International Journal of Computer Vision (2021) 129:1993–2010
https://doi.org/10.1007/s11263-021-01460-0

# Deep Human-Interaction and Association by Graph-Based Learning for Multiple Object Tracking in the Wild

Cong Ma[1] · Fan Yang[1] · Yuan Li[1] · Huizhu Jia[1] · Xiaodong Xie[1] · Wen Gao[1]

Received: 20 December 2019 / Accepted: 20 March 2021 / Published online: 19 April 2021


**Abstract**
Multiple Object Tracking (MOT) in the wild has a wide range of applications in surveillance retrieval and autonomous driving. Tracking-by-Detection has become a mainstream solution in MOT, which is composed of feature extraction and data association. Most of the existing methods focus on extracting targets' individual features and optimizing the association by hand-crafted algorithms. In this paper, we specially consider the interrelation cue between targets and we propose Human-Interaction Model (HIM) to extract interaction features between the tracked target and its surrounding. The interaction model has more discriminative features to distinguish objects, especially in crowded (dense) scene. Meanwhile we propose an efficient end-to-end model, Deep Association Network (DAN), to optimize the association with graph-based learning mechanism. Both HIM and DAN are constructed by three kinds of deep networks, which include Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Graph Neural Network (GNN). The CNNs extract appearance features from bounding box images, the RNNs encoder motion features from historical positions of trajectory. And then the GNNs aim to extract interaction features and optimize graph structure to associate the objects in different frames. In addition, we present a novel end-to-end training strategy for Deep Association Network and Human-Interaction Model. Our experimental results demonstrate performance of our method reaches the state-of-the-art on MOT15, MOT16 and DukeMTMCT datasets.

**Keywords** Multiple Object Tracking in the Wild · Human Interaction Model · Deep Association Network · Graph Neural Network

## 1 Introduction

Multiple Object Tracking (MOT) is one of the most significant components in computer vision technology, which has been widely applied to video surveillance retrieval, scene understanding and autonomous driving. MOT in the wild aims to track the targets in crowded scene. A tracker identifies targets according to their features and associates the same target within different frames as contiguous trajectory. The trajectory is commonly utilized for human behavior analysis, action recognition and human feature supplement. However, MOT is still a challenging task due to unfavorable factors such as occlusion, scene complexity and indistinguishable objects. Existing methods merely consider individual features (e.g. appearance and motion) rather than interrelation cue between objects. Some complicated scenes (as shown in Fig. 1) are therefore difficult to be processed by existing methods. In addition, these methods are composed of multiple independent algorithms, which are not combined together as an integral deep network architecture. In this paper, we specially focus on inter-relation cue between objects to extract interaction features between tracked-target and its surrounding. The interaction model has more discriminative features to distinguish objects, especially in crowded scene. Mean-

Communicated by Mei Chen.

✉ Yuan Li
yuanli@pku.edu.cn

Cong Ma
Cong-Reeshard.Ma@pku.edu.cn

Fan Yang
fyang.eecs@pku.edu.cn

Huizhu Jia
hzjia@pku.edu.cn

Xiaodong Xie
donxie@pku.edu.cn

Wen Gao
wgao@pku.edu.cn

[1] National Engineering Laboratory for Video Technology, Peking University, Beijing, China

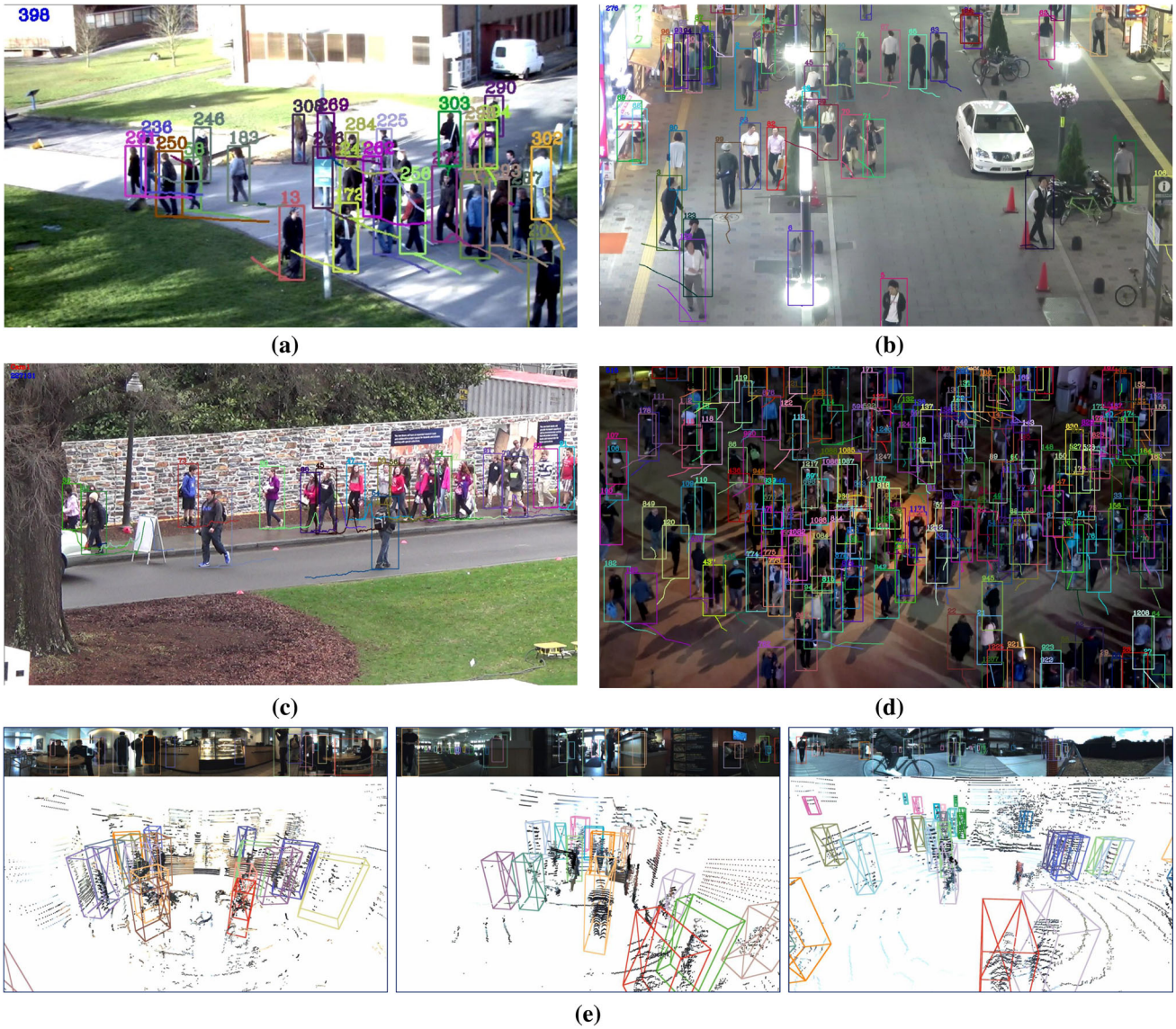

**Fig. 1** The most popular datasets on MOT in the wild currently, **a**:MOT15 (Leal-Taix et al. 2015) **b**:MOT16/MOT17 (Milan et al. 2016) **c**:Duke-MTMCT (Ristani et al. 2016) **d**:MOT20 (Dendorfer et al. 2019) **e**:JRDB (Martín-Martín et al. 2019)

while, we additionally design a novel end-to-end training method to optimize the graph structure instead of hand-crafted method.

Tracking-by-detection (TBD) has been widely used for the MOT task in recent years, which is usually based on the bounding boxes detected by leveraging off-the-shelf object detectors such as Cascade RCNN (Cai and Vasconcelos 2018), Faster RCNN (Ren et al. 2015) and Yolov3 (Redmon and Farhadi 2018). TBD associates the bounding boxes according to their temporal-spatial information in different frames. Specifically, under the assumption that the same individual from different frames has similar characteristics. TBD calculates similarity between bounding boxes according to the extracted features (e.g. appearance, motion, shape). And the tracker associate the bounding boxes which have high-similarity. The bounding boxes are linked together to form a trajectory. Therefore, traditional methods generally are divided into two modules: feature extraction and bounding box association.

Feature extraction aims to comprehensively describe an object using discriminative features, which include colors, textures, positions, boundaries, velocity, structural feature, etc. Recently, deep neural networks such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) have been widely used for the MOT task. However, most existing methods learn features from single individual but they neglect the relationship between the tracked-target and its surroundings information, which causes ”lost-

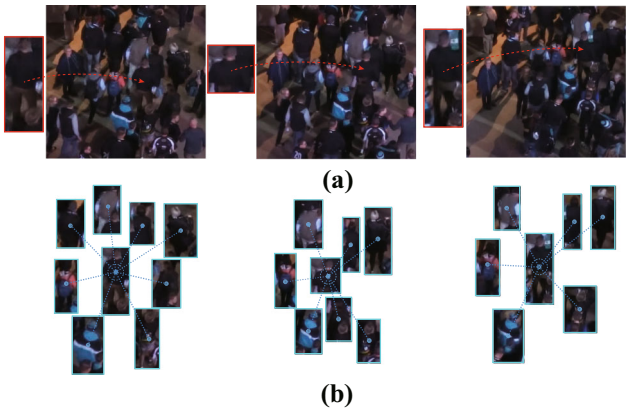


**Fig. 2** Tracking person in crowd by deep interaction feature. **a**: Target and video sequence **b**: Corresponding interaction feature for each frame

tracking" or "mis-tracking" for target in the crowd scenario. Figure 2a shows an example of complex scene at the 146-th, 222-rd and 301-st frame of sequence, respectively. The red bounding box on the left of each picture indicates a tracked person, who are occluded frequently by the other person and even sometime are occluded completely. Traditional methods are not able to track the target effectively. Fortunately, the persons around the target relatively reserve some information such as their appearance and the relative-position with target (as shown in Fig. 2b). Therefore, human-interaction detail becomes a powerful and effective feature for tracking, especially in complex scene.

Bounding boxes association is usually formulated as a Graph Optimization problem, which is about associating and removing the edge between nodes on graph structure constructed by bounding boxes and their similarity. Each bounding box is regarded as a node on the graph, meanwhile the similarity of nodes represents edge weight between nodes. The optimized and pruned graph reserves several sub-graphs, and the nodes on each sub-graph are considered as the same individual, which are labeled the same ID subsequently.

Currently, the graph optimization algorithm on MOT hasn't relied on deep learning strategy, and existing approaches continue to utilize the traditional solution such as Hungarian Algorithm (Sahbani and Adiprawita 2017) and Network Flow Algorithm (Schulter et al. 2017). MOT is a high-level semantic task, compared with low-level semantic or image processing task such as image deblurring, defogging, deraining and even semantic segmentation, MOT is more difficult to construct an end-to-end network by CNN. So far feature extraction and graph optimization are still treated as two independent tasks, they still haven't been combined as an end-to-end model for training together.

In this paper, we propose an end-to-end Deep Association Network (DAN) (as illustrated in Fig. 3), which can jointly learn and process the feature extraction and data association together. The DAN is divided into two parts: 1. Feature Extraction; 2. Graph Optimization. In the feature extraction part, besides the two individual feature extractors (appearance and motion), we additionally propose a novel inter-relation feature extractor, Human-Interaction Model (HIM), whose interaction feature is extracted by Graph Neural Network (GNN). In the Graph Optimization part, we also utilize GNN to replace hand-crafted algorithm to optimize the graph. GNN has the ability to learn and process the topological data from a large amount of data. The advantage of GNN is that it can input arbitrary graph structures. We utilize specific loss function and large-scale tracking training data to train GNN, thus the interactive information can be extracted by GNN, ultimately nodes which belong to the same individual tend to be together. We connect the two parts sequentially for end-to-end training. Our contributions of the framework is as follows:

- End-to-end MOT model framework: We firstly combine the feature extraction and graph optimization from two independent tasks to form an unified task as an end-to-end model. The framework is named as Deep Association Network (DAN).
- More discriminative deep interaction feature: we propose a Human-Interaction Model (HIM) to extract the interrelation details of tracked-target and its surrounding, which is more effective for the targets with frequent occlusion in the crowded scene.
- Special Training Strategy: We specially create graph-structured dataset for training DAN. In order to make it converge better, we set different learning rates for training different layers and train the multiple models stage-by-stage.
- Developable MOT baseline: Deep Association Network is an unprecedented model structure for multiple object tracking, therefore DAN is worth continuing to be explored and be researched on how to improve the performance of MOT.
- Our proposal greatly enhances the effectiveness of tracking in the high-density crowds and complex scenes. Our experiments demonstrate that our method achieves superior effectiveness and robustness over state-of-the-arts.

## 2 Related Work

Multiple Object Tracking has attracted researchers' attention recently. The performance of MOT improves gradually at the MOT benchmark (Milan et al. 2016). Among the methods of MOT, (Henschel et al. 2018; Tang et al. 2017; Xiang et al. 2015; Choi 2015; Kim et al. 2015; Chen et al. 2017) designed an ingenious data association or multiple hypothesis. Schul-

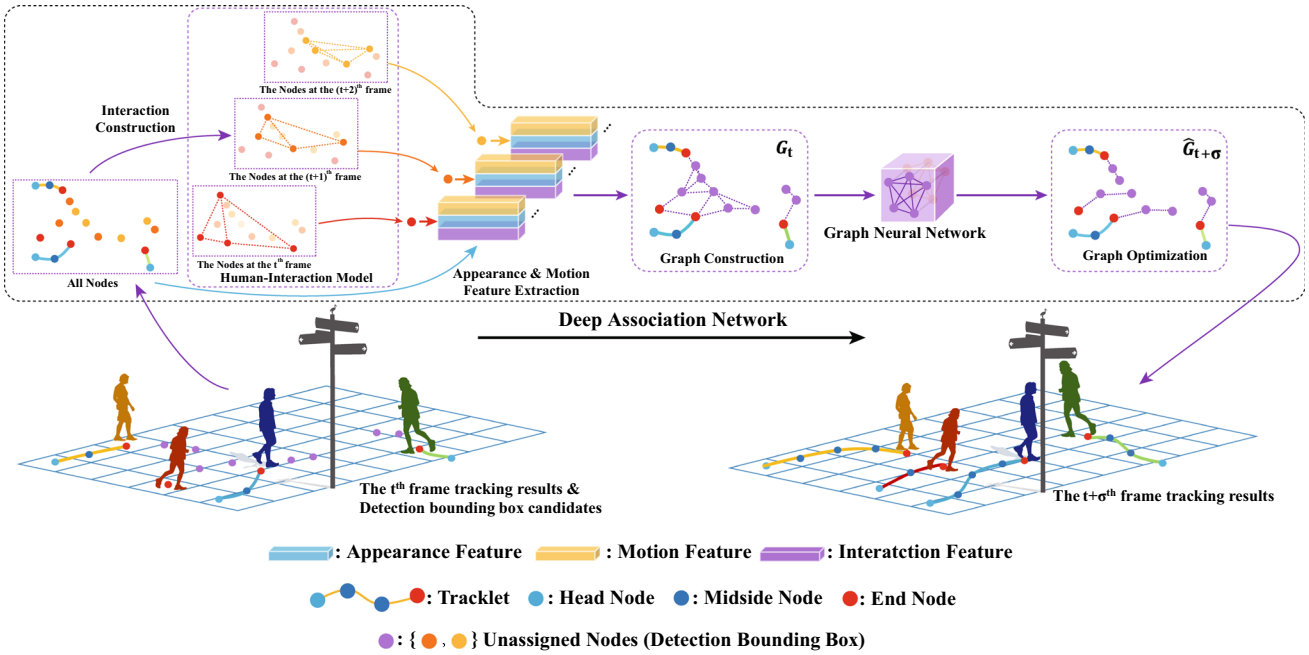


**Fig. 3** Our method pipeline: 1.Extracting feature (appearance, motion and interaction) for each node from detection results; 2.Constructing the graph structure for all of nodes; 3.Optimizing the graph structure and obtaining the association result

ter et al. (2017) is the first to combine feature extraction and hand-crafted graph structure to learn together. Bergmann et al. (2019) accomplishes tracking without specifically targeting any of these tasks. Keuper et al. (2018) combines point trajectories and clustering of bounding boxes to track objects. Chen et al. (2019) addresses the association method at the tracklet-level. Levinkov et al. (2017); Maksai et al. (2017); Ma et al. (2019) presented network flow and graph optimization which are powerful approaches. (Shen et al. 2018) aims at gluing feature learning and data association into a unity by a bi-level optimization formulation.

The appearance model aims to extract human features from an image. Tang et al. (2017); Sadeghian et al. (2017); Yang et al. (2019) train the CNN on the basis of person re-identification (Yang et al. 2020, 2019, 2020) to extract the image features, and Son et al. (2017) utilizes the quadruplet loss to enhance the feature expression. Chu et al. (2017) builds the CNN model to generate visibility maps to solve the occlusion problem. And , Henschel et al. (2018) uses a novel multi-object tracking formulation to incorporate several detectors into an integrated tracking system. Kim et al. (2015) extends the multiple hypothesis by enhancing the detection model. (Ma et al. 2018) address a sophisticated model to process trajectories. Zhu et al. (2018); Gao et al. (2018) propose spatial and temporal attention mechanisms to enhance the performance of MOT. Wang et al. (2019) combines temporal and appearance information together as a unified framework.

The motion model defines the rule of object movement, which is utilized for prediction of trajectory position in the future by their historical positions. Motion model generally is divided into linear position prediction (Son et al. 2017) and non-linear position prediction (Dicle et al. 2013). Hong Yoon et al. (2016) designs the structural constraint by the location of people to optimize assignment. Following the success of RNN models for sequence prediction tasks, (Alahi et al. 2016) proposes social-LSTM to predict the position of each person in a scene.

The interaction model extracts interaction features which describe the inter-relation information between the tracked-target and its neighboring targets. The interaction features not only express the relative position information between targets, it additionally include the visual features of the targets. Nowadays typical interaction models such as social force models and crowd motion pattern models have been widely used on Pedestrian (Crowd) Simulation (Chen et al. 2018; Yang et al. 2020), Anomaly Detection (Zhang et al. 2020; Cai et al. 2020) and Trajectory Forecasting (Sadeghian et al. 2019; Kosaraju et al. 2019). In social force models, targets are considered as agents which determine their velocity, acceleration and position based on characteristics of other objects and the environment. Motion pattern model utilizes collective spatial-temporal structure and various modalities of motion to analyze the behavior of pedestrians. Sadeghian et al. (2019); Kosaraju et al. (2019) predict the behavior and position of targets in the future by interaction model, where (Sadeghian et al. 2019) presents Sophie model for

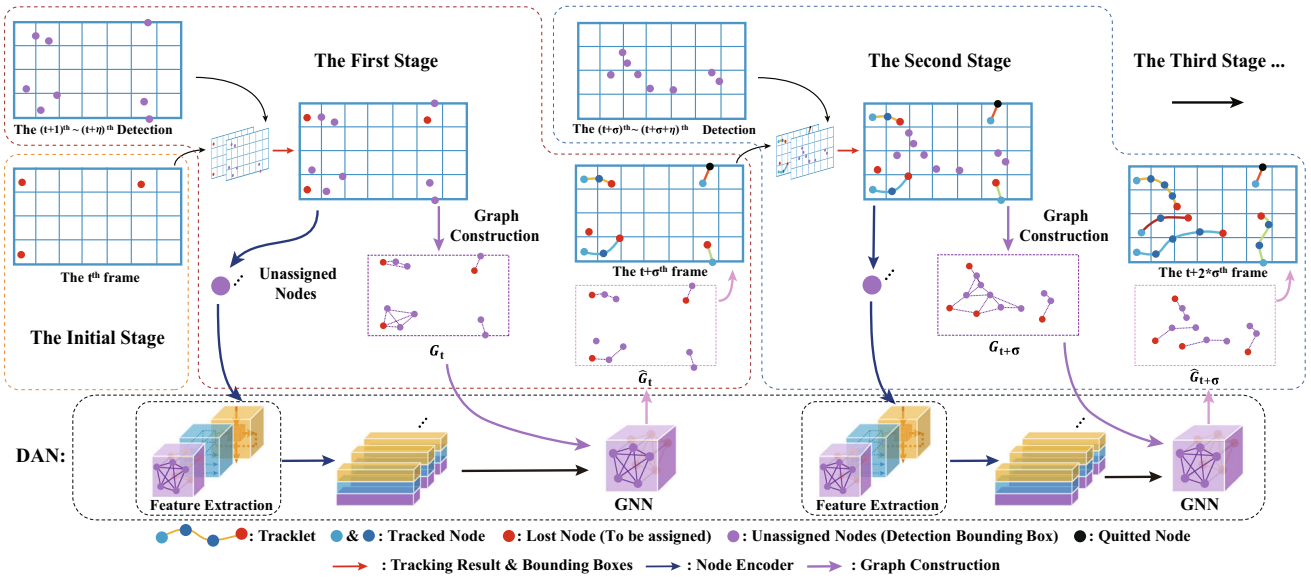


**Fig. 4** The Framework of Deep Association Network. The top of figure describes tracking progress, the bottom of figure illustrates the DAN structure, which is a sequential processing, the figure only shows initial stage and the first two stages

path prediction for multiple interacting agents in a scene, and Kosaraju et al. (2019) addresses Social-BiGAT that generates realistic multi-model trajectory prediction by better modeling the social interactions of pedestrians in a scene. However, interaction models haven't been utilized adequately yet on MOT, a few methods (Sadeghian et al. 2017; Lan et al. 2018; Wang et al. 2015) only utilized the relative positions between the tracked-target and its surrounding targets to extract a simple interaction feature. Hong Yoon et al. (2016) designs a structural constraint by the location of people to optimize assignment. However, for the targets in highly-crowded scenery, only using motion information cannot distinguish the persons who stand close together, so appearance information becomes the most discriminative features. Thus, combining motion and appearance feature together can describe targets' interaction features better.

GNN was previously applied to Natural Language Programming (NLP), physical simulation and etc. For instance, (Battaglia et al. 2018) summarizes the principle and the applications of GNN. Li et al. (2016); Kipf and Welling (2016); Veličković et al. (2018) respectively proposed the GNN variant structure, GGSNN (Gated Graph Sequence Neural Network), GCN (Graph Convolutional Network) and GAT (Graph Attention Network). Duvenaud et al. (2015) focuses on molecule feature descriptor, and each molecule is composed by atoms as a graph structure. Kipf et al. (2018) aims to research physical simulation by GNN, more specifically the interaction of dynamical particles system, meanwhile they realize basketball player trajectories' prediction. Recently, GNN has been utilized for computer vision. GNN-based few-shot transfer learning is presented by Garcia et al. (2017), and polygon refinement for instance segmentation is addressed by Acuna et al. (2018). Yan et al. (2018) adopts spatial-temporal skeleton graph for action recognition, and Shen et al. (2018) constructs relationship graph to train Re-identification model. GNN is able to extract the topological data such as molecule structure, body skeleton, etc. Interaction feature also includes the topological structure (relative position) and node attributes (appearance and motion information), therefore, GNN becomes the most suitable technique to extract interaction features according to inter-relationship and targets' information.

## 3 Deep Association Framework

In this Section, we introduce the modules of Deep Association Network one-by-one. The framework of our network and the definition of the tracking task are described in Sec.3.1. The state transitions for nodes are introduced in Sec.3.2. The regular feature extraction is explained in Sec.3.3. The details for Human-Interaction Model are introduced in Sec.3.4. We describe the strategy of the graph construction in Sec.3.5. Lastly Sec.3.6 gives the strategy of the training of GNN.

### 3.1 Deep Association Pipeline

Deep Association Network (DAN) is composed by three kinds of feature extractors (Appearance Extractor, Motion Encoder and Human-Interaction Model) to parallelly extract the targets' features, and then the features are fed into Graph Neural Network (GNN) to optimize the graph structure (as described in Fig. 4). Firstly, we obtain the detection results for each frame by the detectors. The bounding boxes for

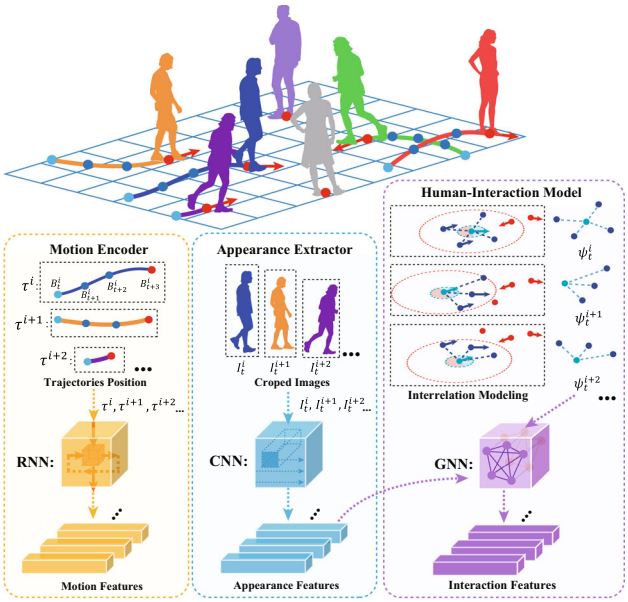


**Fig. 5** The architecture of Feature Extraction, which is consisted of Motion Encoder, Appearance Extractor and Human-Interaction Model

each image are treated as nodes on the graph. The details of each node are extracted by Feature Extraction modules, which are divided into three parts, the architecture of Feature Extraction is illustrated in Fig. 5. Appearance Extractor (AE) is applied to extract appearance features from the cropped images according to the bounding boxes, meanwhile Motion Encoder (ME) is utilized for encoding the corresponding bounding boxes' information, which contains the position, width, height and velocity of the bounding boxes as the motion features. Human-Interaction Model (HIM) generates an interaction feature, which not only includes the relative position between the tracked-target and its neighboring targets, but also fuses the appearance features of the targets together. And then, we concatenate the three types of features together as the nodes' characteristics. After feature extraction, we construct an adjacent matrix according to the spatial-temporal relationship of the bounding boxes within the frames to connect the nodes on the graph structure. The adjacent matrix and the concatenated features are fed into GNN to optimize the graph. GNN propagates node features on the graph structure and learns the relationship between the nodes. Finally, the features of the nodes close to each other have higher similarity, which can be identified as the same person, the nodes are sequentially linked to form a complete trajectory.

We formulate the near-online tracking as a local bounding box association task between the tracked candidates and the current detection results in a video fragment. We define the set of detection results in the $t$-th to $(t+\eta)$-th frames as $\mathcal{D}_t$ ($d_\xi^k \in \mathcal{D}_t, \xi \in [t, t+\eta]$), where $d_\xi^k$ is the $k$-th detection in frame $\xi$, and $\eta$ is the length of the sub-sequence. $\mathcal{C}_t$ ($c_\xi^k \in \mathcal{C}_t, \xi < t$) indicates the tracked results from initial frame to the $t$-th frame, where $c_\xi^k$ is $k$-th tracked-node in frame $\xi$. Bounding boxes association can be perceived as graph optimization. Therefore, we construct a global graph structure $\mathcal{G}$ ($G_t \in \mathcal{G}, G_t = (\mathcal{V}_t, \mathcal{E}_t)$), the global graph $\mathcal{G}$ consists of several local graphs $\{G_1, G_{1+\delta}, G_{1+2\delta}..., G_{1+n\delta}, n\delta \leq L\}$, where $G_*$ indicates local graph constructed from the video fragment from $t$-th to $(t+\eta)$-th frames, $\delta$ is the stride of video fragment on timeline. $L$ indicates total length of the video. $\mathcal{V}_t$ ($v_\xi^k \in \mathcal{V}_t, \xi \in [1, t+\eta], \mathcal{V}_t = \mathcal{C}_t \cup \mathcal{D}_t$) indicates the set of nodes in the graph, each node stands for a bounding box and $v_\xi^k$ denotes the $k$-th node in frame $\xi$, and nodes are defined as 7 dimensions $[t, id, x, y, w, h, s]$ which are the tracklet id by tracker, the object time, the center position $(x, y)$, width and height of the bounding box, and the state $s$ of the node. $e_\xi^{ij} \in \mathcal{E}_t$ is the edge between $v_{\xi_1}^i$ and $v_{\xi_2}^j$ on $G_t$. The cost function of our method is given by

$$argmin\left(\sum_{G_t \in \mathcal{G}} F_S(v_\xi^i, v_\xi^j) e_\xi^{ij} + \sum_{G_{t_1}, G_{t_2} \in \mathcal{G}} F_S(v_\varepsilon^i, v_\varepsilon^j) e_\varepsilon^{ij}\right)$$
$$s.t.\ G_{t_1} \cap G_{t_2} \neq \varnothing \tag{1}$$

The first term in Eq.(1) measures the accuracy of the data association in single graphs $G_t \in \mathcal{G}$ according to output of model, where $\xi \in [t,\ t+\eta]$, $v_\xi^i, v_\xi^j \in G_t$, and $e_\xi^{ij} \in \{0, 1\}$ indicate whether two nodes $v_\xi^i$ and $v_\xi^j$ belong to the same person, $F_S(v_\xi^i, v_\xi^j)$ measures the similarity between the nodes. The second term in Eq.(1) checks whether the association results of adjacent graphs $G_{t_1}, G_{t_2}$ in overlap region are consistent, where $\varepsilon \in [t_2,\ t_1+\eta]$, $t_2 = t_1 + \delta$ indicate the frame index in the overlap region. $v_\varepsilon^i$, $v_\varepsilon^j$ come from the same graph $G_{t_1}$ or $G_{t_2}$. Since the outputs of $G_{t_1}$ or $G_{t_2}$ are different, and $v_\varepsilon^i$, $v_\varepsilon^j$ exist in both of $G_{t_1}$ or $G_{t_2}$, so we need to calculate the cost twice for the same two-nodes. Eq.(1) belongs to a target function, we minimize the cost value of Eq.(1) by adjusting the architecture, hyper-parameter, etc to improve the effectiveness of the model indirectly.

### 3.2 Node State Transitions

In Fig. 4, there are 5 kinds of colors to represent the state $s$ of nodes (e.g. Purple: "Unassigned", Blue and Indigo:"Tracked", Red : "Lost", Black: "Quitted"). The blue and indigo dots are linked (associated) by the corresponding nodes in the next frame, we named these nodes as 'tracked', where blue dot is the first node of a tracklet. The red dots indicate the nodes at the current frame or the nodes not being associated by other nodes in their next frames, the nodes are named as 'lost'. Only one node for each tracklet can be a

'lost' node, which is in the last frame of a tracklet. All of the purple dots come from the detection bounding boxes in the current frame, they are labelled as 'unassigned' nodes.

There are four situations that the node will be switched to other state:

- "Lost"→"Tracked": If the 'lost' nodes are associated by an 'unassigned' node in next few frames, the 'lost' nodes will be turned as 'tracked' nodes.
- "Unassigned"→"Lost": If 'unassigned' nodes are associated by 'lost' nodes, they will be linked to tracklet as the last node and its state turns to a new 'lost' node.
- "Unassigned"→"Quitted": If the 'unassigned' nodes aren't associated by any nodes, they will be labelled as 'quitted'.
- "Lost"→"Quitted": We use RNN to predict the position in the future for every tracklets after each stage by their historical positions to determine whether the node leaves the scene. If the tracklet goes out of the scene, the 'lost' node of the tracklet will be labelled as the 'quitted' node. If a tracklet has not been associated for long time, in general, the difference between the last frame of the tracklet and the current frame is greater than the stride of sequence, $\delta$, the 'lost' node will be labelled as 'quitted'.

### 3.3 Regular Feature Extractor

Regular Feature Extraction is used to extract the classical characteristics of the individual to distinguish the differences between the nodes (bounding boxes). For the same individual in different frames, it has similar features for a period of time such as wearing, position, body size and velocity. These cues are totally summarized as two parts: appearance features and motion features. In our framework, the appearance features are extracted by several shared-weight CNNs, and the motion features are encoded by RNNs.

Appearance Extractor focuses on the pedestrian features (e.g. color, shape and texture) from each bounding box located by the detection model. we treat the appearance model as a person re-idenidentification (Re-ID) task initially to obtain the pre-trained model for CNNs of DAN. We combine the three public Re-ID datasets (Market1501 Zheng et al. 2015), DukeMTMC-ReID Zheng et al. 2017) and CUHK03 Zhong et al. 2017)) to train the homostructural CNNs model of DAN. $f^i_{a,\xi}$ and $f^i_{cls\_a,\xi}$ indicate the outputs of CNN's appearance feature and classification vector, respectively, for node $v^i_\xi$, the $f^i_{a,\xi}$ is the n-dimensional vector, and the $f^i_{cls\_a,\xi}$ is mapped to the K-dimensional vector by a fully-connected layer from $f^i_{a,\xi}$, n denotes the training set class number. $F_a(*)$ represents the model forward function of appearance model:

$$f^i_{cls_a,\xi}, f^i_{a,\xi} = F_A(I^i_\xi), \quad \xi \in [t, t+\eta] \tag{2}$$

where $I^i_\xi$ indicates the croped image of the node $v^i_\xi$. We use the cross-entropy loss $\mathcal{L}_{app}(*)$ in the multi-class classification task for the identification:

$$\begin{aligned} \mathcal{L}_{app}(f^i_{cls\_a,\xi}, p^i_{a,\xi}) &= \sum_{k=1}^{K} -p^i_{a,\xi}[k] \cdot \log(\hat{p}^i_{a,\xi}[k]) \\ \hat{p}^i_{a,\xi} &= softmax(f^i_{cls\_a,\xi}) \end{aligned} \tag{3}$$

where $\hat{p}^i_{a,\xi} \in \mathbb{R}^{1*K}$ is the probability by prediction, $\hat{p}^i_{a,\xi}[k]$ indicates the probability for the $k$-th class. $p^i_{a,\xi} \in \mathbb{R}^{1*K}$ indicates the ground truth label. If the $i$-th target belongs to the $k$-th class, then $p^k_{a,\xi}$=1, others elements=0.

When the identification loss tends to converge, all of the parameters will be loaded into CNNs from DAN as the pre-trained model.

Motion Encoder analyzes the pedestrian movement and predicts the position in the future. The inputs of the motion model include historical locations of tracklet and its corresponding timestamps. Motion Encoder is utilized for encoding the bounding boxes' information (position and shape). The ME (Motion Encoder) model projects the 4 dimensional vector into a m-dimensional vector by LSTM for the assigned nodes, $F^l_M(*)$, and by fully-connected layers for the unassigned nodes, $F^f_M(*)$. LSTM is able to learn sequential data. $f^i_{m,t}$ denotes the motion feature for node $v^i_t$, $F_M(*) = \{F^l_M(*), F^f_M(*)\}$ is the model forward function of motion model:

$$f^i_{m,\xi} = \begin{cases} F^l_M([B^i_{(\xi-L^i)}, ..., B^i_{(\xi-1)}, B^i_\xi]), & v^i_\xi \in Assigned \\ F^f_M(B^i_\xi), & v^i_\xi \in Unassigned \end{cases} \tag{4}$$

where $\xi \in [t, t+\eta]$ is the frame number, $L^i$ indicates the length of the $i$-th tracklet, $B^i_\xi$ represents the bounding box x, y position, width and height, $[x^i_\xi, y^i_\xi, w^i_\xi, h^i_\xi]$, of $v^i_\xi$, which are normalized to [0, 1] by the original image size. We use the position loss to train the motion model in advance, which is described as:

$$\mathcal{L}_{mot} = \sum_{\xi=1}^{L^i-1} \parallel f^i_{m,\xi} - f^i_{m,\xi+1} \parallel_2^2 \tag{5}$$

The motion features $f^i_{m,\xi}$ and $f^i_{m,\xi+1}$ are generated by LSTM model and FC model, respectively.

### 3.4 Human-Interaction Model Extractor

Human-Interaction Model (HIM) is a novel model for extracting features for MOT, which combines the information of tracked-target and its neighboring targets. Compared

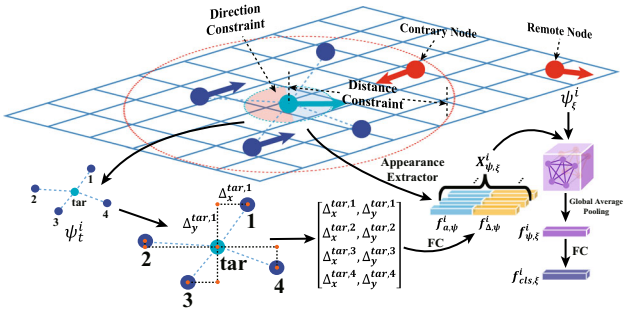


**Fig. 6** The Illustration of Building Relationships and Interaction Extraction

with the previous interaction extractor, HIM not only utilizes the relative position information between tracked node and its surrounding nodes, but also fuses appearance features of the targets together. HIM describes where the persons are located around the tracked-target and additionally describes what the persons look like. Interaction feature has more effectiveness for the target who is occluded frequently or heavily in the highly-crowded scenery.

To build relationships between the target and the other objects, we set up a connecting rule to filter the nodes which are unsuitable to extract features for tracked-target (as shown in in Fig. 6). The set of node candidates are defined as $u_\xi^j \in U$. The connecting rule includes two constraints, distance constraint and direction constraint. The candidate node $u_\xi^j$ which is over the maximum distance limit or against the direction with tracked-target $v_\xi^i$ cannot be connected in relationship. The distance constraint is described as:

$$\begin{cases} 1, & \parallel P_\xi^i - P_\xi^j \parallel_2^2 \leq \varrho_{dis} \\ 0, & \parallel P_\xi^i - P_\xi^j \parallel_2^2 > \varrho_{dis} \end{cases} \tag{6}$$

where $P_\xi^i$, $P_\xi^j$ are center positions of the bounding boxes $B_\xi^i$, $B_\xi^j$, respectively. $\varrho_{dis}$ is the maximum distance limit parameter. The direction constraint is defined as:

$$\begin{cases} 1, & cosine(v_\xi^i[\boldsymbol{v}_\xi], u_\xi^j[\boldsymbol{v}_\xi], ) \geq 0 \ \vee \ u_\xi^j[\boldsymbol{v}_\xi] = 0 \\ 0, & cosine(v_\xi^i[\boldsymbol{v}_\xi], u_\xi^j[\boldsymbol{v}_\xi], ) < 0 \end{cases} \tag{7}$$

where $v_\xi^i[\boldsymbol{v}_\xi], u_\xi^j[\boldsymbol{v}_\xi]$ are the corresponding velocities of the target and the candidate at $[t, t+\eta]$-th frame. The kept candidates are connected to the target node as the graph structure $\psi_\xi^i \in \mathbb{R}^{(p+1)*(p+1)}$, where $p$ is the number of reserved candidates. The candidates' features are regarded as node characteristics $X_{\psi,\xi}^i \in \mathbb{R}^{(p+1)*(n+m)}$, which are composed of the corresponding nodes' appearance feature $f_{a,\psi}^i$ from AE and the relative position $f_{\Delta,\psi}^i = FC([[\Delta_x^{tar,1}, \Delta_y^{tar,1}]; ...[\Delta_x^{tar,(p+1)}, \Delta_y^{tar,(p+1)}]])$, where $\Delta_x^{tar,*}$ and $\Delta_y^{tar,*}$ indicate the distance on x, y axis between the target and the corresponding neighboring node, $f_{\Delta,\psi}^i \in \mathbb{R}^{(p+1)*m}$ is the output of FC. The HIM is a Graph Neural Network structure, $F_\psi(*)$ represents the model forward function of the interaction model, which is described as:

$$f_{cls\_\psi,\xi}^i, f_{\psi,\xi}^i = F_\psi(A_{\psi,\xi}^i, X_{\psi,\xi}^i) \tag{8}$$

where $A_{\psi,\xi}^i \in \mathbb{R}^{(p+1)*(p+1)}$ is the adjacent matrix, which represents the connection between node $i$ and its surrounding nodes at frame $\xi$. $p$ is the number of nodes around node $i$. $X_{\psi,\xi}^i \in \mathbb{R}^{(p+1)*(n+m)}$ is the feature matrix of these nodes. $f_{cls\_\psi,\xi}^i$ and $f_{\psi,\xi}^i$ are the interaction feature and classification feature of node $i$, respectively. To process graph data using convolution operation, we transform the adjacent matrix to the frequency domain by Laplace Transform (Kipf and Welling 2016). In this way, an adjacent matrix is transformed to a Laplace Matrix. We adopt the Symmetric Normalized Laplacian Matrix $L$ in implementation, which is defined as:

$$L = \Gamma^{-\frac{1}{2}} A \Gamma^{-\frac{1}{2}} \tag{9}$$

where $\Gamma$ is Degree Matrix, which is the diagonal matrix where each element $\Gamma[i,i]$ is the degree of the vertex $i$ (number of edges attached to the vertex i). $A$ (Adjacent Matrix) is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph. For Laplacian Transform, convolution in time domain equals point-wise multiplication in frequency domain. The graph convolution simplifies to:

$$g_\theta \star x \approx \theta(\Psi_N + \Gamma^{-\frac{1}{2}} A \Gamma^{-\frac{1}{2}}) x \tag{10}$$

where $\star$ indicates convolution, and $g_\theta$ is a filter parameterized by $\theta$ in the Fourier domain. $x$ is the feature matrix, $\theta$ is a learnable parameter, and $\Psi_N$ is the identity matrix. Note that $\Psi_N + \Gamma^{-\frac{1}{2}} A \Gamma^{-\frac{1}{2}}$ has eigenvalues in the range $[0, 2]$. Repeated application of Eq.(10) leads to numerical instabilities and exploding/vanishing gradients when used in a deep neural network model. To alleviate this problem, we introduce the following re-normalization trick: $\Psi_N + \Gamma^{-\frac{1}{2}} A \Gamma^{-\frac{1}{2}} \rightarrow \tilde{\Gamma}^{-\frac{1}{2}} \tilde{A} \tilde{\Gamma}^{-\frac{1}{2}}$, with $\tilde{A} = A + \Psi_N$, $\tilde{\Gamma}[i,i] = \sum_j \tilde{A}[i,j]$, the Eq.(10) is approximated as follows:

$$\hat{X} = \tilde{\Gamma}^{-\frac{1}{2}} \tilde{A} \tilde{\Gamma}^{-\frac{1}{2}} X \Theta \tag{11}$$

where $X \in \mathbb{R}^{N*C}$ is feature matrix, $N$ is the number of nodes, $C$ is input channel ($C$-dimensional feature vector). $\Theta \in \mathbb{R}^{N*F}$ is a learnable parameter of GCN. $\hat{X} \in \mathbb{R}^{N*F}$ is the convolved feature matrix. The Eq.(8) is expended as:

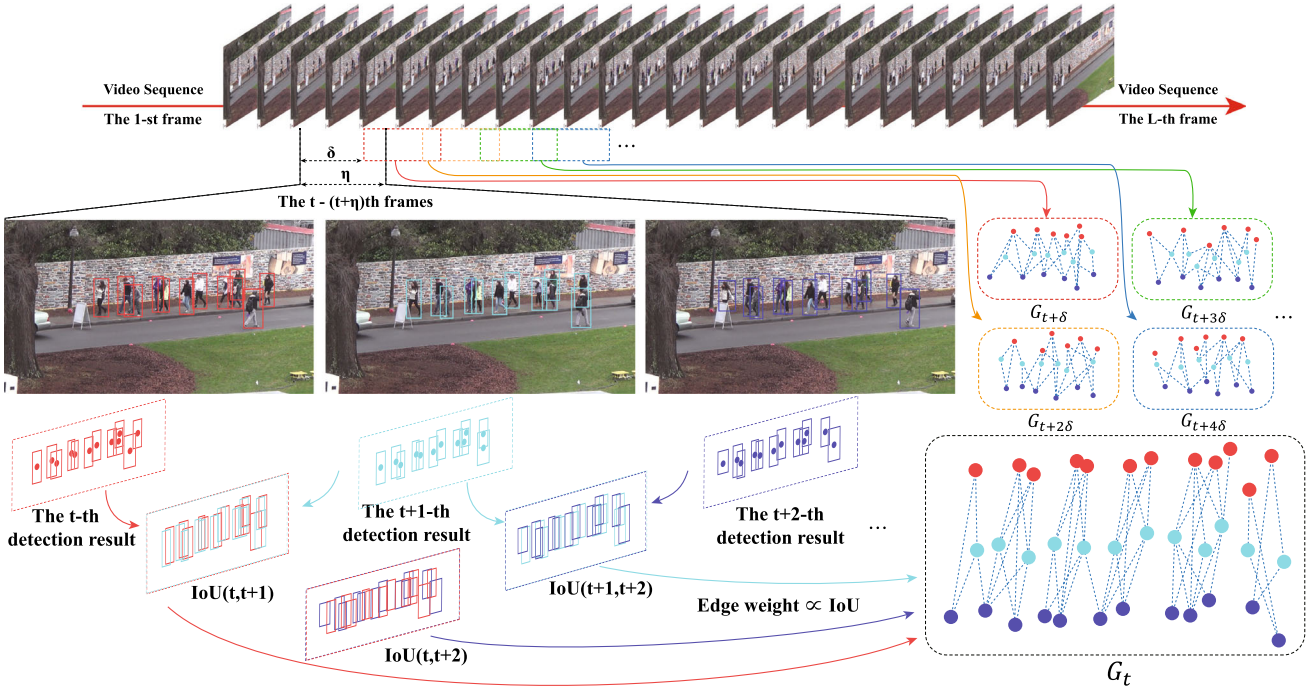


**Fig. 7** Illustration of the Graph Construction. The whole sequence is divided into several sub-sequence, and the strategy constructs the graph by bounding boxes overlap and time interval in sub-sequence

$$\hat{X}^i_{\psi,\xi} = ReLU(\tilde{\Gamma}^{-\frac{1}{2}}\tilde{A}^i_{\psi,\xi}, \tilde{\Gamma}^{-\frac{1}{2}}X^i_{\psi,\xi}\Theta) \tag{12}$$

$$\tilde{A} = A + \Psi_N,\ \tilde{\Gamma}[i,i] = \sum_j \tilde{A}[i,j] \tag{13}$$

$\Psi_N$ is the identity self-connections matrix, and the $\tilde{A}$ is the combination of adjacency matrix and self-connections. $\tilde{\Gamma}$ indicates a degree matrix of graph $G$, and $\Theta \in \mathbb{R}^{(p+1)*(n+m)}$ is a learnable parameters on GNN, and feature matrix $\hat{X}^i_{\psi,\xi} \in \mathbb{R}^{(p+1)*l}$ indicates output the GNN. And then, $\hat{X}^i_{\psi,\xi}$ are global Max Pooling on node-level:

$$f^i_{\psi,\xi} = MaxPooling(X^i_{\psi,\xi}) \tag{14}$$

$f_{\psi,\xi}$ is a $l$-dimensional vector, which represents the interaction feature. We also adopt the classification loss $\mathcal{L}_{int}$ for training HIM, which is defined as:

$$\mathcal{L}_{int}(f^i_{cls\_\psi,\xi}, p^i_{\psi,\xi}) = \sum_{k=1}^{K} -p^i_{\psi,\xi}[k] \cdot \log(\hat{p}^i_{\psi,\xi}[k])$$
$$\hat{p}^i_{\psi,\xi} = softmax(f^i_{cls\_\psi,\xi}) \tag{15}$$

where the value of $p^i_{\psi,\xi}$ represents ground truth label for the $i$-th target.

The features generated by AE, ME and HIM are concatenated together, which is defined as:

$$f^i_\xi = Concatenate[f^i_{a,\xi}, f^i_{m,\xi}, f^i_{\psi,\xi}] \tag{16}$$

where $f^i_\xi$ is the feature of node $v^i_\xi$, and then we concatenate all of the nodes within $[t, t+\eta]$ as:

$$X_t = Concatenate[f^1_t; ...; f^i_t; f^1_{t+1}; ...; f^j_{t+\eta}] \tag{17}$$

where $X_t \in \mathbb{R}^{N*(n+m+l)}$ indicates the feature matrix, $N$ is the number of all of nodes within $[t, t+\eta]$.

### 3.5 Graph Construction

Before feeding into GNN, we firstly divide the video sequence as several sub-sequences, and pre-process the bounding boxes for each sub-sequence. The bounding boxes are constructed and normalized as graph-structured data. The graph structure consists of nodes and edges represented as $G = (\mathcal{V}, \mathcal{E})$ (mentioned in Sec.3.1), where the nodes $v \in \mathcal{V}$ represent the bounding boxes, and the edges $e \in \mathcal{E}$ denote the spatial-temporal relationship between nodes. The graph construction is described in Fig. 7. We divide the whole video sequence into several video fragments (sub-sequence). $\eta$ is the length of video fragments, for instance in order to generate the graph $G_t$, we firstly extract the $t$-th frame to the $(t+\eta)$-th frame images and select nodes from sub-sequence in terms of the node state. The state of the nodes can be divide into "Tracked", "Lost", "Unassigned", and "Quitted" (described in Sec.3.2).Only 'lost' and 'unassigned' nodes will be selected for graph construction. We calculate the node's bounding boxes IoU (Intersection over Union)

between adjacent frames, the edge weight is proportional to IoU value and inversely proportional to the frame interval. The edge weight $e_{\varepsilon i,\xi j}$ of graph $G_t$ between $B_\varepsilon^i$ and $B_\xi^j$ is defined as:

$$e_{\varepsilon i,\xi j} = IoU(B_\varepsilon^i, B_\xi^j) * (1 - \mu * min(abs(\varepsilon - \xi), \lambda)) \\ s.t.\ \ \varepsilon, \xi \in [t, t+\eta], \ \ \varepsilon < \xi, \ \ \mu, \lambda > 0 \tag{18}$$

where

$$IoU(B_\varepsilon^i, B_\xi^j) = \frac{area(B_\varepsilon^i \cap B_\xi^j)}{area(B_\varepsilon^i \cup B_\xi^j)} \tag{19}$$

where constant $\mu$ is used for adjusting the influence of frame interval on edge weight, and constant $\lambda$ is the upper limit of frame interval. The representation of graph $G_t$ includes adjacent matrix $A_t \in \mathbb{R}^{N*N}$, $A_t[i,j] = e_{i,j}$ and features matrix $X_t \in \mathbb{R}^{N*(n+m+l)}$, $X_t[i] = [f_a^i, f_m^i, f_\psi^i]$, which are introduced in Eq.(16) and Eq.(17).

### 3.6 Graph Optimization by GNN

Graph Neural Network (GNN) aims to learn the topological data pattern to represent the graph structure feature, which encodes the node features and updates the representation vector in the graph. Better than CNN and RNN, GNN has more significant effects on the graph structure based task, such as molecule classification and particle interaction simulation.

The target of multiple object tracking task is to locate every pedestrian's position at each moment. So we associate the node ID and connect the nodes which belong to the same person as the tracking result. The MOT method address this problem by Data Association, which involves network flow, graph-cut and feature clustering, so GNN is able to optimize the graph node feature and edge weights between nodes. we adopt the Graph Convolutional Network (GCN) (Kipf and Welling 2016) as the network backbone. The adjacent matrix $A_t$ and feature matrix $X_t$ are denoted as the GCN input, and the GCN outputs include updated $\hat{A}_t$ and $\hat{X}_t$. $F_G(*)$ indicates the model forward function of GCN:

$$\hat{A}, \hat{X} = F_G(A, X) \tag{20}$$

where forward function $F_G$ is similar to Eq.(12) and Eq.(13)

$$\hat{X} = ReLU(\tilde{\Gamma}^{-\frac{1}{2}} \tilde{A} \tilde{\Gamma}^{-\frac{1}{2}} X \Theta) \tag{21}$$

where, feature matrix $\hat{X} \in \mathbb{R}^{N*q}$ denotes one of the GCN output, each node feature is a $q$-dimension vector. The updated adjacency matrix $\hat{A} \in \mathbb{R}^{N*N}$ is given by:

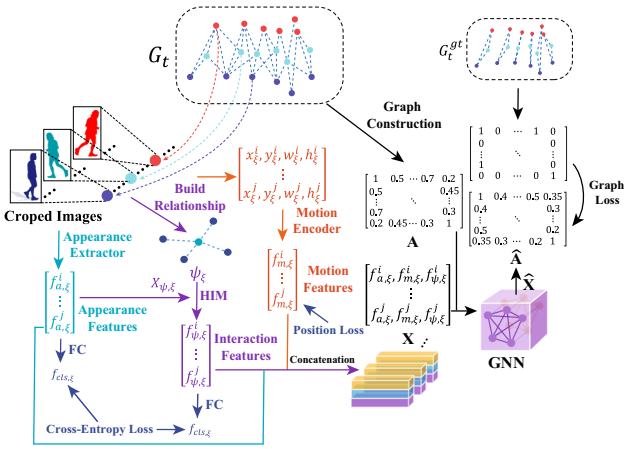


**Fig. 8** The explanation of DAN training and Loss Function

$$\hat{A} = \frac{(\hat{X}_{norm} * \hat{X}_{norm}^T) + 1}{2} \tag{22}$$

where

$$\hat{X}_{norm}[i] = \frac{\hat{X}[i]}{||\hat{X}[i]||}, \quad i \in [1, N] \tag{23}$$

where Eq.(23) indicates row-wise normalization, $\hat{X}[i]$ indicates the $i$-th row of $\hat{X}$. Every element of $\hat{A}$ is between 0 to 1. $\hat{A}[i,j]$ indicates the association probability between the $i$-th node and the $j$-th node. The multi-layers GCN feedward function is shown as:

$$\hat{A}_1, \hat{X}_1 = F_{G_1}(A, X) \tag{24}$$

$$\hat{A}_\zeta, \hat{X}_\zeta = F_{G_{\zeta-1}}(\hat{A_{\zeta-1}}, \hat{X_{\zeta-1}}), \ \zeta > 1 \tag{25}$$

Finally, to train the DAN, we design a Graph Loss $\mathcal{L}_G(*)$, which is defined as:

$$\mathcal{L}_G(\hat{A_\zeta}, G_t^{gt}) = \sum_{e_{ij}^{gt} \in \mathcal{E}_t^{gt}} (e_{ij}^{gt} - e_{ij}) \\ + \sum_{e_{ij}^{gt} \notin \mathcal{E}_t^{gt}} \sigma * (e_{ij} - e_{ij}^{gt}) \tag{26}$$

where $G_t^{gt}$ is the ground truth graph structure which is computed previously, $G_t^{gt} = (\mathcal{V}_t^{gt}, \mathcal{E}_t^{gt})$, $e_{ij}^{gt} \in \mathcal{E}_t^{gt}$, $e_{ij}^{gt} = \{0, 1\}$, and $\sigma$ is the loss weight of Graph Loss. The total loss $\mathcal{L}$ of the training DAN includes cross-entropy loss $\mathcal{L}_{app}$, $\mathcal{L}_{int}$ for AE and HIM, position loss $\mathcal{L}_{mot}$ for ME, and graph loss $\mathcal{L}_G$ for GCN (described in Fig. 8).

## 4 Experiments

### 4.1 MOT Datasets

Our method is evaluated on public benchmark (MOT Challenge (Milan et al. 2016)), which includes MOT15 (Leal-Taix et al. 2015), MOT16 (Milan et al. 2016), Duke-MTMCTT (Ristani et al. 2016) and MOT20 (Dendorfer et al. 2019). All datasets contain large-scale video sequences from different cameras and scenes. The training sets provide ground truth bounding boxes and IDs by annotator, and testing sets only give the detection results by detector.

**MOT15** includes 22 sequences which are divided into one half for training and the other half for testing. The testing data contains over 10 minutes of footage and 61440 annotated bounding boxes, the videos are from moving cameras or static cameras, respectively. MOT15 additionly provides the detection results by DPM (Felzenszwalb et al. 2010) as the tracker inputs.

**MOT16** is a classical evaluation dataset comparing several tracking methods on MOT Challenge, which includes 14 sequences captured from surveillance, hand-held shooting and driving recorder by static cameras and moving cameras. The length of each video is about 500-1500 frames. And the dataset also provides the detections by DPM (Felzenszwalb et al. 2010).

**DukeMTMCT** is a large scale dataset for multiple-camera multiple-object tracking, which are captured by 8 surveillance cameras at different viewing angles include 2800 identities (person) in Duke University . The video duration of each camera is 86 minutes, which is split into training set (0-50 min) and testing set (50-86 min). In addition, the dataset provided DPM (Felzenszwalb et al. 2010) and Openpose (Cao et al. 2018) detection results for each frame as the tracker input.

**MOT20** has been carefully selected to challenge trackers and detectors on extremely crowded scenes. In contrast to previous challenges, the total dataset contains 8 sequences. All sequences are filmed in high resolution from an elevated viewpoint, and the mean crowd density reaches 246 pedestrians per-frame which is 10 times larger than the previous benchmark. MOT20 utilizes a Faster R-CNN (Ren et al. 2015) with ResNet101 backbone on the MOT20 training sequences as the tracking input.

### 4.2 Implementation Details

In our experiments, DAN consists of feature extraction and graph optimization. For feature extraction, appearance extractor is composed of CNN, whose architecture backbone is SeResNet-50 (Hu et al. 2018). The tracked-targets' images are resized to 256×256 from the cropped images and the outputs of CNN produces appearance feature $f_{a,t}$, a 2048-dimensional vector to describe the image. Motion Encoder (ME) is composed of 3-layers of LSTM network, batch-normalization and ReLU. Bounding box information [x,y,w,h], a 4-dimensional vector is raised to $4 \rightarrow 64 \rightarrow 512$ dimensional vector finally by ME. Human-Interaction Model is a GNN structure. To build the relationship, we set the maximum distance by the sum of average width and height of all the bounding boxes in the current frame. The relatively position $[[\Delta_x^{tar,1}, \Delta_y^{tar,1}]; ...[\Delta_x^{tar,p}, \Delta_y^{tar,p}]]$ are input into 3-layers fully-connected network. The corresponding appearance feature from feature extractor and relative position are jointly fed into the GNN model. The output of GNN is a 2048-dimensional vector to express the target surrounding feature. To construct relationship rapidly, we adopt a function "radius_neighbors_graph()" from sklearn library to implement graph construction for each target. The function rapidly connects the nodes whose distance is less than radius as an adjacent matrix. We only generate two adjacent matrices for each frame, one representing the adjacent matrix $A_{dis} \in \mathbb{R}^{N*N}$ with distance constraint, the other denoting the adjacent matrix $A_{ori} \in \mathbb{R}^{N*N}$ with orientation constraint, where $A_{dis/or}[i, j] \in 0, 1$, and then Inter-connection Matrix $A_{\psi}$ is calculated by $A_{dis}$ & $A_{ori}$. The training set graph size is restricted to 64-512 nodes per graph, the length of subsequence $\eta$ is 30 frames, and the stride of sequence $\delta$ is 20. For each step of epoch, we replace data batch size with graph size to feed CNN model. For graph construction, the frame interval impact factor $\mu$ is 0.08 and the upper limit constant $\lambda$ is 10. The input of GCN is a N × 2560-dimensional vector, where N is the number of nodes on graph, and we adopt two-layers GCN and the graph loss weight $\sigma$ is 2. The training optimizer is the AdamOptimizer (Kingma et al. 2014). We load the pre-trained weight on CNN, the learning rate is less than the others models. The initial learning rate for CNN is set to 1e-5 and learning rates for the others are set to 0.001. The model converges finally at the 150th epoch.

### 4.3 Ablation Study

Table 1 shows the tracking performance with different components and network on validation datasets. The table is divided into three groups, the first group compares results between without and with Human-Interaction Model. The interaction feature is able to associate the same target before and after occlusion in the crowd, as a result, it can drastically reduce Id-switch (IDSw.) from 1286 to 874 , Fragments (Frag.) from 2181 to 1862 and improve MOTA by 4.1% and IDF1 by 4.2%.

The second group demonstrates the difference between Hungarian Algorithm and Graph Network Association. MOTA, IDF1 are increased by 1.5%, 0.7%, respectively, when hand-crafted optimization (H) is replaced by GNN

**Table 1** Tracking results on validation dataset (MOT15-PETS09-S2L1, MOT16-04, MOT20-05) with different components. (A): Appearance Extractor; (M): Motion Encoder; (H): Hungarian Algorithm; (I): Human-Interaction Model; (G): Graph Network Association

| Tracker | MOTA↑ | IDF1↑ | IDSw.↓ | Frag↓ |
|---|---|---|---|---|
| A+M+H | 53.4 | 49.0 | 1286 | 2181 |
| A+M+I+H | 57.5 | 53.2 | 874 | 1862 |
| A+M+H | 53.4 | 49.0 | 1286 | 2181 |
| A+M+$G_{256}$ | 54.9 | 49.7 | 1114 | 2013 |
| A+M+I+$G_{256}$ | **59.6** | **55.7** | **746** | **1643** |
| A+M+I+H | 57.5 | 53.2 | 874 | 1862 |
| A+M+I+$G_{64}$ | 56.6 | 52.1 | 1072 | 2058 |
| A+M+I+$G_{128}$ | 59.4 | 55.1 | 865 | 1685 |
| A+M+I+$G_{256}$ | **59.6** | **55.7** | **746** | **1643** |
| A+M+I+$G_{512}$ | 58.5 | 55.0 | 964 | 1881 |

($G_{256}$). On this basis, adding interaction model can further enhance the tracking performance.

The third group shows the results on graph structures with different number of nodes, where $G_{256}$ indicates that the graph structure includes no more than 256 nodes. We firstly replace (H) by $G_{64}$, however the result is not satisfactory, because the fewer nodes we get, the shorter sub-sequence $\eta$ is. GNN is difficult to associate the occluded target with un-occluded target in short time. Increasing the number of nodes can enlarge the length of sub-sequence, the same targets before and after occlusion can be associated. Therefore, MOTA and IDF1 are improved gradually until the number of nodes equals 256. The same target's appearances change (illumination, angle, scale, etc.) in long-term, too many nodes in graph leads the network to be unable to distinguish the feature of the same and different targets well. As a consequence, $G_{512}$ decreases the performance slightly. Finally, we adopt $G_{256}$ as the graph optimization part.

## 4.4 DAN training Details

To train the DAN, we firstly divide training sequences into many short sub-sequence, and the length of each sub-sequences is about 20-30 frames. If every frame includes 15-20 pedestrians, each sub-sequence has 64-512 nodes and we construct them as a graph. We set the maximum number of nodes at graph construction phase, $G_{256}$ indicates that the number of nodes is no more than 256, otherwise the rest of the nodes are moved to the next sub-sequences. For the whole sequence, we can get about 2k-10k graphs, and the number of graph depends on sub-sequence length and sample stride length. And we shuffle these graphs for each epoch, and our experiment implements on the Pytorch framework by 4 Nvidia Titan X GPUs, and device 0,1,2 are used for loading

**Algorithm 1:** Deep Association Network Training Process

**Input**: $\mathcal{G} = \{G_1, G_{1+\delta}, G_{1+2\delta}..., G_{1+n\delta}], n\delta < L\}$,
$G_t = \{\mathcal{V}_t : \{I_t, B_t\}, \mathcal{E}_t : \{A_t\}\}$,
$gt = \{p_{a,\xi}, p_{\psi,\xi}, G_t^{gt}\}$,
$F_A, F_M, F_\psi$ *pretrain weight*
**Output**: $F_A, F_M, F_\psi, F_G$ *model weight*

**for** $epoch = 1 : max\_epoch$ **do**
  $\mathcal{G}'$= shuffle($\mathcal{G}$);
  **for** $G_t$ *in* $\mathcal{G}'$, $gt$ **do**
    $A_t, \mathcal{V}_t = G_t$;
    $I_\xi, B_\xi = \mathcal{V}_t,\ \xi \in [t, t+\eta]$;
    $f_{cls\_a,\xi}^{gt}, f_{cls\_\psi,\xi}^{gt}, G_t^{gt} = gt$;
    $f_{cls\_a,\xi}, f_{a,\xi} = F_A(I_\xi)$;
    $cls\_a\_loss = \mathcal{L}_{app}(f_{cls\_a,\xi}, p_{a,\xi})$;
    **for** *i=1:num_track* **do**
      $f_{m,\varepsilon}^i = F_M(B_\varepsilon^i),\ \varepsilon \in [\eta, \eta + L^i - 1]$;
      $mot\_loss$+=$\mathcal{L}_{mot}(f_{m,\varepsilon}^i, f_{m,\varepsilon+1}^i)$
    $\psi_\xi = Build\_Relationship(\mathcal{V}_t)$
    $X_{\psi,\xi} = Select\_Node(f_{a,\xi}, \psi_\xi)$;
    $f_{cls\_\psi,\xi}, f_{\psi,\xi} = F_\psi(\psi_\xi, X_{\psi,\xi})$;
    $cls\_\psi\_loss = \mathcal{L}_{int}(f_{cls\_\psi,\xi}^i, p_{\psi,\xi})$;
    $f_\xi = Concatenate[f_{a,\xi}, f_{m,\xi}, f_{\psi,\xi}]$;
    $X_t = Concatenate[f_t^1; ...; f_t^i; f_{t+1}^1; ...; f_{t+\eta}^j]$;
    $\hat{A}_t, \hat{X}_t = F_G(A_t, X_t)$;
    $graph\_loss = \mathcal{L}_G(\hat{A}_t, G_t^{gt})$;
    $mot\_loss = mot\_loss + graph\_loss$;
    $F_M.backward(mot\_loss, lr_m)$; //retain_graph
    $app\_loss = cls\_a\_loss + graph\_loss$;
    $F_A.backward(app\_loss, lr_a)$; //retain_graph
    $int\_loss = cls\_\psi\_loss + graph\_loss$;
    $F_\psi.backward(int\_loss, lr_i)$; //retain_graph
    $F_G.backward(graph\_loss, lr_g)$;
**Return:** $F_A, F_M, F_\psi, F_G$.

the CNN model to extract appearance features, and then the features transfers to device 3 to calculate motion encoder and GNN model. The processing of network forward and backward are shown as **Algorithm 1**. Shuffling graph is only used in training phase. However, in inference phase, the model processes the sequence stage-by-stage according to the order of sequence as shown in Fig. 8. For each epoch, we need to pass all of training set once (We used 12 sequences for training, each sequence has 450-1200 frames). However, we set the length of sub-seqence = 30 and stride = 20, so actually we have o pass 1.5 times length of sequence for each epoch. It takes about 80-90 minutes (about 4500s). It takes about 7-8 days to finish 150 epochs.

## 4.5 Inter-relationship Building Results

Human Interaction Model is utilized for extracting interaction features between target and neighbors. The inter-relationship buildup is illustrated in Fig. 9, where Fig. 9a shows the video frames at different times. The cropped

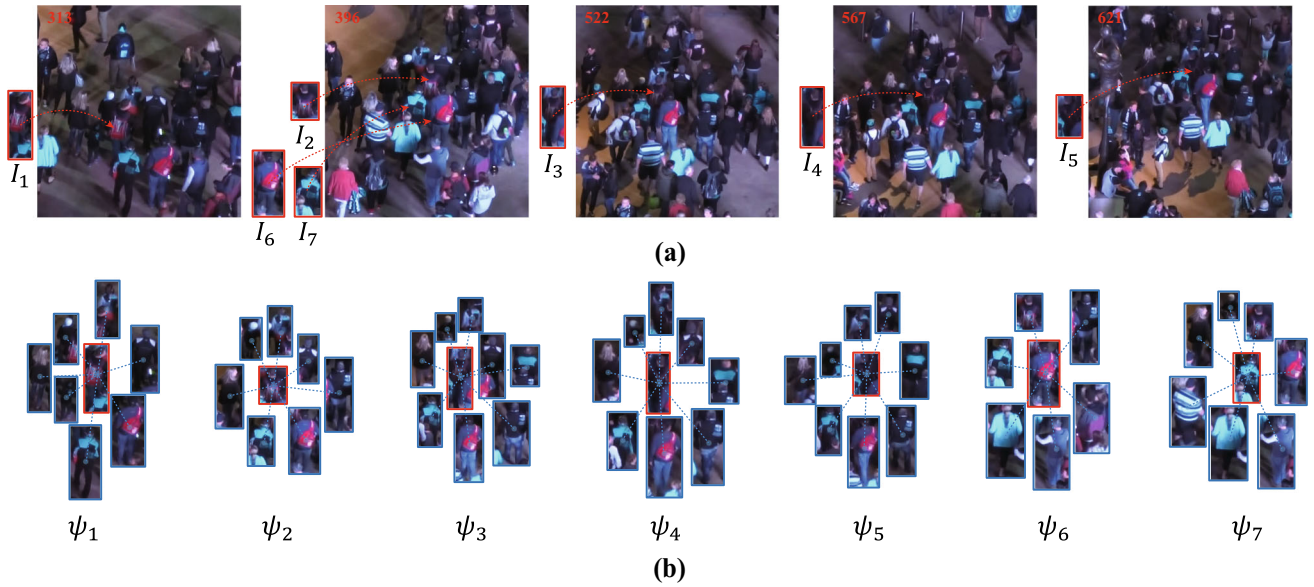


**Fig. 9** The performance of Inter-relationship Building. **a**: Video frames and tracked-targets at different times. **b**: Corresponding Inter-relationship structure of targets

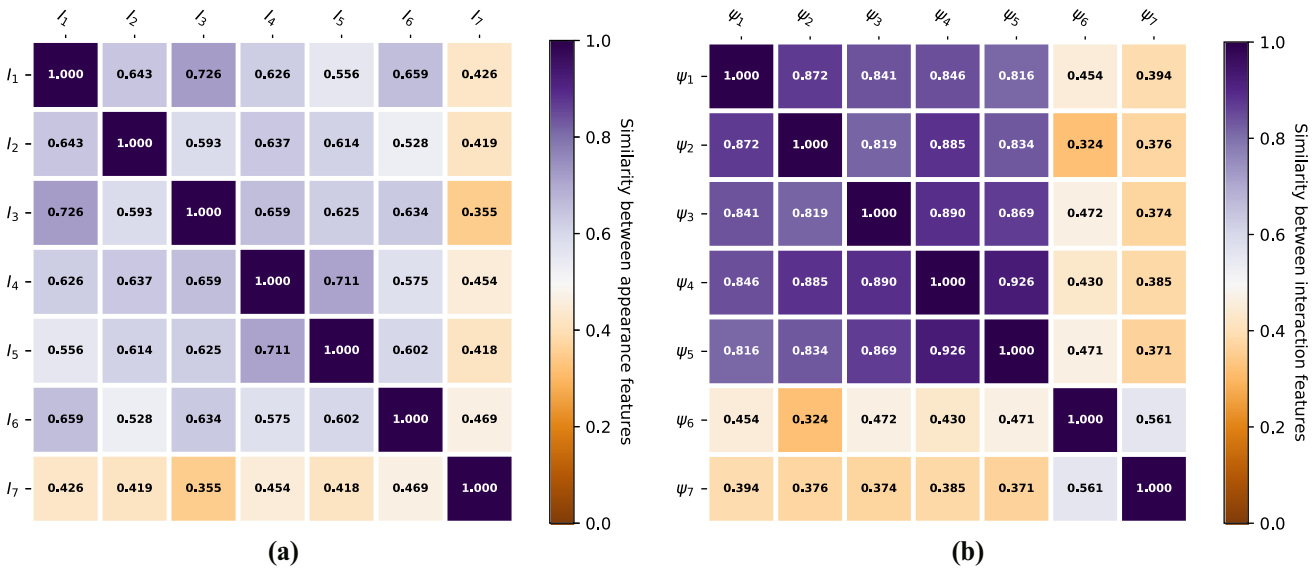


**Fig. 10** **a**: Cosine similarity between appearance features of targets $I_1 \rightarrow I_7$ from Fig. 9.**a**; **b**: Cosine similarity between interaction features of targets $\psi_1 \rightarrow \psi_7$ from Fig. 9.**b**

images $I_{1\rightarrow5}$ on the left of each frame indicates the tracked-target. Due to the frequently occlusion and illumination variation, the target's appearance and bounding box shape change quite much, furthermore, the most prominent feature (a red knapsack) of target is erased in some frames, which causes the appearance features for the same target at different frames to become indistinguishable. Fig. 9b shows the results of inter-relationship structure $\psi_{1\rightarrow5}$ for $I_{1\rightarrow5}$. Besides, the figure additionally shows the target's neighbor $I_{6\rightarrow7}$ and relationship $\psi_{6\rightarrow7}$ to contrast the difference of feature with the target, where one of the neighbor has a similar red knapsack as the target, another neighbor has the different appearance from the target.

## 4.6 Appearance and Interaction Comparison

Fig. 10 demonstrates the appearance and interaction cosine similarity between the same target at different frames and different target at the same frames. For appearance similarity estimation, we calculate cosine similarity as a 7×7 matrix $S_I$ between targets $I_{1\rightarrow7}$, where $S_I[i, j]$ indicates the similarity between $I_i$ and $I_j$ (as shown in Fig. 10a). In

**Table 2** Results on the MOT16 test dataset

| Tracker | MOTA↑ | IDF1↑ | MT↑ | ML↓ | FP↓ | FN↓ | IDSw.↓ | Frag↓ |
|---|---|---|---|---|---|---|---|---|
| AMIR (Sadeghian et al. 2017) | 47.2 | 46.3 | 14.0% | 41.6% | **2681** | 92856 | 774 | 1675 |
| NLLMPa (Wang et al. 2016) | 47.6 | 47.3 | 17.0% | 40.4% | 5844 | 89093 | 629 | 768 |
| MOTDT (Long et al. 2018) | 47.6 | 50.9 | 15.2% | 38.3% | 9253 | 85431 | 792 | 1858 |
| FWT (Henschel et al. 2017) | 47.8 | 44.3 | 19.1% | 38.2% | 8886 | 85487 | 852 | 1534 |
| GCRA (Ma et al. 2018) | 48.2 | 48.6 | 12.9% | 41.1% | 5104 | 88586 | 821 | 1117 |
| STRN (Xu et al. 2019) | 48.5 | 53.9 | 17.0% | 34.9% | 9038 | 84178 | 747 | 2919 |
| TLMHT (Sheng et al. 2018) | 48.7 | 55.3 | 15.7% | 44.5% | 6632 | 86504 | 413 | 642 |
| LMP (Tang et al. 2017) | 48.8 | 51.3 | 18.2% | 40.1% | 6654 | 86245 | 481 | **595** |
| KCF16 (Chu et al. 2019) | 48.8 | 47.2 | 15.8% | 38.1% | 5875 | 86567 | 906 | 1116 |
| AFN (Shen et al. 2018) | 49.0 | 48.2 | 19.1% | **35.7**% | 9508 | 82506 | 899 | 1383 |
| eTC (Wang et al. 2019) | 49.2 | **56.1** | 17.3% | 40.3% | 8400 | 83702 | 606 | 882 |
| NOTA (Chen et al. 2019) | 49.8 | 55.3 | 17.9% | 37.7% | 7248 | 83614 | 614 | 1372 |
| DHIAN(Ours) | **50.0** | 54.3 | **20.0**% | 37.8% | 5103 | 88551 | 617 | 839 |

**Table 3** Results on the DukeMTMCT test dataset

| Tracker | MOTA↑ | IDF1↑ | MT↑ | ML↓ | FP↓ | FN↓ | IDSw.↓ | Frag↓ |
|---|---|---|---|---|---|---|---|---|
| PT_BIPCC (Maksai et al. 2017) | 59.3 | 71.2 | 666 | 234 | 71381 | 361673 | **298** | 799 |
| BIPCC (Ristani et al. 2016) | 59.4 | 70.1 | 665 | 234 | 68634 | 361589 | 290 | **783** |
| MTMC_ReIDp (Zhang et al. 2017) | 70.7 | 79.2 | 726 | 143 | 52408 | 277762 | 449 | 1060 |
| MTMC_CDSC (Tesfaye et al. 2017) | 70.9 | 77.0 | 740 | 110 | 38655 | 268398 | 693 | 4717 |
| MYTRACKER (Yoon et al. 2018) | 73.8 | 80.3 | 914 | 72 | **35580** | 193253 | 406 | 1116 |
| TAREIDMTMC (Chen et al. 2017) | 83.3 | **83.8** | 1051 | 17 | 44691 | 131220 | 383 | 2428 |
| DAN(Ours) | **86.7** | 82.0 | **1088** | **9** | 37073 | **102930** | 928 | 4357 |

addition, we also calculate cosine similarity as $S_{\psi}$ between groups $\psi_{1\to5}$ (as shown in Fig. 10b). We firstly compare appearance feature similarity of the same target $I_{1\to5}$, where the value ranges between [0.556, 0.726]. However, due to the highly similar appearance between the target $I_{1\to5}$ and the first neighbor $I_6$, it's difficult to distinguish them by similarity value ([0.528, 0.659]). On the contrary, the interaction feature reserves some discriminative characteristics, the similarity value ranges between targets $\psi_{1\to5}$ is limited to [0.816,0.926], meanwhile the value between targets $\psi_{1\to5}$ and the first neighbor $\psi_6$ is decreased to [0.324, 0.471]. The similarity is already distinguishable in appearance feature (such as targets $I_{1\to5}$ and the second neighbor $I_7$), their interaction features $\psi_{1\to5}$ and $\psi_7$ also remain discriminative.

### 4.7 MOT Evaluation Metrics

The MOT Challenge Benchmark adopts the standard CLEAR-MOT mapping (Bernardin and Stiefelhagen 2008) and ID measures (Ristani et al. 2016) for evaluating MOT performance. The main metrics for MOT are MOTA and IDF1. MOTA (Multiple Object Tracking Accuracy) measures the effect of tracking for each tracklet, which depends on True Positives (TP) ,False Positives (FP), False Negatives (FN) and Id Switches (IDSw), $MOTA = 1 - \frac{FP+FN+IDS}{TP}$. Total number of Fragment (Frag), Mostly tracked targets (MT), Mostly lost targets (ML) are used for evaluating tracklet integrity as the reference indexes. The IDF1 (ID F1 Score) is the ratio of correctly identified detection over the average number of true and computed detections, which depends on ID True Positives (IDTP), ID False Positives (IDFP) and ID False Negatives (IDFN), $IDF1 = \frac{2*IDTP}{2*IDTP+IDFN+IDFP}$. IDP (ID Precision) indicates fraction of computed detections that are correctly identified, $IDP = \frac{IDTP}{IDTP+IDFP}$. IDR (ID Recall) means fraction of ground-truth detections that are correctly identified, $IDR = \frac{IDTP}{IDTP+IDFN}$.

### 4.8 Tracker Results Comparison

Our method is evaluated on the MOT Challenge testing set. We compare the performance with the state-of-the-art methods on the benchmark. The comparison of methods on MOT16 and DukeMTMCT are shown in Table 2 and Table 3. The most crucial evaluating parameters include MOTA and

**Table 4** Results on the MOT15&16 on static camera test set

| Tracker | MOT15-TUD-Crossing | | MOT15-PETS09-S2L2 | | Tracker | MOT16-01 | | MOT16-03 | | MOT16-08 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | MOTA↑ | IDF1↑ | MOTA↑ | IDF1↑ | | MOTA↑ | IDF1↑ | MOTA↑ | IDF1↑ | MOTA↑ | IDF1↑ |
| DAN (Sun et al. 2019) | 57.6 | 48.9 | 36.7 | 28.5 | AFN (Shen et al. 2018) | 29.8 | 39.0 | 57.0 | 49.5 | 32.4 | 37.2 |
| AP_HWDPL_p (Chen et al. 2017) | 61.3 | 64.1 | 38.9 | 34.3 | DAN (Sun et al. 2019) | 35.4 | 34.0 | 57.0 | 54.9 | 27.1 | 27.3 |
| FAMNet (Chu and Ling 2019) | 65.9 | 51.1 | 54.2 | 36.1 | STRN (Xu et al. 2019) | 36.7 | 49.4 | 55.6 | 57.9 | 33.5 | 42.7 |
| STRN (Xu et al. 2019) | 68.0 | 64.1 | 47.9 | 32.9 | KCF16 (Chu et al. 2019) | 36.9 | 36.3 | 57.2 | 50.1 | 28.8 | 33.7 |
| KCF (Chu et al. 2019) | 76.0 | 69.1 | 51.8 | 37.9 | eTC (Wang et al. 2019) | 37.4 | 47.0 | 57.6 | 61.7 | 36.2 | 44.6 |
| AMIR15 (Sadeghian et al. 2017) | 73.7 | 57.4 | 47.0 | 36.3 | TLMHT (Sheng et al. 2018) | 37.8 | 48.9 | 58.2 | 60.2 | 33.4 | **46.7** |
| Tracker++ (Bergmann et al. 2019) | 78.3 | 58.3 | 44.5 | 28.4 | LMP (Tang et al. 2017) | 33.9 | 51.2 | 56.1 | 53.3 | 33.8 | 37.4 |
| MPNTrack15 | 80.1 | 66.0 | 50.3 | 35.4 | FAMNet (Chu and Ling 2019) | 40.2 | 32.9 | 56.9 | 46.8 | 25.7 | 26.3 |
| JointMC (Keuper et al. 2018) | **80.9** | 60.2 | **56.0** | 41.1 | NOTA (Chen et al. 2019) | 42.0 | 48.4 | 57.1 | 59.7 | 35.4 | 44.4 |
| DHIAN(Ours) | 80.4 | **82.1** | 52.1 | **42.8** | DHIAN(ours) | **46.0** | **60.0** | **58.3** | **62.3** | **37.0** | 41.9 |

IDF1. Compared with the previous methods, our proposal aims to extract the more efficient features to avoid the "mis-tracking" and "lost-tracking". Consequently, Table 2 shows that DHIAN (ours) achieves the best performance on MOTA, MT. The others parameters (such as IDF1, ML, etc.) reaches relatively high score for MOT16 dataset. As shown in Table 3, our method exceeds the previous highest MOTA score by 3.3%, and DHIAN also achieves the excellent score on MT, ML, FN. The video from Duke-MTMCT dataset is captured from surveillance, but MOT16 blends moving videos and static videos. Since moving camera affects trajectory prediction and relationship construction, our methods has better performance on Duke-MTMCT than MOT16. Therefore, we specially compare performance on static video of MOT15 and MOT16 (as shown in Table 4). Our results ranks the 1st or 2nd place with MOTA and IDF1 among the start-of-the-art methods. Fig. 11 illustrates the DHIAN visualization results on MOT15 (the 1st-3rd rows), MOT16 (the 4th-6th rows), Duke-MTMCT (the 7th row) and MOT20 (the 8th row).

## 5 Conclusion and Future Work

In this paper, we concentrate on Multiple Object Tracking (MOT) in the wild. Frequent occlusion and rapid illumination variation influence the tracked- targets' feature description. In addition, existing methods use the hand-crafted method to optimize graph structure, which causes that feature extraction module and graph optimization module cannot be combined as a whole end-to-end network. Therefore, we specially consider the interrelation cue between objects and design Human-Interaction Model (HIM) to extract the details of target and its surrounding. Meanwhile we propose an efficient end-to-end model, Deep Association Network (DAN), to optimize the association with graph-based learning mechanism. Our proposal is evaluated on 4 public datasets (MOT15, MOT16, Duke-MTMCT and MOT20). The algorithm achieves MOTA up to 50.0, 86.7 and IDF1 up to 54.3, 82.0 on MOT16 and DukeMTMCT, respectively. The visualization results are shown in Fig. 11. Our method has better performance in complex crowd scene and static cameras.

In the future, we intend to optimize our model and improve processing efficiency. And we will continue to explore the novel model structure and relationship building strategy to extract interaction feature more effectively. To solve the frequent occlusion problem, we intend to combine the head-detector and body-detector as collaborative tracking of pedestrians. And we will design a newer motion encoder to replace the current RNN structure for trajectory prediction. Lastly, we attempt to utilize transfer learning to improve the model robustness.

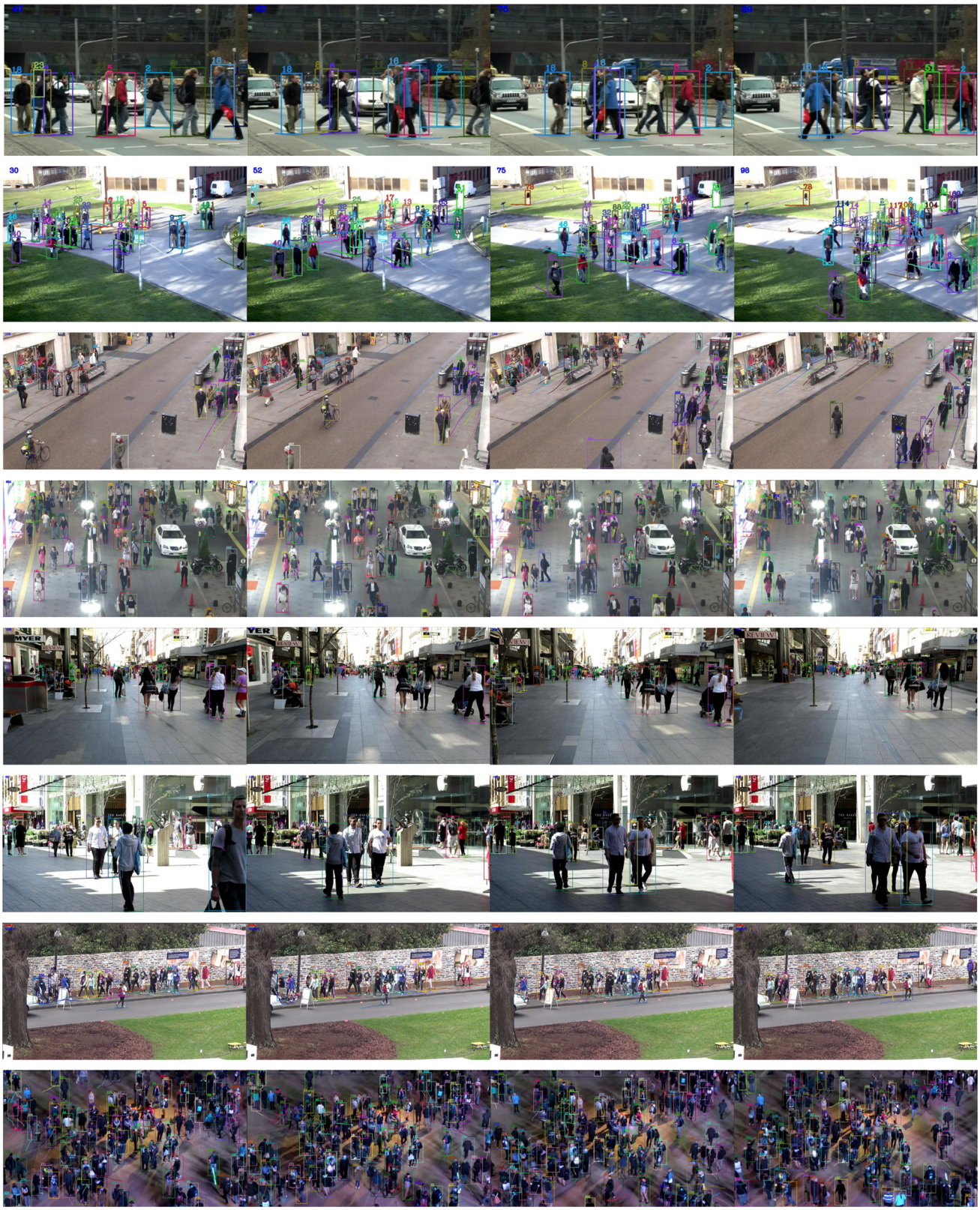

**Fig. 11** The visualization results on MOT15, MOT16, Duke-MTMCT and MOT20

## References


Leal-Taix, L., Milan, A., Reid, I., Roth, S., & Schindler, K. (2015). Motchallenge 2015: Towards a benchmark for multi-target tracking. arXiv preprint arXiv:1504.01942.

Milan, A., Leal-Taix, L., Reid, I., Roth, S., & Schindler, K. (2016). Mot16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831.

Ristani, E., Solera, F., Zou, R., Cucchiara, R., Tomasi, C.: Performance measures and a data set for multi-target, multi-camera tracking. In: *ECCV workshop on Benchmarking Multi-Target Tracking*. (2016)

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., & Leal-Taixe, L. (2019). Cvpr19 tracking and detection challenge: How crowded can it get? arXiv preprint arXiv:1906.04567.

Martín-Martín, R., Rezatofighi, H., Shenoi, A., Patel, M., Gwak, J., Dass, N., Federman, A., Goebel, P., & Savarese, S. (2019). Jrdb: A dataset and benchmark for visual perception for navigation in human environments. arXiv preprint arXiv:1910.11792.

Cai, Z., & Vasconcelos, N. (2018). Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6154–6162.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. 91–99.

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Sahbani, B., & Adiprawita, W. (2017). Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system. In: ICSET. 109–115.

Schulter, S., Vernaza, P., Choi, W., & Chandraker, M. (2017). Deep network flow for multi-object tracking. In: *CVPR*. 6951–6960.

Milan, A., Taix, L.L., Reid, I.D., Roth, S., & Schindler, K. (2016) MOT16: A benchmark for multi-object tracking. CoRR **abs/1603.00831**.

Henschel, R., Leal-Taix, L., Cremers, D., & Rosenhahn, B. (2018). Fusion of head and full-body detectors for multi-object tracking. In: *Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Tang, S., Andriluka, M., Andres, B., & Schiele, B. (2017). Multiple people tracking by lifted multicut and person reidentification. In: *CVPR*. 3539–3548.

Xiang, Y., Alahi, A., & Savarese, S. (2015). Learning to track: Online multi-object tracking by decision making. In: *ICCV*. 4705–4713.

Choi, W. (2015). Near-online multi-target tracking with aggregated local flow descriptor. In: *ICCV*. 3029–3037.

Kim, C., Li, F., Ciptadi, A., & Rehg, J.M. (2015). Multiple hypothesis tracking revisited. In *ICCV*. 4696–4704.

Chen, J., Sheng, H., Zhang, Y., & Xiong, Z. (2017). Enhancing detection model for multiple hypothesis tracking. In: *CVPR Workshops*. 18–27.

Bergmann, P., Meinhardt, T., & Leal-Taixe, L. (2019). Tracking without bells and whistles. *ICCV* .

Keuper, M., Tang, S., Andres, B., Brox, T., & Schiele, B. (2018). Motion segmentation & multiple object tracking by correlation co-clustering. *IEEE transactions on pattern analysis and machine intelligence, 42*(1), 140–53.

Chen, L., Ai, H., Chen, R., & Zhuang, Z. (2019). Aggregate tracklet appearance features for multi-object tracking. *IEEE Signal Processing Letters*.

Levinkov, E., Uhrig, J., Tang, S., Omran, M., Insafutdinov, E., Kirillov, A., Rother, C., Brox, T., Schiele, B., Andres, B.: Joint graph decomposition and node labeling: Problem, algorithms, applications. CVPR (2017)

Maksai, A., Wang, X., Fleuret, F., & Fua, P. (2017). Globally consistent multi-people tracking using motion patterns. ICCV .

Ma, C., Li, Y., Yang, F., Zhang, Z., Zhuang, Y., Jia, H., & Xie, X. (2019). Deep association: End-to-end graph-based learning for multiple object tracking with conv-graph neural network. In: *ICMR, ACM* ,253–261.

Shen, H., Huang, L., Huang, C., & Xu, W. (2018). Tracklet association tracker: An end-to-end learning-based association approach for multi-object tracking. arXiv preprint arXiv:1808.01562 .

Sadeghian, A., Alahi, A., & Savarese, S. (2017). Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *ICCV* .

Yang, F., Yan, K., Lu, S., Jia, H., Xie, X., & Gao, W. (2019). Attention driven person re-identification. *Pattern Recognition, 86,* 143–155.

Yang, F., Yan, K., Lu, S., Jia, H., Xie, D., Yu, Z., et al. (2020). Part-aware progressive unsupervised domain adaptation for person re-identification. *IEEE Transactions on Multimedia,* 1–1.

Yang, F., Yan, K., Lu, S., Jia, H., Xie, X., & Gao, W. (2019). Attention driven person re-identification. *Pattern Recognition, 86,* 143–155.

Yang, F., Yan, K., Lu, S., Jia, H., Xie, D., Yu, Z., et al. (2020). Part-aware progressive unsupervised domain adaptation for person re-identification. *IEEE Transactions on Multimedia*.

Son, J., Baek, M., Cho, M., & Han, B. (2017). Multi-object tracking with quadruplet convolutional neural networks. In: *CVPR*. 5620–5629.

Chu, Q., Ouyang, W., Li, H., Wang, X., Liu, B., & Yu, N. (2017). Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In: *CVPR*. 4836–4845

Ma, C., Yang, C., Yang, F., Zhuang, Y., Zhang, Z., Jia, H., & Xie, X. (2018). Trajectory factory: Tracklet cleaving and re-connection by deep siamese bi-gru for multiple object tracking. *ICME* .

Zhu, J., Yang, H., Liu, N., Kim, M., Zhang, W., & Yang, M.H. Online multi-object tracking with dual matching attention networks. In: *ECCV*. (September 2018)

Gao, X., & Jiang, T. (2018) . Osmo: Online specific models for occlusion in multiple object tracking under surveillance scene. In: *2018 ACM Multimedia Conference on Multimedia Conference*. 201–210.

Wang, G., Wang, Y., Zhang, H., Gu, R., & Hwang, J.N. (2019). Exploit the connectivity: Multi-object tracking with trackletnet. In: *Proceedings of the 27th ACM International Conference on Multimedia, ACM* .482–490.

Dicle, C., Camps, O.I., & Sznaier, M. (2013). The way they move: Tracking multiple targets with similar appearance. In: *ICCV*. 2304–2311.

Hong Yoon, J., Lee, C.R., Yang, M.H., & Yoon, K.J. (2016). Online multi-object tracking via structural constraint event aggregation. In: *CVPR*. 1392–1400.

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, & L., Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. In: *CVPR*. 961–971.

Chen, X., Treiber, M., Kanagaraj, V., & Li, H. (2018). Social force models for pedestrian traffic-state of the art. *Transport reviews, 38*(5), 625–653.

Yang, D., Redmill, K., & Ozguner, U. (2020). A multi-state social force based framework for vehicle-pedestrian interaction in uncontrolled pedestrian crossing scenarios. arXiv preprint arXiv:2005.07769 .

Zhang, M., Li, T., Yu, Y., Li, Y., Hui, P., & Zheng, Y. (2020). Urban anomaly analytics: Description, detection and prediction. IEEE Transactions on Big Data .

Cai, L., Chen, Z., Luo, C., Gui, J., Ni, J., Li, D., & Chen, H. (2020). Structural temporal graph neural networks for anomaly detection in dynamic graphs. arXiv preprint arXiv:2005.07427.

Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, H., & Savarese, S. (2019). Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1349–1358.

Kosaraju, V., Sadeghian, A., Martín-Martín, R., Reid, I., Rezatofighi, H., &Savarese, S. (2019). Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In: Advances in Neural Information Processing Systems. 137–146

Lan, L., Wang, X., Zhang, S., Tao, D., Gao, W., & Huang, T. S. (2018). Interacting tracklets for multi-object tracking. *IEEE Transactions on Image Processing, 27*(9), 4585–4597.

Wang, X., Türetken, E., Fleuret, F., & Fua, P. (2015). Tracking interacting objects using intertwined flows. *IEEE transactions on pattern analysis and machine intelligence, 38*(11), 2312–2326.

Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., & Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 .

Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2016). Gated graph sequence neural networks. *ICLR* .

Kipf, T.N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *ICLR* .

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. Graph attention networks. ICLR (2018) accepted as poster.

Duvenaud, D.K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., & Adams, R.P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In: Advances in neural information processing systems. 2224–2232.

Kipf, T., Fetaya, E., Wang, K.C., Welling, M., & Zemel, R. (2018). Neural relational inference for interacting systems. ICML .

Garcia, V., & Bruna, J. (2018). Few-shot learning with graph neural networks. ICLR.

Acuna, D., Ling, H., Kar, A., & Fidler, S. (2018). Efficient interactive annotation of segmentation datasets with polygon-rnn++. In: CVPR. 859–868.

Yan, S., Xiong, & Y., Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. AAAI .

Shen, Y., Li, H., Yi, S., Chen, D., & Wang, X. (2018). Person re-identification with deep similarity-guided graph neural network. In: *ECCV*, Springer .508–526.

Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., & Tian, Q. (2015). Scalable person re-identification: A benchmark. In: *Proceedings of the IEEE International Conference on Computer Vision*. 1116–1124

Zheng, Z., Zheng, L., & Yang, Y. (2017). Unlabeled samples generated by gan improve the person re-identification baseline in vitro. arXiv preprint arXiv:1701.07717 **3**.

Zhong, Z., Zheng, L., Cao, D., & Li, S. (2017). Re-ranking person re-identification with k-reciprocal encoding. In: Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE .3652–3661.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE TPAMI, 32*(9), 1627–1645.

Cao, Z., Hidalgo, G., Simon, T., Wei, S.E., & Sheikh, Y. (2018). Openpose: realtime multi-person 2d pose estimation using part affinity fields. arXiv preprint arXiv:1812.08008 .

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. ICLR .

Wang, B., Wang, L., Shuai, B., Zuo, Z., Liu, T., Luk Chan, K., & Wang, G. (2016) . Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association. In: *CVPR Workshops*. 1–8

Long, C., Haizhou, A., Zijie, & Z., Chong, S. (2018). Real-time multiple people tracking with deeply learned candidate selection and person re-identification. *ICME*

Henschel, R., Leal-Taix, L., Cremers, & D., Rosenhahn, B. (2017). A novel multi-detector fusion framework for multi-object tracking. CoRR .

Xu, J., Cao, Y., Zhang, Z., & Hu, H. (2019). Spatial-temporal relation networks for multi-object tracking. arXiv preprint arXiv:1904.11489 .

Sheng, H., Chen, J., Zhang, Y., Ke, W., Xiong, Z., & Yu, J. (2018). Iterative multiple hypothesis tracking with tracklet-level association. *IEEE Transactions on Circuits and Systems for Video Technology*.

Chu, P., Fan, H., Tan, C.C., & Ling, H. (2019). Online multi-object tracking with instance-aware tracker and dynamic model refreshment. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE* . 161–170

Maksai, A., Wang, X., Fleuret, F., & Fua, P. (2017). Non-markovian globally consistent multi-object tracking. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE , 2563–2573.

Ristani, E., Solera, F., Zou, R., Cucchiara, R., & Tomasi, C. (2016). Performance measures and a data set for multi-target, multi-camera tracking. In: *European Conference on Computer Vision*, Springer . 17–35.

Zhang, Z., Wu, J., Zhang, X., & Zhang, C. (2017). Multi-target, multi-camera tracking by hierarchical clustering: Recent progress on dukemtmc project. arXiv preprint arXiv:1712.09531 .

Tesfaye, Y.T., Zemene, E., Prati, A., Pelillo, M., & Shah, M. (2017). Multi-target tracking in multiple non-overlapping cameras using constrained dominant sets. arXiv preprint arXiv:1706.06196 .

Yoon, K., Song, Y.m., & Jeon, M. (2018). Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views. IET Image Processing .

Sun, S., Akhtar, N., Song, H., Mian, A. S., & Shah, M. (2019). Deep affinity network for multiple object tracking. *IEEE transactions on pattern analysis and machine intelligence*.

Chen, L., Ai, H., Shang, C., Zhuang, Z., & Bai, B. (2017). Online multi-object tracking with convolutional neural networks. In: *2017 IEEE International Conference on Image Processing (ICIP)*, IEEE , 645–649.

Chu, P., & Ling, H. (2019). Famnet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking. In: *Proceedings of the IEEE International Conference on Computer Vision*. 6172–6181

Bernardin, K., & Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing, 2008*(1), 246309.