinality \( d \), \( G \) is a \( d \)-uniform hypergraph (i.e., \( E_{d'} = \emptyset \) for \( d' \neq d \)); otherwise, \( G \) is a non-uniform hypergraph. For node \( v \), we denote its neighborhood as \( N(v) \), which is the set of nodes connected to \( v \).

Similar to [281], we define a dense structure on \( G \) as a sub-hypergraph that has the maximum affinities combining all hyperedges, edges and self-loops of nodes. We introduce an indicator variable \( y = (y_1, \cdots, y_n)^\top \), such that \( \sum_{i=1}^n y_i = 1 \), and \( y_i = \{0, 1/\alpha\} \), where \( \alpha \) is the number of nodes in the dense structure. The affinity summation of the hyperedges, edges and self-loops of nodes of the dense structure can be calculated as

\[
\Theta(y) = \sum_{d=1}^D \lambda_d \sum_{v_{1:d} \in V} A(v_{1:d}) \underbrace{y_1 \cdots y_d}_{d}
\]

where \( v_{1:d} = \{ v_1, \cdots, v_d \} \), \( y_i \) is the indicator variable corresponding to node \( v_i \) (\( i = 1, \cdots, d \)), \( i.e., \), \( y_i = 1/\alpha \) if node \( v_i \) belongs to the dense structure; otherwise, \( y_i = 0 \). Thus, \( y_1 \cdots y_d \) indicates the confidence of the hyperedge (\( d > 2 \)), edge (\( d = 2 \)), or self-loop (\( d = 1 \)) \( v_{1:d} \) included in the dense structure. Weights \( \lambda_1, \cdots, \lambda_D \) are used to balance the significance of different degrees of
hyperedges. Notably, we use the terminology “affinity” to indicate the value associated to each edge/hyperedge, which reflects the similarities of the nodes in the corresponding edge/hyperedge. Meanwhile, the terminology “weight” is adopted to indicate the numbers used to balance the significance of different degrees of hyperedges, edges and self-loops in dense structure searching. The weights of \( d \)-th hyperedges may consist of \( \kappa > 1 \) terms (e.g., the weights of the second degree hyperedges may consist of the appearance similarity and motion consistency between two tracklets). In such cases, the weight \( \lambda_d \) is a vector with the size \( 1 \times \kappa \), and the affinity \( A(v_1:d) \) is also a vector with the size \( \kappa \times 1 \). The affinity summation from degree 1 to \( D \) in (4.1) describes the overall affinity score combining all the hyperedges, edges, and self-loops of the nodes in the dense structure. Thus, we need to maximize the overall affinity score to exploit the dense structures to complete multi-object tracking.

4.3.1 MOT Formulation

We use the non-uniform hypergraph to encode the relations among different tracklets. For each video clip, MOT is initialized by the tracklets. Notably, our definition of tracklet generalizes cases for single detection, i.e., \( m_i = 1 \), or continuous sequence of detections, i.e., the frame index set \( \{t_{i1}, \cdots, t_{im_i}\} \) corresponding the detections on the tracklet, where \( t_{ij} \) is an integer, and \( t_{ij} < t_{ij+1}, j = 1, \cdots, m_i - 1 \). Let \( T = \{T_1, \cdots, T_n\} \) be the tracklet set in the video sequence, where \( T_i = \{B_{i1}, \cdots, B_{im_i}\} \) consists of \( m_i \) frame detections, and \( B_{ij} = (x_{ij}, y_{ij}, w_{ij}, h_{ij}, t_{ij}) \), where \( (x_{ij}, y_{ij}) \) and \( (w_{ij}, h_{ij}) \) are center location and dimension of the detection, and \( t_{ij} \) is the corresponding frame index.

We formulate the MOT problem as searching dense structures on a non-uniform hypergraph \( G(V, E, A) \). Specifically, we only consider the edges/hyperedges with no duplicate nodes, i.e., each edge/hyperedge contains different nodes. We set every node in \( G \) as the starting point, and search the corresponding dense structure from their neighborhoods. To treat each point equally at the beginning, for a starting point \( v_s \), we initialize the indicator variables with the score \( \frac{1}{|N(v_s)|} \), \( i = 1, \cdots, |N(v_s)| \), where \( |N(v_s)| \) is the number of nodes in \( v_s \)’s neighborhood. For node \( v_s \), the dense structure searching problem is formulated as

\[
\arg\max_y \sum_{d=1}^{D} \lambda_d \sum_{v_1:d \in N(v_s)} A(v_1:d) y_1 \cdots y_d \\
\text{s.t.} \sum_{i=1}^{N(v_s)} y_i = 1, \ y_s = \frac{1}{\alpha}, \ \forall i, \ y_i \in \{0, \frac{1}{\alpha}\},
\]

(4.2)
where $\mathcal{N}(v_s)$ is the neighborhood of node $v_s$. Notably, the constraint $y_s = 1/\alpha$ indicates that the node $v_s$ is included in the searched dense structure, and $y_i = 1/\alpha$ indicates that the $i$-th node in $\mathcal{N}(v_s)$ is included in the searched dense structure, otherwise, $y_i = 0$.

The problem in (4.2) is a combinational optimization problem, since we cannot know the number of nodes in the dense structure $\alpha$ priorly. To reduce the complexity of this NP-hard problem, we relax the constraint $y_i \in \{0, \frac{1}{\alpha}\}$ to $y_i \in [0, \frac{1}{\alpha}]$. In addition, we set a minimal size of the sub-hypergraph to be a constant number $\hat{\alpha}$ to avoid the degeneracy, i.e., $\hat{\alpha} \leq \alpha$. Thus, the constraint is converted to $y_i \in [0, \frac{1}{\hat{\alpha}}]$. We would like to highlight that the objective function for dense structure exploiting in [281] is a specific case of (4.2), i.e., if we set $\lambda_{d^*} \neq 0$ for a specific $d^* \geq 3$, and make $\lambda_d = 0$, $\forall d \neq d^*$, the non-uniform hypergraph $\mathcal{G}$ will degenerate into a $d^*$-uniform hypergraph, and the objective in (4.2) becomes similarly to that in [281]. The optimization algorithm in [281] for uniform hypergraph model cannot be directly applied to solve the problem in (4.2).

After exploiting the dense structures, the radical post-processing strategy presented in [282] is adopted to remove the conflicts among the searched dense structures. Then, we stitch the tracklets in each post-processed dense structures to form the long trajectories.

### 4.3.2 Hypergraph Construction

**Enforcing edge/hyperedge constraints.** In the practical MOT scenarios, the objects have two physical constraints: 1) one object cannot occupy two different places at a time; 2) the velocity of an object is below certain maximum possible velocity. As such, in constructing the hypergraph, two nodes connected by one edge/hyperedge should not overlap in time, and the distance between the last and first detections of the tracklet should not larger than the maximal distance that can reach with the maximal possible velocity. These two constraints can reduce the number of edges and hyperedges and computational complexity.

**Calculating self-loop affinity.** We associate a node with a score to reflect its reliability being a true tracklet of an object, i.e., $A(v_i) = \rho(v_i)$, where $\rho(v_i) (0 \leq \rho(v_i) \leq 1)$ is the confident score of the tracklet $v_i$ calculated by averaging the scores of all detections in the tracklet.

**Calculating edge affinity.** The edges in the hypergraph encode the similarities between two nodes (tracklets), which consists of three terms: HSV histogram similarity $\mathcal{P}_{col}(v_i, v_j)$, CNN feature
similarity $\mathcal{P}_{\text{cnn}}(v_i, v_j)$, and local motion similarity $\mathcal{P}_{\text{mot}}(v_i, v_j)$, i.e.,

$$\mathcal{A}(v_i, v_j) = \left[ \mathcal{P}_{\text{col}}(v_i, v_j), \mathcal{P}_{\text{cnn}}(v_i, v_j), \mathcal{P}_{\text{mot}}(v_i, v_j) \right].$$  \hfill (4.3)

Specifically, the HSV histogram similarity is calculated as $\mathcal{P}_{\text{col}}(v_i, v_j) = \varpi(h^-(v_i), h^+(v_j))$, where $\varpi(\cdot, \cdot)$ is the cosine similarity between the HSV histograms of the detections in the last frame of $v_i$ (i.e., $h^-(v_i)$) and the first frame of $v_j$ (i.e., $h^+(v_j)$). Moreover, the CNN feature similarity $\mathcal{P}_{\text{cnn}}(v_i, v_j)$ is calculated as

$$\mathcal{P}_{\text{cnn}}(v_i, v_j) = \frac{1 + \varpi(\mu^-(v_i), \mu^+(v_j))}{2},$$  \hfill (4.4)

where $\mu^-(v_i)$ and $\mu^+(v_j)$ are the CNN features of the detections in the last frame of $v_i$ and the first frame of $v_j$. Finally, the similarity between two bounding boxes based on the generalized KLT tracker \cite{326} is calculated as

$$\mathcal{P}_{\text{mot}}(v_i, v_j) = 1 - \frac{2}{1 + \exp \left( \frac{2\zeta(v_i, v_j)}{\gamma(B_{m_i}^i) + \gamma(B_{l_j}^j)} \right)},$$  \hfill (4.5)

where $\gamma(B_{m_i}^i)$ and $\gamma(B_{l_j}^j)$ are the areas of the detections in the last frame of $v_i$ and the first frame of $v_j$, and $\zeta(v_i, v_j)$ is the number of point trajectories generated by KLT tracker across the bounding boxes of both the first frame of $v_i$ and first frame of $v_j$.

**Calculating hyperedge affinity.** We count the number of local point trajectories passing through the regions of $v_{1:d}$ to calculate the affinities of hyperedges, which encodes the motion consistency of tracklets $v_{1:d}$. Thus, for the $i$-th hyperedge with degree $d$, the affinity is calculated as

$$\mathcal{A}(v_{1:d}) = 1 - \frac{2}{1 + \exp \left( \frac{d\zeta(v_{1:d})}{\sum_{u=1}^{d} \sum_{j=1}^{l_u} \gamma(B_{l_j}^j)} \right)},$$  \hfill (4.6)

where $\zeta(v_{1:d})$ measures the number of local point trajectories crossing all regions of $v_{1:d}$, $l_u$ is the length of tracklet $v_u$, $B_{l_j}^j$ is the $j$-th detection on $v_u$, and $\gamma(B_{l_j}^j)$ is the area of the detection $B_{l_j}^j$. 

51
4.3.3 Near-online tracking

It is difficult to handle all detections in a long video sequences at a time, since it requires large memory and computation sources to construct non-uniform hypergraphs and perform dense structure search on all detections. In order to achieve both accuracy and efficiency, inspired by [41], we use a near-online strategy for MOT. Specifically, after getting $\tau$ video frames at time $t$, we construct a non-uniform hypyergraph to describe the hybrid orders of dependencies among detections and search the dense structures on the hypergraph to generate short tracklets in the temporal window $[t - \tau, t]$. Then, we construct a conventional graph$^3$ to describe the associations between the tracked targets and the short tracklets within $[t - \tau, t]$. After that, we perform the dense structure searching on the conventional graph to associate the short tracklets and the tracked targets to get the final trajectories at the current time stamp. This process is carried out repeatedly every $\tau$ frames to complete the tracking task in the whole video.

4.3.4 Inference

For efficiency, we use the simple pairwise update algorithm [174] to solve the dense structure searching problem on hypergraph $G$ corresponding to node $v_s$ in (4.2). We first form the Lagrangian of the problem as

$$
\mathcal{L}(y, a, b, c) = \Theta(y) - a \cdot \left( \sum_{i=1}^{\left| \mathcal{N}(v_s) \right|} y_i - 1 \right)
+ \sum_{i,i \neq v_s} b_i \cdot y_i + \sum_{i,i \neq v_s} c_i \cdot \left( \frac{1}{\alpha} - y_i \right),
$$

(4.7)

where $a, b = (b_1, \cdots, b_{\left| \mathcal{N}(v_s) \right|})$, and $c = (c_1, \cdots, c_{\left| \mathcal{N}(v_s) \right|})$ are Lagrangian multipliers with $a \geq 0$, $b_i \geq 0$, and $c_i \geq 0$, $i = 1, \cdots, \left| \mathcal{N}(v_s) \right|$. Any local maximizer $y^*$ of the objective function must satisfy the Karush-Kuhn-Tucker (KKT) conditions [138], i.e.,

$$
\begin{cases}
\frac{\partial \Theta(y^*)}{\partial y_i} - a + b_i - c_i = 0, \quad i \neq v_s; \\
\sum_{i,i \neq v_s} y_i^* \cdot b_i = 0; \\
\sum_{i,i \neq v_s} c_i \cdot \left( \frac{1}{\alpha} - y_i^* \right) = 0; \\
a \geq 0, \quad b_i \geq 0, \quad c_i \geq 0, \quad i = 1, \cdots, \left| \mathcal{N}(v_s) \right|; \\
\sum_{i=1}^{\left| \mathcal{N}(v_s) \right|} y_i = 1, \quad y_i \in [0, \frac{1}{\alpha}].
\end{cases}
$$

(4.8)

$^3$The conventional graph is a special case of the non-uniform hypergraph, which only includes the conventional edges in the graph, i.e., $D = 2$. 

52
We define $\phi_i(y) = \frac{\partial \Theta(y)}{\partial y_i}$ as reward at node $v_i$, which is calculated as

$$\phi_i(y) = A(i) + \sum_{d=2}^D \lambda_d \sum_{v_{1:d-1} \in N(v_s)} A(v_{1:d-1}, i) \prod_{j=1}^{d-1} y_{v_j}.$$ 

Since $\forall i, y_i^* \geq 0, b_i \geq 0, \sum_{i \neq v_s} y_i^* \cdot b_i = 0$, we have that if $y_i^* > 0$, then $b_i = 0$. Meanwhile, since $\forall i, c_i \geq 0$, and $y_i^* \leq \frac{1}{a}$, we have that if $0 < y_i^* < \frac{1}{a}$, then $c_i = 0$. In this way, for node $i \neq v_s$, the KKT conditions can be further rewritten as

$$\phi_i(y) = \begin{cases} 
\leq a, & y_i^* = 0, i \neq v_s; \\
= a, & 0 < y_i^* < \frac{1}{a}, i \neq v_s; \\
\geq a, & y_i^* = \frac{1}{a}, i \neq v_s.
\end{cases} \quad (4.9)$$

Based on $y$ and $\alpha$, we can partition the solution space into three disjoint subsets, $\Omega_1(y) = \{i | y_i = 0\}, \Omega_2(y) = \{i | y_i \in (0, \frac{1}{a})\}$, and $\Omega_3(y) = \{i | y_i = \frac{1}{a}\}$. Thus, similar to Theorem 1 in [174], we find that there exists an appropriate $a$, such that (1) the rewards at all node in $\Omega_1(y)$ are no larger than $a$; (2) the rewards at all nodes in $\Omega_2(y)$ are equal to $a$; and (3) the rewards at all nodes in $\Omega_3(y)$ are larger than $a$.

A simple pairwise updating method is used to optimize (4.2). That is, we can increase one component $y_p$ and decrease another one $y_q$ appropriately, to increase the objective $\Theta(y)$. To be specific, we first introduce another variable $y_i'$ that is defined as: $y_i' = y_i$, for $l \neq p$ and $l \neq q$; $y_i' = y_i + \eta$, for $l = p$; and $y_i' = y_i - \eta$, for $l = q$, where $y' = (y_1', \ldots, y_{\|N(v_s)\|}')$ is the updated indicator variable in optimization process. Then, the change of objective after updating is

$$\Delta \Theta(y) = \Theta(y') - \Theta(y) = \varphi_{p,q}(y) \cdot \eta^2 + (\phi_p(y) - \phi_q(y)) \cdot \eta, \quad (4.10)$$

where $\varphi_{p,q}(y) = -\lambda_2 \cdot A(p, q) - \sum_{d=3}^D \lambda_d \sum_{v_{1:d-2} \neq p, q} A(v_{1:d-2}, p, q) \prod_{j=1}^{d-2} y_{v_j}$. 

53
To maximize the objective difference $\Delta \Theta(y)$, we select the updating step $\eta$ as follows:

$$\eta = \begin{cases} 
\min(y_q, \frac{1}{\alpha} - y_p), & \text{if } \varphi_{p,q}(y) \geq 0; \\
\min \left( y_q, \frac{1}{\alpha} - y_p, \frac{\phi_p(y) - \phi_q(y)}{2\varphi_{p,q}(y)} \right), & \text{if } \varphi_{p,q}(y) < 0; \\
\min(y_q, \frac{1}{\alpha} - y_p), & \text{if } \phi_p(y) = \phi_q(y), \varphi_{p,q}(y) > 0.
\end{cases}$$  \hspace{1cm} (4.11)

The Proof of Calculating the Updating Step $\eta$. We present the proof of calculating the updating step $\eta$. As discussed in the work, the objective of dense structure searching on non-uniform hypergraph is defined as

$$\Theta(y) = \sum_{d=1}^{D} \lambda_d \sum_{v_{1:d} \in \mathcal{N}(v_s)} A(v_{1:d}) \prod_{j=1}^{d} y_{v_j}. \hspace{1cm} (4.12)$$

We use the pairwise updating scheme to search the dense structures on the hypergraph to complete the tracking task. Specifically, we increase one component $y_p$ and decrease another one $y_q$ appropriately, to increase $\Theta(y)$, i.e.,

$$y'_l = \begin{cases} 
y_l, & l \neq p, l \neq q; \\
y_l + \eta, & l = p; \\
y_l - \eta, & l = q,
\end{cases} \hspace{1cm} (4.13)$$

where $y' = (y'_1, \cdots, y'_{|\mathcal{N}(v_s)|})$ is the updated indicator variable in the optimization process, and $l = 1, \cdots, |\mathcal{N}(v_s)|$. 

\[\text{In general, we can assume } \phi_p(y) > \phi_q(y). \text{ When } \phi_p(y) < \phi_q(y), \text{ we can exchange indexes } p \text{ and } q \text{ to maximize } \Delta \Theta(y).\]
The objective with the updated indicator variable is calculated as:

\[ \Theta(y') = \lambda_1 \sum_{v_i \neq p,q} A(v_i) y_{v_i} + \lambda_1 A(p)(y_p + \eta) + \lambda_1 A(q)(y_q - \eta) \]

\[ + \lambda_2 \sum_{v_i, v_j \neq p,q} A(v_i, v_j) y_{v_i} y_{v_j} + \lambda_2 \sum_{v_i \neq p,q} A(v_i, p) y_{v_i} (y_p + \eta) \]

\[ + \lambda_2 \sum_{v_i \neq p,q} A(v_i, q) y_{v_i} (y_q - \eta) + \lambda_2 A(p, q)(y_p + \eta)(y_q - \eta) \]

\[ + \sum_{d=3}^{D} \lambda_d \sum_{v_{1:d} \neq p,q} A(v_{1:d}) \prod_{j=1}^{d} y_{v_j} \]

\[ + \sum_{d=3}^{D} \lambda_d \sum_{v_{1:d-1} \neq p,q} A(v_{1:d-1}, p)(y_p + \eta) \prod_{j=1}^{d-1} y_{v_j} \]

\[ + \sum_{d=3}^{D} \lambda_d \sum_{v_{1:d-1} \neq p,q} A(v_{1:d-1}, q)(y_q - \eta) \prod_{j=1}^{d-1} y_{v_j} \]

\[ + \sum_{d=3}^{D} \lambda_d \sum_{v_{1:d-2} \neq p,q} A(v_{1:d-2}, p, q)(y_p + \eta)(y_q - \eta) \prod_{j=1}^{d-2} y_{v_j} \]  \hspace{1cm} (4.14)
The difference of objective after updating is

\[
\Delta \Theta(y) = \Theta(y') - \Theta(y) = \left( \lambda_1 A(p) - \lambda_1 A(q) \right) \cdot \eta
+
\left( \lambda_2 \sum_{v_i \neq p} A(v_i, p) y_{vi} - \lambda_2 \sum_{v_i \neq q} A(v_i, q) y_{vi} \right) \cdot \eta - \lambda_2 A(p, q) \cdot \eta^2
\]
\[+
\sum_{d=3}^{D} \lambda_d \sum_{v_{1:d-1} \neq p} A(v_{1:d-2}, p) \prod_{j=1}^{d-1} y_{vj}
- \sum_{d=3}^{D} \lambda_d \sum_{v_{1:d-1} \neq q} A(v_{1:d-2}, q) \prod_{j=1}^{d-1} y_{vj} \cdot \eta
\]
\[-
\sum_{d=3}^{D} \lambda_d \sum_{v_{1:d-2} \neq p, q} A(v_{1:d-1}, p) \prod_{j=1}^{d-2} y_{vj} \cdot \eta^2
\]

\[
\Delta \Theta(y) = -\left( \lambda_2 A(p, q) + \sum_{d=3}^{D} \lambda_d \sum_{v_{1:d-2} \neq p, q} A(v_{1:d-2}, p, q) \prod_{j=1}^{d-2} y_{vj} \right) \cdot \eta^2
\]
\[+
\left( \lambda_1 A(p) - \lambda_1 A(q) + \sum_{d=2}^{D} \lambda_d \sum_{v_{1:d-1} \neq p} A(v_{1:d-1}, p) \prod_{j=1}^{d-1} y_{vj} \right)
\]
\[- \sum_{d=2}^{D} \lambda_d \sum_{v_{1:d-1} \neq q} A(v_{1:d-1}, q) \prod_{j=1}^{d-1} y_{vj} \cdot \eta.
\]  
(4.15)

Then, we rewrite the difference of objective as

\[
\Delta \Theta(y) = \varphi_{p,q}(y) \cdot \eta^2 + (\phi_p(y) - \phi_q(y)) \cdot \eta,
\]  
(4.16)

where

\[
\varphi_{p,q}(y) = -\lambda_2 \cdot A(p, q)
\]
\[- \sum_{d=3}^{D} \lambda_d \sum_{v_{1:d-2} \neq p, q} A(v_{1:d-2}, p, q) \prod_{j=1}^{d-2} y_{vj},
\]  
(4.17)

\[
\phi_p(y) = \lambda_1 A(p) + \sum_{d=2}^{D} \lambda_d \sum_{v_{1:d-1} \in N(v_s)} A(v_{1:d-1}, p) \prod_{j=1}^{d-1} y_{vj}.
\]  
(4.18)

As discussed in the work, we select an appropriate updating step \( \eta \) to maximize the objective

56
difference $\Delta \Theta(\eta)$\footnote{When $\varphi_{p,q}(\eta) = 0$ and $\phi_p(\eta) = \phi_q(\eta)$, we have $\Delta \Theta(\eta) = 0$. We can not select any $\eta$ to increase the objective. Thus, we ignore this case in discussion.}. Based on the updating strategy presented in (4.13), we have two constraints of $\eta$, i.e., $0 \leq y_p' = y_p + \eta \leq \frac{1}{\alpha}$, and $0 \leq y_q' = y_q - \eta \leq \frac{1}{\alpha}$. Since $0 \leq y_p \leq \frac{1}{\alpha}$ and $0 \leq y_q \leq \frac{1}{\alpha}$, we have $\eta \leq y_q$, and $\eta \leq \frac{1}{\alpha} - y_p$. Notably, in general, we can assume $\phi_p(\eta) \geq \phi_q(\eta)$. When $\phi_p(\eta) < \phi_q(\eta)$, we can exchange indexes $p$ and $q$ to maximize $\Delta \Theta(\eta)$. In this way, we can select the updating step $\eta$ as follows:

- if $\varphi_{p,q}(\eta) \geq 0$, we have $\Delta \Theta(\eta) = \varphi_{p,q}(\eta) \cdot \eta^2 + (\phi_p(\eta) - \phi_q(\eta)) \cdot \eta$. To maximize $\Delta \Theta(\eta)$, we have to satisfy the constraints of $\eta$, i.e., $\eta \leq y_q$, and $\eta \leq \frac{1}{\alpha} - y_p$. We set $\eta = \min(y_q, \frac{1}{\alpha} - y_p)$.
- if $\varphi_{p,q}(\eta) > 0$ and $\phi_p(\eta) = \phi_q(\eta)$, we have $\Delta \Theta(\eta) = \varphi_{p,q}(\eta) \cdot \eta^2$. To maximize $\Delta \Theta(\eta)$, we have to satisfy the constraints of $\eta$, i.e., $\eta \leq y_q$, and $\eta \leq \frac{1}{\alpha} - y_p$. We set $\eta = \min(y_q, \frac{1}{\alpha} - y_p)$.
- if $\varphi_{p,q}(\eta) < 0$, we have $\Delta \Theta(\eta) = \varphi_{p,q}(\eta) \cdot \left(\eta + \frac{\phi_p(\eta) - \phi_q(\eta)}{2 \varphi_{p,q}(\eta)}\right)^2 - \frac{(\phi_p(\eta) - \phi_q(\eta))^2}{4 \varphi_{p,q}(\eta)}$. To maximize $\Delta \Theta(\eta)$ and satisfy the constraints of $\eta$, i.e., $\eta \leq y_q$, and $\eta \leq \frac{1}{\alpha} - y_p$, we set $\eta = \min\left(y_q, \frac{1}{\alpha} - y_p, \frac{\phi_q(\eta) - \phi_p(\eta)}{2 \varphi_{p,q}(\eta)}\right)$.

We use a heuristic strategy to compute a local maximizer $y^*$ of (4.2), i.e., gradually select pairs of nodes $(v_p, v_q)$ to maximize the increase of $\Theta(y)$ by updating the indicator variable $y$ based on the updating step $\eta$ calculated by (4.11). Specifically, from (4.10) and (4.11), we find that (1) if $\phi_p(y) > \phi_q(y)$, there exists $\alpha$ such that the objective $\Theta(y)$ can be increased by updating $y$ based on (4.10); (2) when $\phi_p(y) = \phi_q(y)$ and $\varphi_{p,q}(y) > 0$, the objective $\Theta(y)$ can be increased by increasing either $y_p$ or $y_q$, and decreasing the other one; (3) when $\phi_p(y) = \phi_q(y)$ and $\varphi_{p,q}(y) = 0$, the objective $\Theta(y)$ will not be affected by changing $y$.

Thus, in each iteration, we can select node $v_p$ with the largest reward from set $\Omega_1 \cup \Omega_2$, i.e., $v_p \in \Omega_1 \cup \Omega_2$, and node $v_q$ with the smallest reward from set $\Omega_2 \cup \Omega_3$, i.e., $v_q \in \Omega_2 \cup \Omega_3$, satisfying $\phi_p(y) > \phi_q(y)$, to increase $\Theta(y)$ by increasing $y_p$ and decreasing $y_q$ with an appropriate $\eta$ in (4.11). This process is iterated until the reward of $v_p$ equals to $v_q$. If $\Theta(y)$ can not be increased according to (4.10), then $y$ is already a local maximizer. The overall procedure is summarized in Algorithm 1.
Algorithm 1 Compute the local maximizer $y^*$

Require: The affinity set $A$ corresponding to the hyperedges in $G$, the starting point $y^o = (y^o_1, \cdots, y^o_{|N(v_s)|})$ and the minimal size of sub-hypergraph $\hat{\alpha}$.

1: Initialize the indicator variable $y = y^o$.

2: while $y$ is the local maximizer do

3: Select $v_p \in \Omega_1 \cup \Omega_2$ with the largest reward $\phi_p(y)$;

4: Select $v_q \in \Omega_2 \cup \Omega_3$ with the smallest reward $\phi_q(y)$;

5: if $\phi_p(y) > \phi_q(y)$ then

6: Compute $\eta$ according to (4.11), update $y$ and the corresponding rewards.

7: else if $\phi_p(y) = \phi_q(y)$ then

8: Find another pair of nodes $(v_i, v_j)$ satisfying $\phi_{i,j}(y) > 0$ and $\phi_i(y) = \phi_j(y)$, where $v_i \in \Omega_1 \cup \Omega_2$ and $v_j \in \Omega_2 \cup \Omega_3$.

9: if such a pair exists then

10: Compute the corresponding $\eta$ according to (4.11).

11: Update $y$ and the corresponding rewards.

12: else

13: $y$ is already a local maximizer, i.e., $y^* = y$.

14: end if

15: end if

16: end while

Ensure: The local maximizer indicator variable $y^*$.

4.3.5 Learning

Instead of selecting the weights $\lambda = (\lambda_1, \cdots, \lambda_D)$ in (4.1) empirically, we use a structured SVM [120] to learn $\lambda$ automatically from the training data. Specifically, given a set of ground-truth bounding boxes of objects in the $j$-th training video ($1 \leq j \leq U$, where $U$ is the total number of training videos), we aim to recover the trajectories of objects, which is equivalent to cluster the input bounding boxes into several groups. That is to obtain the indicator variables of the clusters $Y_j = (y_{1,j}, \cdots, y_{k_j,j})$, where $y_{i,j}$ ($1 \leq i \leq k_j$) is the indicator variable of the $i$-th target, and $k_j$ is the total number of targets in the video. The bounding boxes in each group belong to the same target.

The function defined in (4.1) can be rewritten as a linear function of $\lambda$, i.e., $\Theta(Y_j) = \lambda^T \cdot S(Y_j)$, where

$$S(Y_j) = \left[ \sum_{i=1}^{k_j} \sum_{v_i \in V} A(v_i) y_{i,i}, \cdots, \sum_{i=1}^{k_j} \sum_{v_{1:D} \in V} A(v_{1:D}) \prod_{i=1}^{D} y_{i,i} \right].$$ (4.19)

We aim to find the optimal weights $\lambda$ by maximizing the objective function $\Theta(Y_j)$ with the same
input object detections. Then, the objective using a SSVM with margin rescaling is formulated as

\[
\min_{\lambda} \frac{1}{2} \|\lambda\|_2 + C \cdot \sum_{j=1}^{U} \xi_j,
\]

s.t. \( \lambda^T \left( S(Y_j^*) - S(Y_j) \right) + \xi_j \geq \Delta(Y_j, Y_j^*), \)

\( \xi_j \geq 0, \quad j = 1, \ldots, U. \)

Intuitively, this formulation requires that the score \( \lambda^T \cdot S(Y_j^*) \) of any ground-truth annotated video must be larger than the score \( \lambda^T \cdot S(Y_j) \) of any other results \( Y_j \) by the loss \( \Delta(Y_j, Y_j^*) \) minus the slack variable \( \xi_j \). The constant \( C \) adjusts the importance of minimizing the slack variables. The loss function \( \Delta(Y_j, Y_j^*) \) measures how incorrect \( Y_j \) is according to the weighted Hamming loss in [273]. Meanwhile, the SSVM formulation in (4.20) has exponential number of constraints for each training sequence. We use a cutting plane algorithm [120] to solve this problem, which has time complexity linear in the number of training examples.

4.4 Experiments

to evaluate the performance of the proposed MOT method (denoted as NT subsequently), we conduct experiments on several popular MOT evaluation datasets, i.e., the multi-pedestrian tracking [281] (including the PETS09 and ParkingLot sequences), MOT2016 [192], and multi-face tracking [281] datasets. To extract the CNN features of detections, we fine-tune the GoogLeNet [250] based set to set recognition model [177] pre-trained on the ILSVRC CLS-LOC dataset [136] in the MOT16 training set. Specifically, we divide the ground truth trajectories of pedestrians equally to form the two-view structure of [177] in training. We optimize the network [177] using the Stochastic Gradient Descent (SGD) algorithm with 0.9 momentum and 0.0002 weight decay on a Titan X GPU. We set the learning rate to 0.001 for 120k iterations with a mini-batch of size 24. In addition, we conduct the ablation study to demonstrate the effectiveness of non-uniform hypergraph and SSVM learning.

4.4.1 Evaluation Metrics

Following previous MOT methods, we use the widely adopted multi-object tracking accuracy (MOTA) metric [19] to compare the performance of the trackers. MOTA is a cumulative measure comibing false negatives (FN), false positives (FP), and identity switches (IDS). We report
mostly tracked (MT), mostly lost (ML), FP, FN, IDS, and the fragmentation of the tracked objects (FM) to measure a tracker comprehensively. In addition, for the multi-pedestrian and multi-face tracking datasets [281], we also report the multi-object tracking precision (MOTP) score, which computes the total error of tracked positions comparing with the manually annotated ground-truth, with normalization to the hit/miss threshold value. Following the evaluation protocol in MOT2016, we use the ID F1 score (IDF1) [226] instead of MOTP, which is the ratio of correctly identified detections over the average number of ground-truth and computed detections.

4.4.2 Implementation Details

We conduct an experiment to select the maximal degree of the hypergraph $D$. We set $D = 2, \cdots, 5$ while keeping other parameters fixed, and denote the resulting models as NT$_d$(2), $\cdots$, NT$_d$(5). For each maximal degree, we use the sequences in the training set of MOT2016 to learn the weights of different degrees of hyperedges $\lambda = (\lambda_1, \cdots, \lambda_D)$ using SSVM, and use the sequences in multi-pedestrian tracking dataset for testing. The uniform average performance of the trackers in multi-pedestrian tracking dataset is presented in Table 4.1. Specifically, we divide each sequence in the MOT2016 train-set into non-overlapping sequences of 14 frames. And then, we take the detections that have more than 50% overlap with the ground-truth as true detections to collect training samples for the weights $\lambda$ learning.

As shown in Table 4.1, NT achieves the best performance with the maximal degree $D = 4$, indicated by higher MOTA and lower IDS and FM scores. We notice that the performance of NT decreases when $D > 4$, this may be because the hypergraph with excessive high degree fails to describe the motion patterns of objects well, particularly for the objects moving fast with drastic variations of directions. Thus, we set $D = 4$ in our experiments, and the learned weights of different degree of hyperedge are $\lambda_1 = 0.58535$, $\lambda_2 = [0.15576, 3.0332, 0.34388]$, $\lambda_3 = 1.2879$, and $\lambda_4 = 0.22324$. The batch size $\tau$ in near-online tracking is set to 7. The minimal size of the sub-hypergraph is set as $\hat{\alpha} = 2$. We fix all parameters to these values in the experiments.

4.4.3 Ablation Study.

To demonstrate the contribution of non-uniform hypergraph, we construct two variants of the proposed NT tracker by removing the hyperedges with certain degrees, i.e., NT$_r$(3) and NT$_r$(4), and evaluate them on the multi-pedestrian tracking dataset [281], shown in Table 4.1. The results
Table 4.1: Comparisons of variants of the proposed NT tracker on multi-pedestrian tracking dataset.

<table>
<thead>
<tr>
<th>Variants</th>
<th>$D$</th>
<th>$\lambda$</th>
<th>MOTA</th>
<th>MOTP</th>
<th>IDS</th>
<th>FM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NT_d(2)</td>
<td>2</td>
<td>learned</td>
<td>67.5</td>
<td>62.4</td>
<td>103.7</td>
<td>92.2</td>
</tr>
<tr>
<td>NT_d(3)</td>
<td>3</td>
<td>learned</td>
<td>68.8</td>
<td>64.5</td>
<td>83.8</td>
<td>76.2</td>
</tr>
<tr>
<td>NT_d(4)</td>
<td>4</td>
<td>learned</td>
<td>68.9</td>
<td>65.0</td>
<td>68.3</td>
<td>68.8</td>
</tr>
<tr>
<td>NT_d(5)</td>
<td>5</td>
<td>learned</td>
<td>68.5</td>
<td>64.7</td>
<td>61.5</td>
<td>63.7</td>
</tr>
<tr>
<td>NT_r(4)</td>
<td>4</td>
<td>$\lambda_i = 0, i = 3$</td>
<td>68.4</td>
<td>63.5</td>
<td>72.7</td>
<td>74.2</td>
</tr>
<tr>
<td>NT_r(5)</td>
<td>5</td>
<td>$\lambda_i = 0, i = 3, 4$</td>
<td>67.6</td>
<td>63.5</td>
<td>64.3</td>
<td>66.0</td>
</tr>
<tr>
<td>NT_e(2)</td>
<td>2</td>
<td>$\lambda_i = 1, i = 1$, $i = 2$</td>
<td>67.1</td>
<td>62.6</td>
<td>103.7</td>
<td>87.0</td>
</tr>
<tr>
<td>NT_e(3)</td>
<td>3</td>
<td>$\lambda_i = 1, i = 1, \cdots, 3$</td>
<td>67.5</td>
<td>63.7</td>
<td>103.3</td>
<td>87.5</td>
</tr>
<tr>
<td>NT_e(4)</td>
<td>4</td>
<td>$\lambda_i = 1, i = 1, \cdots, 4$</td>
<td>67.4</td>
<td>63.7</td>
<td>104.0</td>
<td>86.7</td>
</tr>
<tr>
<td>NT_e(5)</td>
<td>5</td>
<td>$\lambda_i = 1, i = 1, \cdots, 5$</td>
<td>67.1</td>
<td>64.6</td>
<td>93.2</td>
<td>81.7</td>
</tr>
</tbody>
</table>

in Table 4.1 shows that removing the hyperedges with degrees 3 and 4 will negatively affect the performance (i.e., reduce 0.5% and 0.9% MOTA scores), which shows that exploiting different degrees of dependencies among objects is important for MOT performance.

Besides, to demonstrate the contribution of SSVM, in Table 4.1 we present the performance of non-uniform hypergraph based trackers with equal weights of different degrees of hyperedges in multi-pedestrian tracking, denoted as NT_e(2), \cdots, NT_e(5). The NT_d(i) methods perform consistently better than the NT_e(i) methods with the same maximal degrees, e.g., NT_d(2) vs. NT_e(2), and NT_d(5) vs. NT_e(5), where $i = 2, \cdots, 5$. The results show that using SSVM to learn the weights of hyperedges of different degrees can improve the performance.

4.4.4 Results on Multi-Pedestrian Tracking

We perform experiments for the multi-pedestrian tracking on five sequences from the PETS09 dataset [72]: S2L1 (795 frames), S2L2 (436 frames), S2L3 (240 frames), S1L1-1 (221 frames), and S1L1-2 (241 frames), and ParkingLot sequence from [312] (996 frames). These sequences are captured in the crowded surveillance scenes with frequent occlusions, abrupt motion, illumination changes, etc. Following [281, 6], we report the uniform average scores on different metrics over sequences of the proposed NT algorithm, as well as 5 state-of-the-art trackers, i.e., KSP [17], DPMF [212], CEM [5], DCT [6] and FH^2T [281], in Table 4.2. The tracking results of previous methods are taken from [281]. For fair and comprehensive comparisons, we use the same frame detections, ground-truth annotations as well as the evaluation protocol provided by the authors of
Table 4.2: Comparison of the proposed tracker with the previous trackers in multi-pedestrian tracking sequences.

<table>
<thead>
<tr>
<th>Method</th>
<th>MOTA</th>
<th>MOTP</th>
<th>MT[%]</th>
<th>ML[%]</th>
<th>FP</th>
<th>FN</th>
<th>IDS</th>
<th>FM</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSP</td>
<td>45.5</td>
<td>67.1</td>
<td>33.4</td>
<td>35.6</td>
<td>107.8</td>
<td>2223.2</td>
<td>42.2</td>
<td>49.8</td>
</tr>
<tr>
<td>DPMF</td>
<td>51.6</td>
<td>70.0</td>
<td>21.5</td>
<td>27.0</td>
<td>68.8</td>
<td>1897.0</td>
<td>61.8</td>
<td>80.7</td>
</tr>
<tr>
<td>CEM</td>
<td>55.7</td>
<td>66.6</td>
<td>30.1</td>
<td>21.7</td>
<td>127.3</td>
<td>1652.8</td>
<td>63.7</td>
<td>56.7</td>
</tr>
<tr>
<td>DCT</td>
<td>58.1</td>
<td>67.6</td>
<td>43.1</td>
<td>21.3</td>
<td>119.5</td>
<td>1610.2</td>
<td>64.2</td>
<td>53.2</td>
</tr>
<tr>
<td>FH\textsuperscript{T}</td>
<td>66.2</td>
<td>64.9</td>
<td>54.3</td>
<td>14.7</td>
<td>194.5</td>
<td>1150.8</td>
<td>45.2</td>
<td>73.7</td>
</tr>
<tr>
<td>NT</td>
<td>68.9</td>
<td>65.0</td>
<td>58.2</td>
<td>9.6</td>
<td>252.7</td>
<td>974.3</td>
<td>68.3</td>
<td>68.8</td>
</tr>
</tbody>
</table>

We train the set-to-set recognition method \[177\] based on the pre-trained GoogLeNet \[250\] on the training set of MOT2016 to extract the CNN features of the detections.

As shown in Table 4.2, we find that our NT tracker performs better than the state-of-the-art methods on several important metrics (e.g., MOTA, MT, and ML). Specifically, NT improves 2.7\% and 3.9\% average MOTA and MT scores, and reduces 5.1\% average ML score, against the second best tracker FH\textsuperscript{T} \[281\]. This may be attributed to that our method uses non-uniform hypergraph instead of uniform hypergraph in \[281\], especially for tracking in crowded scenes with different motions and frequent occlusions of objects. By the way, we notice that the FH\textsuperscript{T} method \[281\] performs better than the methods (e.g., DPMF \[212\] and DCT \[6\]), both only considering the similarities between pairs of tracklets (i.e., FH\textsuperscript{T} \[281\] produces 14.6\% and 8.1\% higher average MOTA score than DPMF \[212\] and DCT \[6\]), which indicates that exploiting the high-order similarities among multiple tracklets is crucial for MOT.

4.4.5 Results on MOT2016 Benchmark

The MOT2016 benchmark \[192\] is a collection of 14 video sequences (7/7 for training and testing, respectively), with a relatively high variations in object movements, camera motion, viewing angle and crowd density. The benchmark primarily focuses on pedestrian tracking. The ground-truths for testing set are strictly invisible to all methods, i.e., all results on testing set were submitted to the respective testing servers for evaluation. We use the training set to learn the parameters of the proposed algorithm, and submit our results on testing set for evaluation, shown in Table 4.3. For a fair comparison with the state-of-the-art MOT methods, we use the reference object detections provided by the benchmark \[192\]. We train the set to set recognition method \[177\] based on the pre-trained GoogLeNet \[250\] on the training set of MOT2016 to extract the
Table 4.3: Comparison of the proposed tracker with the state-of-the-art trackers in the test set of the MOT2016 benchmark (accessed on 08/18/2018).

<table>
<thead>
<tr>
<th>Method</th>
<th>MOTA</th>
<th>IDF1</th>
<th>MT [%]</th>
<th>ML [%]</th>
<th>FP</th>
<th>FN</th>
<th>IDS</th>
<th>FM</th>
<th>Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>online:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EAMTT</td>
<td>38.8</td>
<td>42.4</td>
<td>7.9</td>
<td>49.1</td>
<td>8,114</td>
<td>102,452</td>
<td>965</td>
<td>1,657</td>
<td>11.8</td>
</tr>
<tr>
<td>DCCRF</td>
<td>44.8</td>
<td>39.7</td>
<td>14.1</td>
<td>42.3</td>
<td>5,613</td>
<td>94,133</td>
<td>968</td>
<td>1,378</td>
<td>0.1</td>
</tr>
<tr>
<td>STAM</td>
<td>46.0</td>
<td>50.0</td>
<td>14.6</td>
<td>43.6</td>
<td>6,895</td>
<td>91,117</td>
<td>473</td>
<td>1,422</td>
<td>0.2</td>
</tr>
<tr>
<td>AMIR</td>
<td>47.2</td>
<td>46.3</td>
<td>14.0</td>
<td>41.6</td>
<td>2,681</td>
<td>92,856</td>
<td>774</td>
<td>1,675</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>offline:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quad</td>
<td>44.1</td>
<td>38.3</td>
<td>14.6</td>
<td>44.9</td>
<td>6,388</td>
<td>94,775</td>
<td>745</td>
<td>1,096</td>
<td>1.8</td>
</tr>
<tr>
<td>INT</td>
<td>45.4</td>
<td>37.7</td>
<td>18.1</td>
<td>38.7</td>
<td>13,407</td>
<td>85,547</td>
<td>600</td>
<td>930</td>
<td>4.3</td>
</tr>
<tr>
<td>MHT</td>
<td>45.8</td>
<td>46.1</td>
<td>16.2</td>
<td>43.2</td>
<td>6,412</td>
<td>91,758</td>
<td>590</td>
<td>781</td>
<td>0.8</td>
</tr>
<tr>
<td>NLPa</td>
<td>47.6</td>
<td>47.3</td>
<td>17.0</td>
<td>40.4</td>
<td>5,844</td>
<td>89,093</td>
<td>629</td>
<td>768</td>
<td>8.3</td>
</tr>
<tr>
<td>FWT</td>
<td>47.8</td>
<td>44.3</td>
<td>19.1</td>
<td>38.2</td>
<td>8,886</td>
<td>85,487</td>
<td>852</td>
<td>1,534</td>
<td>0.6</td>
</tr>
<tr>
<td>LMP</td>
<td>48.8</td>
<td>51.3</td>
<td>18.2</td>
<td>40.1</td>
<td>6,654</td>
<td>86,245</td>
<td>481</td>
<td>595</td>
<td>0.5</td>
</tr>
<tr>
<td>near-online:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOMT</td>
<td>46.4</td>
<td>53.3</td>
<td>18.3</td>
<td>41.4</td>
<td>9,753</td>
<td>87,565</td>
<td>359</td>
<td>504</td>
<td>2.6</td>
</tr>
<tr>
<td>Ours</td>
<td>47.5</td>
<td>43.6</td>
<td>19.4</td>
<td>36.9</td>
<td>13,002</td>
<td>81,762</td>
<td>1,035</td>
<td>1,408</td>
<td>0.8</td>
</tr>
</tbody>
</table>

CNN features of the detections.

In Table 4.3, we compare the proposed NT method to the state-of-the-art methods including EAMTT [233], Quad [247], MHT [125], STAM [43], NOMT [41], AMIR [230], NLPa [150], FWT [106], LMP [254], INT [141], and DCCRF [328]. Our NT method performs on par with the state-of-the-art trackers (e.g., FWT and LMP) in terms of tracking accuracy. Specifically, LMP uses additional person re-identification datasets to train a deep StackNet with body part fusion to associate pedestrians across frames, achieving the top tracking accuracy (i.e., 48.8% MOTA), while FWT incorporates multiple detectors to improve the tracking performance. In contrast to the aforementioned methods using complex appearance model, our NT algorithm focuses on exploiting different degrees of dependencies among tracklets to assemble various kinds of appearance and motion patterns. The appearance modeling strategies proposed in those methods are complementary to our NT tracker. Meanwhile, we notice that NT achieves better performance than the high-order information based MHT in terms of tracking accuracy (47.5% vs. 45.8%), which implies that exploiting adaptive dependencies among objects is important for MOT.
Table 4.4: Comparison of the proposed tracker with other state-of-the-art trackers in the SubwayFace dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>MOTA</th>
<th>MOTP</th>
<th>MT[%]</th>
<th>ML[%]</th>
<th>FP</th>
<th>FN</th>
<th>IDS</th>
<th>FM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEM</td>
<td>18.9</td>
<td>71.4</td>
<td>18.8</td>
<td>37.4</td>
<td>1185.3</td>
<td>4095.3</td>
<td>69.8</td>
<td>100.3</td>
</tr>
<tr>
<td>KSP</td>
<td>32.8</td>
<td>74.0</td>
<td>15.1</td>
<td>32.2</td>
<td>648.5</td>
<td>3589.3</td>
<td>70.0</td>
<td>82.3</td>
</tr>
<tr>
<td>DCT</td>
<td>37.6</td>
<td>73.7</td>
<td>25.5</td>
<td>12.6</td>
<td>1235.0</td>
<td>2691.0</td>
<td>66.3</td>
<td>59.3</td>
</tr>
<tr>
<td>DPMF</td>
<td>42.6</td>
<td>73.7</td>
<td>24.6</td>
<td>14.3</td>
<td>679.0</td>
<td>2858.3</td>
<td>62.8</td>
<td>74.0</td>
</tr>
<tr>
<td>FH\textsuperscript{T}</td>
<td>45.8</td>
<td>73.4</td>
<td>27.4</td>
<td>11.5</td>
<td>742.3</td>
<td>2634.0</td>
<td>43.0</td>
<td>57.3</td>
</tr>
<tr>
<td>NT</td>
<td>53.1</td>
<td>70.4</td>
<td>34.2</td>
<td>8.5</td>
<td>648.5</td>
<td>2292.8</td>
<td>37.5</td>
<td>36.3</td>
</tr>
</tbody>
</table>

4.4.6 Multi-Face Tracking

In addition to pedestrian tracking, we also evaluate NT on the SubwayFaces dataset used in [281]. The dataset consists of four sequences, namely S001, S002, S003, and S004 with 1,199, 1,000, 1,600, and 1,001 frames, captured from surveillance videos in subway with manually annotations. We compare our approach with five state-of-the-art MOT algorithms, i.e., CEM [5], KSP [17], DCT [6], DPMF [212] and FH\textsuperscript{T} [281], with uniform average scores on different metrics over sequences presented in Table 4.4. We use the same input detections, ground-truth annotations and the evaluation protocol as [281], and the results of the state-of-the-art trackers in Table 4.4 are taken from [281]. We use pre-trained AlexNet [136] to extract the CNN features of the detected faces.

As presented in Table 4.4, we find that our approach achieves the best performance on almost all evaluation metrics except MOTP. Specifically, the NT method produces 7.3\% and 6.8\% larger average MOTA and MT scores, and 3.0\% lower average ML score, comparing to the second best FH\textsuperscript{T} tracker. The evaluated sequences are recorded in the unconstrained scenes with fast motion, illumination variations, motion blurs and frequent occlusions. Since different degrees of dependencies among objects are considered, our method is able to exploit different types of motion patterns to improve the tracking performance, indicated by the consistent highest scores of almost all metrics (i.e., MOTA, MT, ML, FP, FN, IDS, and FM). Meanwhile, comparison with the state-of-the-art methods, our approach tracks the objects more robustly even when occlusions occur, indicated by the IDS, FM and FN scores. However, the linear interpolation is used in our method to estimate the occluded parts of the trajectories, which is not accurate enough to achieve good MOTP score, especially for crowded scenes containing non-linear motion patterns.
4.4.7 Qualitative Tracking Results

We present some qualitative results of the proposed NT algorithm in Figure 4.2, Figure 4.3, and Figure 4.4. More tracking results of our tracker are presented in the video demo. The proposed NT algorithm achieves good results mainly due to the introduction of non-uniform hypergraph learning in tracking task, which has much stronger descriptive power to accommodate different scenarios than the conventional graph or uniform hypergraph.
Figure 4.4: Tracking results of the proposed NT tracker in the test set of MOT2016 benchmark. The video frames are taken from the MOT2016 benchmark [192].

4.4.8 Running Time

We implement the NT algorithm in C++ without any code optimization. To demonstrate the running time of NT, we run it five times using a single thread on a laptop with a 2.8 GHz Intel processor and 16 GB memory. Given the detections with the corresponding CNN features, the average speeds on the multi-pedestrian tracking dataset, MOT2016 dataset, and multi-face tracking dataset are 7.9, 0.8, and 9.0 frame per second (FPS), respectively. In summary, it runs approximately 8 FPS with 20 targets in scenarios.

4.5 Conclusions

In this work, we propose a non-uniform hypergraph learning based near-online MOT method, which assembles different degrees of dependencies among tracklets in a unified objective. In contrast to previous graph or hypergraph based methods, our formulation exploit different high-degree cues among multiple tracklets in a computationally efficient way. Extensive experiments on several datasets, including the multi-pedestrian and multi-face tracking datasets, and MOT2016 benchmark, show that our method achieves comparable performance regarding to the state-of-the-arts.
CHAPTER 5
Space-Time Graph Convolutional Networks for Crowd Counting, Localization and Tracking

In this chapter, I go a step further to attempt to solve the crowd counting, localization and tracking tasks jointly using a space-time graph convolutional network, which combines multi-scale features in consecutive frames and uses the graph convolution operations instead of conventional convolution operation to exploit the context information of neighboring targets in association.

5.1 Motivation

Crowd counting attracts much research in recent years due to its wide range of applications, such as video surveillance [37], crowd analysis [327] and public safety [195]. It aims to estimate the number of objects in scenarios. However, several challenging factors such as heavy occlusion, large-scale variation, and perspective changes impede the deployments of such technology in real scenarios.

Recent methods formulate crowd counting as the density map estimation problem, \( i.e. \), pixel-wise regression task, and attempt to use deep convolutional networks to learn discriminative features for density map estimation. For example, Zhang et al. [321] propose the multi-column network formed by three branches with different kernel sizes to generate the density map. Liu et al. [175] develop an attention-injective deformable convolutional network, which contains two concatenated networks including the attention-aware network and the multi-scale deformable network. Despite great progress has been achieved, these algorithms still have room for improvement by integrating dependencies among objects. Meanwhile, the spatio-temporal context information in video sequences is also effective to improve the counting accuracy. Xiong et al. [295] propose a convolutional LSTM model to capture both the spatial and temporal dependencies for crowd counting. Zhang et al. [318] combine fully convolutional neural networks and LSTM using residual learning to perform vehicle counting.

In contrast to the aforementioned methods, I attempt to go a further step to jointly solve the
density map estimation, object localization, and tracking. In this way, these three tasks are able to help each other to improve the performance. Note that, there are few researches on object localization and tracking in crowded scenes. A Space-Time Graph Convolutional Network (STGNet) is designed to solve these three tasks, which is formed by four modules, i.e., the Siamese feature extraction subnetwork, the density map estimation branch, the localization branch, and the association branch. First, multi-scale feature maps in sequential frames are combined and enhanced by the U-Net style architecture [227] with deformable convolution to exploit temporal coherency across frames. After that, followed by the enhanced features, both counting and localization subnets are used to estimate counting and localization maps at different stage. We concatenate the multi-scale feature maps at different stage for final prediction in counting and localization subnets. Moreover, the association subnet is introduced to compute the correlations of feature maps to predict the motion offsets of objects for tracking. Meanwhile, the graph convolution operations are used to exploit the relations of neighboring objects for accurate motion prediction. The whole network is trained in an end-to-end manner with the multi-task loss and Adam optimizer [126]. Finally, the min-cost flow method [212] is used to generate long trajectories of targets for post-processing. We conduct comprehensive experiments on a few challenging datasets including Shanghaitech A and B [321] and UCF-QNRF [114], and DroneCrowd [280] to demonstrate the effectiveness of the proposed method.

To clarify my contributions in this joint work, my efforts are mainly devoted to algorithmic design as well as the implementation and experiments.

5.2 Related Work

Crowd counting. Different from traditional approach [288, 147, 268] using a sliding window detector to detect individual objects, recent crowd counting methods [321, 232, 166, 35, 187] formulate the task as the estimation of density maps instead of localizing individual objects in crowded scenes. In [321], the multi-column network (MCNN) formed by three branches using different kernel sizes is proposed to solve the large-scale variation problem in counting. Moreover, the Switching CNN model [232] trains several independent density regressors on the image patches, where each regressor has the same architecture with MCNN. Li et al. [166] use VGG16 as the front-end network for feature extraction and the dilated convolution layers to construct the back-end network. Cao et al. [35] design the encoder-decoder based scale aggregation network for
crowd counting, where the encoder uses scale aggregation to extract multi-scale features and the decoder uses transposed convolutions to generate high-resolution density maps. Ma et al.\cite{187} develop the Bayesian loss function to construct the density contribution probability model from the point annotations, achieving substantial improvements over the conventional loss functions.

Compared to crowd counting in images, videos are rich in temporal consistency information, which is effective to improve the accuracy. To exploit the temporal consistency in videos, Xiong et al.\cite{295} jointly capture both spatial and temporal dependencies using the bidirectional ConvLSTM model. Zhang et al.\cite{318} design the FCN-rLSTM network to jointly estimate the density and counts of vehicles using the connected FCN and LSTM models in the residual learning fashion. In \cite{76}, the locality-constrained spatial transformer network is proposed to relate the density maps of neighbouring frames for video crowd counting. Different from the aforementioned methods, our STGNet fuses multi-scale feature maps in sequential frames to learn discriminative representation for crowd counting, which is effective in exploiting the temporal coherency for better performance.

**Crowd localization and tracking.** Besides crowd counting, the crowd localization and tracking are also important but challenging tasks to determine the locations and trajectories of each individual object. However, there only exists a handful of methods \cite{186, 114, 178}. Ma et al.\cite{186} propose the 2D integer programming based method for joint object detection and counting. Idrees et al.\cite{114} design the composition loss to simultaneously solves the counting, density map estimation and localization tasks. Liu et al.\cite{178} develop the point-supervised deep detection network to detect the sizes and locations of human heads to count them in crowds.

In terms of crowd tracking, Ren et al.\cite{222} propose a people tracking framework that fuses the sparse kernelized correlation filter response map with an predicted crowd density map using the convolutional neural network (CNN). Moreover, a new tracking-by-counting strategy \cite{223} is developed for multi-object tracking in crowded scenes, which formulates the detection, counting, and tracking of objects as the network flow program. Recently, Wen et al.\cite{280} employ the non-maximal suppression with the min-cost flow framework to generate the trajectories of targets. In contrast to the aforementioned methods, our method focuses on modeling the context information among objects using the graph convolution operations to predict their motion offsets.

**Graph convolutional networks.** Graph Convolutional Network (GCN) is a powerful tool to model the relations among objects \cite{235}. Gilmer et al.\cite{89} propose the message passing neural networks operating on the undirected graphs. To solve the vanishing gradient problem in shallow models,
Li et al. [154] introduce the residual/dense connections and dilated convolutions to GCN. In [289], the convolution operation is applied on 3D point clouds, which trains multi-layer perceptrons on local point coordinates to approximate the continuous weight and density functions in convolutional filters efficiently. Similarly, Wang et al. [276] propose a new EdgeConv layer to deal with point clouds for classification and segmentation. EdgeConv incorporates local neighborhood information of graphs in each layer of the network. Recently, Luo et al. [182] propose the hybrid graph neural network to enhance the multi-scale features for crowd density estimation and localization. In contrast to previous methods using the task-specific feature maps of different scales as nodes, our method focuses on modeling the relations between object points in the images.

5.3 Space-Time Graph Convolutional Network

To solve the object localization, tracking and counting tasks, our STGNet takes two frames with $\tau$ frame gap as input, and predicts the density map, localization of objects, and the motion offsets of objects in these two frames. After that, the non-maximal suppression and min-cost flow association method [212] are used to generate the locations and long trajectories of objects in video sequences.

localize people’s head and generate their long trajectories in video sequences. Specifically, as shown in Figure 5.1 our model consists of the feature extractor, counting, localization, and association subnets.
5.3.1 Network architecture

The architecture of our STGNet is shown in Figure 5.1. As shown in Figure 5.1, our model is formed by four modules, i.e., the feature extractor, the density map estimation subnet, the localization subnet, and the association subnet. We will discuss each module in detail in the following sections.

**Feature extractor.** As shown in Figure 5.1, we use two branches of the first four groups of convolution layers in the VGG-16 network [242] with the shared parameters to construct the feature extraction subnetwork, which is effective to exploit the discriminative multi-scale features. Inspired by U-Net [227], we introduce the skip connections in our network, which includes a contracting path (left side) and an expansive path (right side).

Specifically, the contracting path is the first four convolution groups of the VGG-16 network [242], where each stage is followed by one $2 \times 2$ max pooling layer with stride 2 for downsampling. Notably, after the VGG-16 backbone, one $3 \times 3$ deformable convolutional layer [332] is used to enhance sufficient discriminative representation. For the expansive path, we use one transposed convolutional layer to upsample the resolution and halve the number of channels of the feature maps at each stage in the contracting path. To maintain efficiency, one $3 \times 3$ and $1 \times 1$ convolutional layers are followed to further halve the number of channels of concatenated feature maps with the corresponding feature maps from the contracting path.

**Counting and localization subnets.** Based on multi-scale feature maps, the goal of counting and localization subnets is to output the corresponding density and localization maps by three $3 \times 3$ convolutional layers, respectively. Supervised by ground-truth density and localization maps (discussed in Section 5.3.3), we can estimate both the density and accurate localization of people heads.

As shown in Figure 5.2, we concatenate different feature maps to collect multi-scale features, two of which are up-sampled by bilinear interpolation. The concatenated multi-scale feature maps are further fused for final prediction in counting and localization subnets. Specifically, inspired by the work of [287], the feature maps are re-weighted in terms of both channel and spatial attentions. After that, one $3 \times 3$ convolutional layer is used to predict the final density and localization maps.

**Association subnet.** Since the localization subnet focuses on regressing all the pixels in the map coarsely, we further develop the association subnet to calculate the offsets of targets in adjacent
frames for better performance. As shown in Figure 5.3, the input of the proposed branch includes fused localization maps and correlation maps, which is processed as follows.

- First, based on the merged localization map, we use one $3 \times 3$ max-pooling layer to obtain the points with local maximal confidence. After that, $K$ points are selected randomly to feed into the PointConv operation \[289\] for modeling context information.

- Second, based on the correlation map, we use one $3 \times 3$ convolutional layer to extract the corresponding features of selected points. Then, we feed the proposal points with the corresponding features into 3 PointConv layers and following Multi-layer Perceptron (MLP) to exploit the context information surrounding the object point. Note that the nearest $\alpha$ neighbouring points on the feature map are used in the PointConv operation.

Finally, we can output the locations and offsets of the targets in sequential frames.
5.3.2 Loss Function

The loss function in the proposed method is formed by three terms, including the density, localization, and association losses, i.e.,

\[
L = \frac{1}{N} \sum_{n=1}^{N} \left( \omega_{\text{den}} L_2(\hat{\Phi}_n, \Phi_n) + \omega_{\text{loc}} L_2(\hat{\Upsilon}_n, \Upsilon_n) \\
+ \omega_{\text{ass}} L_1(\hat{O}_{n,n+1}, O_{n,n+1}; P_n) \right),
\]

(5.1)

where \(N\) is the batch size. \(\hat{\Phi}_n\) and \(\Phi_n\) are the predicted and ground-truth density maps of the \(n\)-th image. \(\hat{\Upsilon}_n\) and \(\Upsilon_n\) are the predicted and ground-truth localization maps of the \(n\)-th image, and \(P_n\) is the set of selected points based on \(\hat{\Upsilon}_n\). \(\hat{O}_{n,n+1}\) and \(O_{n,n+1}\) are the predicted and ground-truth motion offsets of targets in the \(n\)-th and \((n + 1)\)-th samples in the mini-batch, respectively. \(\omega_{\text{den}}\), \(\omega_{\text{loc}}\) and \(\omega_{\text{ass}}\) are the pre-set balancing factors.

For the first two terms, we use the mean-squared loss to supervise density and localization maps. Both the density and localization losses are defined as

\[
\begin{align*}
L_2(\hat{\Phi}_n, \Phi_n) &= \sum_{s=1}^{S} \sum_{i=1}^{W_n^s} \sum_{j=1}^{H_n^s} \lambda_{\text{den}}^s \cdot \|\hat{\phi}^n(i, j, s) - \phi^n(i, j, s)\|_2^2, \\
L_2(\hat{\Upsilon}_n, \Upsilon_n) &= \sum_{s=1}^{S} \sum_{i=1}^{W_n^s} \sum_{j=1}^{H_n^s} \lambda_{\text{loc}}^s \cdot \|\hat{\upsilon}^n(i, j, s) - \upsilon^n(i, j, s)\|_2^2,
\end{align*}
\]

(5.2)

where \(H_n^s\) and \(W_n^s\) are the size of feature maps at the \(s\)-th stage of the \(n\)-th image. \(\lambda_{\text{den}}\) and \(\lambda_{\text{loc}}\) are the pre-set balancing factors for loss value at different stages. For the third term, we use the \(\ell_1\) loss is used to measure the differences between the predicted and ground-truth offsets of selected points. It is computed as follows.

\[
L_1(\hat{O}_{n,n+1}, O_{n,n+1}; P_n) = \sum_{k \in P_n} \|\hat{O}_{n,n+1}(P_n(k)) - O_{n,n+1}(P_n(k))\|_1,
\]

(5.3)

where \(\hat{O}_{n,n+1}(P_n(k))\) and \(O_{n,n+1}(P_n(k))\) are the normalized offsets of the predicted and ground-truth localization maps at point \(P_n(k)\) of the \(n\)-th image.
5.3.3 Ground-truth generation

According to [321], we generate ground-truth maps for density estimation and localization using Gaussian distributions. Specifically, for the density map, we calculate the Gaussian distribution with a normalized Gaussian kernel based on each object point. Then, the ground-truth density map is generated by adding the spatial distribution of all objects. The geometry-adaptive kernel for density map is calculated as

$$\Phi(x) = \sum_{i=1}^{K} \delta(x - x_i) * G(x, \sigma_i), \quad \text{s.t. } \sigma_i = \gamma \hat{d}_i,$$

where $\Phi(x)$ is the density map and $K$ is the number of objects in the image. The delta function $\delta(x - x_i)$ corresponds to the object at pixel $x_i$. $\hat{d}_i$ is the average distance between the object and its nearest neighbouring object. $\delta(x - x_i)$ is convolved with a Gaussian kernel with the standard deviation $\sigma_i = \gamma \hat{d}_i$ resulting in the ground-truth density map.

On the other hand, for the localization map, similar to [329], we combine the Gaussian distribution based on each object point with a fixed Gaussian kernel. Specifically, we take the maximal values for two overlapping Gaussian distributions. Notably, multi-scale ground-truth maps are generated for supervision of predicted maps at different stages.

5.3.4 Optimization

Data augmentation. The images in the training subset are randomly flipped and cropped to increase the diversity in the training phase. Due to the limitations of computational resources, for the images with the size larger than $1920 \times 1080$, we first resize the images to $1920 \times 1080$, and crop some image patches with the long side larger than 960 but smaller than 1920. For the video based dataset, we equally divide the video frames into $2 \times 2$ patches, and use the divided 4 patches for training.

Multi-stage Training. The whole network is trained in an end-to-end manner with the learning rate of $10^{-5}$ using the Adam optimization algorithm [126]. Notably, we use two stages to train the proposed model empirically. For the first stage in the training phase, we disable the point localization refinement module in the localization subnet and focus on generating coarse density

\[1\]We set $\gamma = 0.3$ in the experiment.
Table 5.1: Comparison of the proposed method to the state-of-the-art methods on three challenging datasets.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MCNN [321]</td>
<td>CVPR 2016</td>
<td>110.2 173.2</td>
<td>26.4 41.3</td>
<td>277.0 426.0</td>
</tr>
<tr>
<td>C-MTL [243]</td>
<td>AVSS 2017</td>
<td>101.3 152.4</td>
<td>20.0 31.1</td>
<td>252.0 514.0</td>
</tr>
<tr>
<td>SwitchCNN  [232]</td>
<td>CVPR 2017</td>
<td>90.4 135.0</td>
<td>21.6 33.4</td>
<td>228.0 445.0</td>
</tr>
<tr>
<td>CP-CNN [244]</td>
<td>ICCV 2017</td>
<td>73.6 106.4</td>
<td>20.1 30.1</td>
<td>- -</td>
</tr>
<tr>
<td>SaCNN [316]</td>
<td>WACV 2018</td>
<td>86.8 139.2</td>
<td>16.2 25.8</td>
<td>- -</td>
</tr>
<tr>
<td>ACSCP [237]</td>
<td>CVPR 2018</td>
<td>75.7 102.7</td>
<td>17.2 27.4</td>
<td>- -</td>
</tr>
<tr>
<td>IG-CNN [231]</td>
<td>CVPR 2018</td>
<td>72.5 118.2</td>
<td>13.6 21.1</td>
<td>- -</td>
</tr>
<tr>
<td>Deep-NCL  [240]</td>
<td>CVPR 2018</td>
<td>73.5 112.3</td>
<td>18.7 26.0</td>
<td>- -</td>
</tr>
<tr>
<td>CSRNet [166]</td>
<td>CVPR 2018</td>
<td>68.2 115.0</td>
<td>10.6 16.0</td>
<td>- -</td>
</tr>
<tr>
<td>CL-CNN [114]</td>
<td>ECCV 2018</td>
<td>- - -</td>
<td>- - 132.0 191.0</td>
<td></td>
</tr>
<tr>
<td>ic-CNN [216]</td>
<td>ECCV 2018</td>
<td>68.5 116.2</td>
<td>10.7 16.0</td>
<td>- -</td>
</tr>
<tr>
<td>SANet [35]</td>
<td>ECCV 2018</td>
<td>67.0 104.5</td>
<td>8.4 13.6</td>
<td>- -</td>
</tr>
<tr>
<td>SFCN [270]</td>
<td>CVPR 2019</td>
<td>64.8 107.5</td>
<td>7.6 13.0</td>
<td>102.0 171.4</td>
</tr>
<tr>
<td>ADCrowdNet [175]</td>
<td>CVPR 2019</td>
<td>63.2 98.9</td>
<td>7.6 13.9</td>
<td>- -</td>
</tr>
<tr>
<td>TEDnet [119]</td>
<td>CVPR 2019</td>
<td>64.2 109.1</td>
<td>8.2 12.8</td>
<td>113.0 188.0</td>
</tr>
<tr>
<td>STANet [280]</td>
<td>arXiv 2019</td>
<td>63.7 101.5</td>
<td>7.4 11.0</td>
<td>107.6 174.8</td>
</tr>
<tr>
<td>DUBNet [201]</td>
<td>AAAI 2020</td>
<td>64.6 106.8</td>
<td>7.7 12.5</td>
<td>105.6 180.5</td>
</tr>
<tr>
<td>STGNet</td>
<td>-</td>
<td>62.1 104.5</td>
<td>7.5 12.3</td>
<td>94.0 158.9</td>
</tr>
</tbody>
</table>

and localization maps in the first 200 epochs. For the second stage, we fix the parameters of U-Net style feature extractor and fine-tune both density and localization subnets with all the three loss terms in the next 100 epochs. For the training phase, we set the batch size $N = 2$ and $N = 1$ for video and image based datasets respectively.

### 5.4 Experiment

We evaluate the counting performance of the proposed method on four datasets, i.e., Shanghaitech Part A [321], Shanghaitech Part B [321], UCF-QNRF [114] and DroneCrowd [280]. Since the DroneCrowd dataset is the only one with trajectory annotations of people heads, we perform crowd localization and tracking on it. All experiments are conducted on a workstation with Intel Xeon Silver 4210R@2.40GHz CPU, 32GB RAM, and 2 NVIDIA GeForce GTX 2080Ti GPU cards.
5.4.1 Implementation details

To balance the loss values in (5.1), the pre-set weights are set as \( \omega_{\text{den}} = 1.0, \omega_{\text{loc}} = 0.0005 \) and \( \omega_{\text{ass}} = 0.05 \). In (5.2), we set the balancing factors to \( \lambda_{\text{den}} = \{0.5, 0.125, 0.025, 0.5\} \) and \( \lambda_{\text{loc}} = \{0.125, 0.5, 2.0, 0.125\} \). According to the previous works, we set the fixed Gaussian kernel to generate the ground-truth for the Shanghaitech Part B [321], UCF-QNRF [114] and DroneCrowd datasets, i.e., \( \sigma = 7.5 \) for density maps and \( \sigma = 5.0 \) for localization maps. The number of selected points in the point localization refinement module is set as \( K = 512 \). For the PointConv layers described in Section 5.3.1, we set the number of neighbouring points \( \alpha = 8 \).

5.4.2 Evaluation metrics

Crowd counting. According to the work of [314, 321], we use the mean absolute error (MAE) and mean squared error (MSE) to evaluate the accuracy and robustness of crowd counting methods, i.e.,

\[
\text{MAE} = \frac{1}{\sum_{c=1}^{C} N_c} \sum_{c=1}^{C} N_c \sum_{n=1}^{N_c} |Z(c,n) - \hat{Z}(c,n)|, \\
\text{MSE} = \sqrt{\frac{1}{\sum_{c=1}^{C} N_c} \sum_{c=1}^{C} N_c \sum_{n=1}^{N_c} |Z(c,n) - \hat{Z}(c,n)|^2},
\]  

(5.5)

where \( C \) is the number of video clips, \( N_c \) is the number of frames in the \( c \)-th video. \( Z(c,n) \) and \( \hat{Z}(c,n) \) are the ground-truth and predicted number of people in the \( n \)-th frame of the \( c \)-th video clip, respectively.

Crowd localization. Similar to [280], we use the mean average precision (L-mAP) at various distance thresholds (1, 2, 3, \cdots, 25 pixels) for a fair comparison, where false positive and negative detections are penalized. Meanwhile, the results with three specific distance thresholds are given, i.e., L-AP@10, L-AP@15, and L-AP@20 pixels.

Crowd tracking. According to the tracking evaluation protocol in [206, 280], we sort the output tracklets generated by each algorithm, based on the average confidence of their detections with the same identity. If the matched percentage between the predicted and ground-truth tracklets is larger than the threshold (i.e., 0.05, 0.10, and 0.20), the tracklet is considered to be correct. Note that the threshold of matching distance is set to 25 pixels. Finally, the mean average precision (T-mAP) scores with different thresholds (i.e., T-AP@0.05, T-AP@0.10, and T-AP@0.20) are used.
for evaluation.

5.4.3 Result Analysis on Crowd Counting

As presented in Table 5.1, the proposed STGNet is evaluated on the Shanghaitech A and B [321], and UCF-QNRF [114] datasets, compared with 17 state-of-the-art crowd counting methods. Besides, we compare our method with 11 existing approaches on DroneCrowd [280] dataset, as shown in Table 5.2. To deal with image based datasets, we have only one image as input. Then, we replace the input and output of the association module from correlation map as localization feature map and from point offsets to point confidence. In that way, the association subnet is formulated as the point localization refinement subnet.

Shanghaitech. The Shanghaitech [321] dataset is formed by two parts, i.e., Part A and Part B. In total, it includes 1,198 images and 330,165 annotated people heads. Specifically, Part A is collected from the Internet, most of which contains a large number of people; while Part B is extracted from busy streets of metropolitan areas in Shanghai.

As shown in Table 5.1, our method outperforms most compared crowd counting methods and achieve 62.1 MAE and 104.5 MSE on Part A, and 7.5 MAE and 12.3 MSE on Part B. STANet [280] performs slightly better than our method on Part B, but worse than our method on Part A. The results indicate the effectiveness of our method.

UCF-QNRF. The UCF-QNRF [114] dataset annotates 1,251,642 people head in 1,535 HD images. It is formed by training set (1,201 images) and testing set (334 images). Compared with Shanghaitech [321], the images in UCF-QNRF [114] have larger resolution, which are collected from various scenes with diversified viewpoints, densities and lighting variations.

From Table 5.1 it can be observed that our STGNet achieves 94.0 MAE and 158.9 MSE. It is attributed to the proposed point localization refinement subnet (degraded from the association subnet), resulting in considerable improvement than compared methods [321, 243, 232, 114, 270, 119, 280, 201].

DroneCrowd. The DroneCrowd dataset consists of the training and testing sets, with 82 and 30 sequences, respectively. Meanwhile, ten video-level attributes are defined, i.e., Cloudy, Sunny, and Night for the illumination condition, Large and Small for the size of objects, and Crowded and Sparse for the density level.
As presented in Table 5.2, our STGNet performs the state-of-the-art compared to existing methods. For example, we achieve the best 16.5 MAE score in comparison to the second best STANet [280] and the third best CSRNet [166]. It indicates that our method can output more accurate density maps for crowd counting. Moreover, we report the results in terms of different visual attributes. It is worth mentioning that our method performs not well on the Night subset. This is because there is not enough context information in night scenes. In contrast, our method performs well in terms of Cloudy and Sunny by extracting more discriminative features. In summary, our STGNet merges the correlated multi-scale feature maps by graph convolution, resulting in better performance.

### 5.4.4 Result Analysis on Crowd Localization and Tracking

As discussed in Section 5.3, similar as [280], we extract target points based the localization maps in each frame for crowd localization. We also extend two top crowd counting methods including STANet [280] and CSRNet [166] for localization and tracking by finding local peaks on the density maps using a preset threshold, and applying the min-cost flow algorithm [212] to generate the corresponding trajectories. We also show some qualitative results in Figure 5.4.

**Crowd Localization.** As shown in Table 5.3, our STGNet-L obtains the best localization accuracy of 36.98% L-mAP score, significantly better that the compared methods including STANet [280] and CSRNet [166]. It indicates that our method can generate more accurate target points based on the localization maps. However, it still remains much room of improvement for crowd localization.

<table>
<thead>
<tr>
<th>Method</th>
<th>Overall MAE</th>
<th>Large MAE</th>
<th>Small MAE</th>
<th>Cloudy MAE</th>
<th>Sunny MAE</th>
<th>Night MAE</th>
<th>Crowded MAE</th>
<th>Sparse MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCNN [21]</td>
<td>34.7 42.5</td>
<td>36.8 44.1</td>
<td>31.7 40.1</td>
<td>21.0 27.5</td>
<td>39.0 43.9</td>
<td>67.2 68.7</td>
<td>29.5 35.3</td>
<td>37.7 46.2</td>
</tr>
<tr>
<td>C-MTL [243]</td>
<td>56.7 65.9</td>
<td>53.5 63.2</td>
<td>61.5 69.7</td>
<td>59.5 66.9</td>
<td>56.6 67.8</td>
<td>48.2 58.3</td>
<td>81.6 88.7</td>
<td>42.2 47.9</td>
</tr>
<tr>
<td>MSCNN [313]</td>
<td>58.0 75.2</td>
<td>58.4 77.9</td>
<td>57.5 71.1</td>
<td>64.5 85.8</td>
<td>53.8 65.5</td>
<td>46.8 57.3</td>
<td>91.4 106.4</td>
<td>38.7 48.8</td>
</tr>
<tr>
<td>LFCCN [143]</td>
<td>136.9150.6</td>
<td>126.3140.3</td>
<td>152.8164.8</td>
<td>147.1160.3</td>
<td>137.1151.7</td>
<td>105.6113.8</td>
<td>208.5211.1</td>
<td>95.4 110.0</td>
</tr>
<tr>
<td>SwitchCNN [232]</td>
<td>66.5 77.8</td>
<td>61.5 74.2</td>
<td>74.0 83.0</td>
<td>56.0 63.4</td>
<td>69.0 80.9</td>
<td>92.8 105.8</td>
<td>67.7 79.8</td>
<td>65.7 76.7</td>
</tr>
<tr>
<td>ACSCP [237]</td>
<td>48.1 60.2</td>
<td>57.0 70.6</td>
<td>34.8 39.7</td>
<td>42.5 46.4</td>
<td>37.3 44.3</td>
<td>86.6 106.6</td>
<td>36.0 41.9</td>
<td>55.1 68.5</td>
</tr>
<tr>
<td>AMDCN [60]</td>
<td>165.6167.7</td>
<td>166.7168.9</td>
<td>163.8165.9</td>
<td>160.5162.3</td>
<td>174.8177.1</td>
<td>162.3164.3</td>
<td>165.5167.7</td>
<td>165.6167.8</td>
</tr>
<tr>
<td>CSRNet [166]</td>
<td>19.8 25.6</td>
<td>17.8 25.4</td>
<td>22.9 25.8</td>
<td>12.8 16.6</td>
<td>19.1 22.5</td>
<td>42.3 45.8</td>
<td>20.2 24.0</td>
<td>19.6 26.5</td>
</tr>
<tr>
<td>StackPooling [111]</td>
<td>68.8 77.2</td>
<td>68.7 77.1</td>
<td>68.8 77.3</td>
<td>66.5 75.9</td>
<td>74.0 83.4</td>
<td>65.2 67.4</td>
<td>95.7 101.1</td>
<td>53.1 59.1</td>
</tr>
<tr>
<td>DA-Net [336]</td>
<td>36.5 47.3</td>
<td>41.5 54.7</td>
<td>28.9 33.1</td>
<td>45.4 58.6</td>
<td>26.5 31.3</td>
<td>29.5 34.0</td>
<td>56.5 68.3</td>
<td>24.9 28.7</td>
</tr>
<tr>
<td>STGNet [280]</td>
<td>16.7 19.8</td>
<td>17.0 20.0</td>
<td>16.4 19.5</td>
<td>14.3 17.2</td>
<td>19.0 21.1</td>
<td><strong>19.6 24.1</strong></td>
<td>19.4 22.1</td>
<td><strong>15.2 18.4</strong></td>
</tr>
<tr>
<td>STGNet (den)</td>
<td>17.8 24.3</td>
<td>20.4 27.4</td>
<td><strong>13.9 18.6</strong></td>
<td>13.0 17.7</td>
<td>18.6 22.3</td>
<td>30.7 40.2</td>
<td>20.1 25.0</td>
<td>16.5 23.9</td>
</tr>
<tr>
<td>STGNet (den+loc)</td>
<td>16.6 21.3</td>
<td><strong>15.3 20.3</strong></td>
<td>18.5 22.7</td>
<td><strong>12.1 15.6</strong></td>
<td>15.0 18.4</td>
<td>33.0 36.2</td>
<td><strong>16.8 20.5</strong></td>
<td>16.4 21.8</td>
</tr>
<tr>
<td>STGNet (den+loc+ass)</td>
<td><strong>16.5 21.2</strong></td>
<td>15.5 20.7</td>
<td>18.0 21.9</td>
<td>12.5 15.9</td>
<td><strong>14.0 17.0</strong></td>
<td>33.5 36.9</td>
<td>17.5 20.7</td>
<td>15.9 21.5</td>
</tr>
</tbody>
</table>
Table 5.3: Crowd localization accuracy on the DroneCrowd dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>L-mAP</th>
<th>L-AP@10</th>
<th>L-AP@15</th>
<th>L-AP@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSRNet-L</td>
<td>14.40%</td>
<td>15.13%</td>
<td>19.77%</td>
<td>21.16%</td>
</tr>
<tr>
<td>STANet-L</td>
<td>28.43%</td>
<td>30.53%</td>
<td>36.33%</td>
<td>39.12%</td>
</tr>
<tr>
<td>STGNet-L (den)</td>
<td>11.06%</td>
<td>11.22%</td>
<td>14.66%</td>
<td>16.16%</td>
</tr>
<tr>
<td>STGNet-L (den+loc)</td>
<td>36.89%</td>
<td>41.07%</td>
<td>46.21%</td>
<td>48.91%</td>
</tr>
<tr>
<td>STGNet-L (den+loc+ass)</td>
<td>36.98%</td>
<td>41.15%</td>
<td>46.27%</td>
<td>48.98%</td>
</tr>
</tbody>
</table>

Table 5.4: Crowd tracking accuracy on the DroneCrowd dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>T-mAP</th>
<th>T-AP@0.05</th>
<th>T-AP@0.10</th>
<th>T-AP@0.20</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSRNet-T</td>
<td>9.68%</td>
<td>17.55%</td>
<td>7.80%</td>
<td>3.71%</td>
</tr>
<tr>
<td>STANet-T</td>
<td>23.76%</td>
<td>30.96%</td>
<td>23.05%</td>
<td>17.28%</td>
</tr>
<tr>
<td>STGNet-T (den)</td>
<td>5.84%</td>
<td>11.09%</td>
<td>4.30%</td>
<td>2.13%</td>
</tr>
<tr>
<td>STGNet-T (den+loc)</td>
<td>26.30%</td>
<td>32.82%</td>
<td>25.82%</td>
<td>20.26%</td>
</tr>
<tr>
<td>STGNet-T (den+loc+ass)</td>
<td>26.47%</td>
<td>33.40%</td>
<td>25.83%</td>
<td>20.19%</td>
</tr>
</tbody>
</table>

Crowd Tracking. We generate the long trajectories of objects using the min-cost flow method [212] to associate the localization results in each individual frame. As presented in Table 5.4, our STGNet achieves the top T-mAP score of 26.47%, which is over 2% gain compared with STANet [280]. It indicates that our proposed association subnet can capture temporal coherence of the targets in the video. However, the low performance is far from satisfactory in real applications.

5.4.5 Ablation study

To study the influence of important components in our STGNet, we construct three STGNet variants including “den”, “den+loc” and “den+loc+ass”. Specifically, STGNet (den) indicates the STGNet method without both localization and association subnets. STGNet (den+loc) indicates the STGNet method without the association subnet. STGNet (den+loc+ass) has all the subnets. We evaluate them on the DroneCrowd dataset for three sub-tasks, which are analyzed as follows.

First, as presented in Table 5.2, the accuracy and robustness of crowd counting are improved considerably by adding localization and association subnets in our STGNet. Moreover, the MAE and MSE scores are decreased in many subnets such as Large, Sunny, and Crowded. Notably, STGNet (den) performs better than the other two variants in the Small subset. This is maybe due to the difficulty to represent small objects. The similar cases can be also found in the Night subset.

Second, as presented in Table 5.3, it can be seen that STGNet-L (den) performs significantly
worse than the other variants. We speculate that density map based localization results is not accurate by combining Gaussian distribution of each target point. Besides, STGNet-L (den+loc+ass) performs slightly better than STGNet-L (den+loc). This it attributed to the association subnet to capture temporal coherence.

Third, as presented in Table 5.4, we observe the similar trend as that in crowd localization. That is, STGNet-T (den+loc+ass) achieves better results than STGNet-L (den+loc) (26.47% vs. 26.30%). This is because the association subnet facilitates exploiting temporal information to generate better trajectories. Since the localization results of STGNet-L (den) are inaccurate, the corresponding tracking results of STGNet-T (den) only achieve 5.85% T-mAP score.
5.5 Conclusion

In this work, we propose the space-time graph convolutional network to solve crowd counting, localization, and tracking subtasks. Notably, the proposed association subnet facilitates generating more accurate density and localization maps, resulting in more accurate results. Extensive experiments conducted on four challenging datasets demonstrate that the proposed method performs favorably against the state-of-the-art methods.
CHAPTER 6
Conclusion

6.1 Contributions

In this thesis, I investigate to use the powerful graph model to solve object tracking and counting tasks, which is effective to exploit the context information within and among objects to improve the robustness in various scenarios.

First, to solve the deformable object tracking with the occlusion and clutter background challenges, I propose a hybrid structure hypergraph based tracker using non-uniform hypergraph to describe the dependencies among target parts in consecutive frames. That is, each node in the hypergraph represents a candidate part, and each edge/hyperedge encodes the dependencies of the parts. An approximate dense structure exploiting algorithm is used to determine the parts belonging to the target. In this way, the center location and scale of target is voted by the extracted parts of targets.

Second, I design a non-uniform hypergraph learning method for multi-object tracking, which use the edges/hyperedges with various degrees to describe the dependencies among different objects. The nodes in the hypergraph correspond to tracklets, and the edges/hyperedges with different degrees encode similarities among tracklets to assemble various kinds of appearance and motion patterns. Notably, the hyperedges with different degrees are mixed and the weights of them are learned automatically from the data using SSVM. After that, the tracking task is completed by exploiting the dense structures of the hypergraph.

Third, I go a further step by designing a space-time graph convolutional network (STGNet) to solve the crowd counting, localization and tracking tasks jointly. In contrast to the general scenes, the crowded scenes contains different challenges, such as view point and scale variations, background clutter, and small scales. To that end, the proposed STGNet method combines multi-scale feature maps in sequential frames and outputs the enhanced features using deformable convolution operations for better performance. Specifically, STGNet is formed by four modules, i.e., the Siamese feature extraction subnetwork, the density map estimation branch, the localization
branch, and the association branch. Note that the graph convolution operations are used to exploit
the relations of neighboring targets instead of the conventional convolution operations. Finally, the
min-cost flow method are adopted to generate long trajectories of targets.

6.2 Future Work

To improve the performance of visual tracking, we can learn a holistic target representation
updated jointly with the local superpixel representation. Moreover, I also would like to develop a
more efficient algorithm to search dense structures over a large number of superpixels within the
non-uniform hypergraph to satisfy the real-time requirements in real applications.

To deal with multi-object tracking, different optimization strategies are worth to investigate
to solve the dense structure problem on non-uniform hypergraphs accurately and efficiently. More
investigation into how to model the appearance and motion variations of targets and integrate them
into the hypergraph framework would also be valuable to improve the robustness of the proposed
method in various challenging scenarios.

Based on the proposed STGNet method, we plan to use more light-weight backbone net-
work to ensure the running efficiency, especially on the embedded system. Meanwhile, scene
understanding is another direction worth to explore to improve the performance. For example,
the algorithm is able to suppress false counting, localization and trajectories based on the general
knowledge of the scene understanding outputs, i.e., the people are walking only on the street rather
on the building.

In summary, this thesis has done some useful attempts to solve the object tracking and count-
ing tasks using the effective graph model. I hope my attempts could play a small part in promoting
the developments of the research of the object tracking and counting fields.
### APPENDIX A

**Table of Symbols**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma$</td>
<td>Frame buffer.</td>
</tr>
<tr>
<td>$M$</td>
<td>The size of frame buffer.</td>
</tr>
<tr>
<td>$m$</td>
<td>The latest frame index in the frame buffer.</td>
</tr>
<tr>
<td>$G$</td>
<td>Non-uniform hypergraph.</td>
</tr>
<tr>
<td>$V$</td>
<td>The node set of the hypergraph.</td>
</tr>
<tr>
<td>$E$</td>
<td>The edge set of the hypergraph.</td>
</tr>
<tr>
<td>$A$</td>
<td>The weight set corresponding to the edges.</td>
</tr>
<tr>
<td>$d$</td>
<td>The degree set of the hypergraph.</td>
</tr>
<tr>
<td>$A(v_i)$</td>
<td>Self-loop affinity.</td>
</tr>
<tr>
<td>$A(v_i, v_j)$</td>
<td>Pair-wise edge affinity.</td>
</tr>
<tr>
<td>$A(v_i, v_j)$</td>
<td>High-order hyperedge affinity.</td>
</tr>
<tr>
<td>$U(v_i)$</td>
<td>The features corresponding to the node $v_i$.</td>
</tr>
<tr>
<td>$\chi(\cdot, \cdot)$</td>
<td>the chi-squared distance between two features.</td>
</tr>
<tr>
<td>$\lambda_1$</td>
<td>The balancing factor of the pairwise dependency.</td>
</tr>
<tr>
<td>$C(v)$</td>
<td>The center location of node $v$.</td>
</tr>
<tr>
<td>$m_v$</td>
<td>The frame index of node $v$.</td>
</tr>
<tr>
<td>$Q(m_v)$</td>
<td>The predicted location of node $v$.</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>The balancing factor of high-order dependency.</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>The number of nodes in the dense structure.</td>
</tr>
<tr>
<td>$y = (y_1, \cdots, y_n)$</td>
<td>the indicator variable corresponding to node $v_p$.</td>
</tr>
<tr>
<td>$D$</td>
<td>The maximal degree of the edges in $G$.</td>
</tr>
<tr>
<td>$\lambda_d$</td>
<td>The preset influence factors of edges with different degree.</td>
</tr>
<tr>
<td>$N(v_s)$</td>
<td>The neighborhood of node $v_s$.</td>
</tr>
<tr>
<td>$\hat{\alpha}$</td>
<td>The minimum nodes in the dense structure.</td>
</tr>
<tr>
<td>$y^*$</td>
<td>The local maximizer for the dense structure extraction.</td>
</tr>
<tr>
<td>${t^i_{1, \cdots, m_i}}$</td>
<td>The frame index set in a video clip.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>{t^i_1, \ldots, t^i_m}</td>
<td>The frame index set in a video clip.</td>
</tr>
<tr>
<td>T = {T_1, \ldots, T_n}</td>
<td>The tracklet set in the video sequence.</td>
</tr>
<tr>
<td>\varpi</td>
<td>the cosine similarity between the HSV histograms of the detections in the last frame and the first frame.</td>
</tr>
<tr>
<td>\mu</td>
<td>The CNN features of the detections in a frame.</td>
</tr>
<tr>
<td>\gamma</td>
<td>The areas of the detections in a frame.</td>
</tr>
<tr>
<td>\eta</td>
<td>The updating step.</td>
</tr>
<tr>
<td>N</td>
<td>The batch size.</td>
</tr>
<tr>
<td>\hat{\Phi}_n</td>
<td>The estimated density maps of the (n)-th image.</td>
</tr>
<tr>
<td>\Phi_n</td>
<td>The ground-truth density maps of the (n)-th image.</td>
</tr>
<tr>
<td>\hat{\Upsilon}_n</td>
<td>The estimated localization maps of the (n)-th image.</td>
</tr>
<tr>
<td>\Upsilon_n</td>
<td>The ground-truth localization maps of the (n)-th image.</td>
</tr>
<tr>
<td>\hat{P}_n</td>
<td>The set of selected points based on (\hat{\Upsilon}_n).</td>
</tr>
<tr>
<td>\hat{O}_{n,n+1}</td>
<td>The predicted motion offsets of targets in the (n)-th and ((n+1))-th samples in the mini-batch.</td>
</tr>
<tr>
<td>O_{n,n+1}</td>
<td>The ground-truth motion offsets of targets in the (n)-th and ((n+1))-th samples in the mini-batch.</td>
</tr>
</tbody>
</table>
# APPENDIX B

## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>Area under curve</td>
</tr>
<tr>
<td>CF</td>
<td>Correlation filter</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional neural network</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep neural networks</td>
</tr>
<tr>
<td>GNN</td>
<td>Graph Neural Networks</td>
</tr>
<tr>
<td>HOG</td>
<td>Histogram of oriented gradients</td>
</tr>
<tr>
<td>LBP</td>
<td>Local binary pattern</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Square Error</td>
</tr>
<tr>
<td>MOT</td>
<td>Multi-Object Tracking</td>
</tr>
<tr>
<td>OPE</td>
<td>One-pass evaluation</td>
</tr>
<tr>
<td>RoI</td>
<td>Region of Interest</td>
</tr>
<tr>
<td>SSVM</td>
<td>Structural support vector machine</td>
</tr>
<tr>
<td>SOT</td>
<td>Single Object Tracking</td>
</tr>
</tbody>
</table>
BIBLIOGRAPHY


[34] Jiale Cao, Yanwei Pang, and Xuelong Li, Triply supervised decoder networks for joint detection and segmentation, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Computer Vision Foundation / IEEE, 2019, pp. 7392–7401.


[70] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian, Centernet: Keypoint triplets for object detection, IEEE/CVF International Conference on Computer Vision, IEEE, 2019, pp. 6568–6577.


[77] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman, Detect to track and track to detect, IEEE International Conference on Computer Vision, 2017, pp. 3057–3065.


[118] Xiaolong Jiang, Peizhao Li, Yanjing Li, and Xiantong Zhen, *Graph neural based end-to-end data association framework for online multiple-object tracking*, CoRR abs/1907.05315 (2019).


[162] Shengkun Li, Dawei Du, Longyin Wen, Ming-Ching Chang, and Siwei Lyu, *Hybrid structure hypergraph for online deformable object tracking*, 2017 IEEE International Confer-


[303] Fan Yang, Wongun Choi, and Yuanqing Lin, *Exploit all the layers: Fast and accurate CNN object detector with scale dependent pooling and cascaded rejection classifiers*, IEEE Com-


