Graph Networks for Multiple Object Tracking

Jiahe Li* Xu Gao* Tingting Jiang
NELVT, Department of Computer Science, Peking University, China
{jiaheli, gaoxu1024, ttjiang}@pku.edu.cn

Abstract

Multiple object tracking (MOT) task requires reasoning the states of all targets and associating these targets in a global way. However, existing MOT methods mostly focus on the local relationship among objects and ignore the global relationship. Some methods formulate the MOT problem as a graph optimization problem. However, these methods are based on static graphs, which are seldom updated. To solve these problems, we design a new near-online MOT method with an end-to-end graph network. Specifically, we design an appearance graph network and a motion graph network to capture the appearance and the motion similarity separately. The updating mechanism is carefully designed in our graph network, which means that nodes, edges and the global variable in the graph can be updated. The global variable can capture the global relationship to help tracking. Finally, a strategy to handle missing detections is proposed to remedy the defect of the detectors. Our method is evaluated on both the MOT16 and the MOT17 benchmarks, and experimental results show the encouraging performance of our method.

1. Introduction

Multiple object tracking (MOT) is a crucial problem in computer vision because of its academic and commercial potential. However, it is still tough and challenging to deal with MOT problem, due to factors like similar appearance, abrupt appearance changes, frequent occlusions, interactions among multiple object and large camera motion.

Most of graph models [48, 29, 5, 41, 6, 42] are often static graphs. Intuitively, we can construct a graph by treating the objects and the detections as nodes and the associations between objects and detections as edges. Based on the graph, these graph models formulate MOT problem as the min-cost network flow optimization problem. These static graph models will fail when the information contained by nodes or edges is outdated. If these graph models had been able to update the nodes and edges, a better tracking performance would be achieved. To this end, nodes and edges need to be updated. Moreover, these static graph models do not model the global interaction among objects.

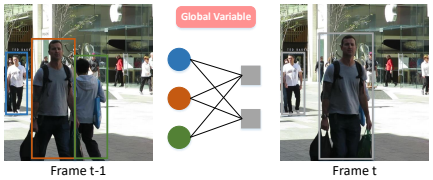

Figure 1. The graph network consists of colorful circle nodes (objects in frame $t-1$), grey square nodes (detections in frame t), edges (connections between objects and detections) and a global variable. Our method can update all components iteratively.

Recently, it is shown that the graph network has the ability of reasoning by comprehensive experiments [2]. As shown in Fig. 1, the graph network consists of three parts, including the node, the edge and the global variable. The graph network updates nodes and edges iteratively and reasonably, which can remedy the defect of previous graph models. In the graph network, the global variable is expected to capture the global relationship among all nodes and edges through the updating mechanism, such as interaction among nodes. Hence, the graph network provides a solution to the MOT problem. However, with an improper design of the graph network, directly applying the graph network on the MOT problem will fail.

Appearance and motion are essential cues for dealing with MOT problem. Many factors result in poor appearances, such as, similar appearance, abrupt appearance changes and frequent occlusions. During that time, motion-based models perform better than appearance-based models. When large camera motion and inaccurate detections cause bad motion estimation, appearance-based models are more robust than motion-based models. Hence, we design two graph networks, including the appearance graph network and the motion graph network, which use the appearance feature and the motion feature as the attribute of nodes respectively. The interaction is also important for dealing with MOT problem. In the graph network, the global variable is expected to capture the interaction among objects.

*Contributed equally. Xu Gao is currently working at Baidu. Source codes of this paper are available: https://github.com/yinizhizhu/GNMOT.

We propose a new near-online MOT method with two end-to-end graph networks inspired by [3] which provides a new basic graph network framework. We carefully design our graph networks according to the unique characteristic of MOT problem. For the two end-to-end graph networks which handle appearance and motion respectively, we design four updating modules to get the similarity scores between objects and detections for association. Specifically, it contains a node updating module, a global variable updating module, and two edge updating modules. In fact, the updating mechanism contains the inference of the graph network. Besides, it can also keep the historical information of nodes through the long-term tracking. Finally, to remedy the defect of the detectors, two strategies for handling missing detections are proposed. It includes a single object tracker (SOT) strategy and a detection recovery strategy. Our method is evaluated on both the MOT16 and the MOT17 benchmarks, and experimental results show the encouraging performance. To our best knowledge, this is the first paper that applies the graph network on MOT problem.

The main contributions are as follows: (1) We propose a new near-online MOT method with an end-to-end graph network framework followed by strategies for handling missing detections. (2) The updating mechanism is carefully designed in our graph networks, which allows the inference of the graph network.

2. Related Works

Multiple Object Tracking. In recent works, many existing MOT methods follow the tracking-by-detection strategy, which tackle MOT problem by running a detector first and associate those detections afterwards. These methods can be categorized into offline approaches, online approaches and near-online approaches. Offline approaches use detections from the entire sequence, and then conduct a global optimization, including graph optimization methods [48, 29, 4, 5, 6] and hierarchical methods [47, 41, 42, 32, 25]. In comparison, online approaches only use detections from both the current frame and the previous frames. Recent online MOT works model MOT problem as a data association problem between the tracked objects and the detections. The key is to evaluate the similarity between each tracked object and each detection by using different networks (e.g. Recurrent Neural Networks [27, 12, 33, 49, 16] and Reinforcement Learning [43, 30]) and different mechanisms (e.g. Attention mechanism [10, 49, 16] and the single object tracking mechanism for pre-processing [10, 49]). Recently, some near-online MOT methods are proposed, which are similar to the online MOT approaches but allow to re-associate objects in recent frames. Choi [8] proposes an aggregated local flow descriptor to provide a robust similarity measure for association. Fagot-Bouquet *et al.* [11] combine a sparse representation-based appearance model to improve multi-frame data association. Kim *et al.* [20] utilize an LSTM network to score the track proposals in a near-online multiple hypothesis tracking framework. However, these online/near-online methods do not consider the global relationship between all tracked objects and detections.

Graph-Related Methods. Some offline methods use graph models to formulate the MOT problem [48, 29, 5, 41, 6, 42, 35, 44]. Although most of these methods can build connections between objects, they are lack of reasoning among objects, and nodes/edges cannot be updated.

Due to the ability of reasoning, graph networks have been designed to solve some problems in other computer vision areas. For instance, Shen *et al.* [36] propose a graph network to solve the person re-identification problem, and the graph network is used to guide the similarity between each probe-gallery pair. Many other graph network methods are concluded into a unified structure by [3]. In general cases, the graph network consists of nodes, edges and the global variable. These components are updated iteratively, which is regarded as the inference process.

Our model can infer the relationship among objects. Besides, our model can update nodes and edges, which is more flexible than traditional graph models.

3. Problem Formulation

3.1. Notions

The following notions are defined at frame t.

Trajectory: A trace formulated by bounding boxes from all the frames between frame 1 and frame $t-1$.

Detection: A bounding box in current frame t. The detection set is denoted by $\mathcal{D}_t$.

Object: The last bounding box on one trajectory. Objects before frame t comprise the object set for frame t, which is denoted by $\mathcal{O}_t$. If the object occurred in frame $t-1$, it is a **normal object**. Otherwise, it is a **missing object**.

3.2. Pipeline

The pipeline is demonstrated in Fig. 2. As shown in Fig. 2, there are four procedures: feature extraction, graph networks, data association and missing detections handling.

Feature Extraction. First, we extract the appearance features and the motion features from both objects and detections. To be specific, the appearance feature is extracted from a convolutional neural network (CNN). The motion feature is a 6-dimension vector, including the 2D coordinates of the top-left corner, width, height and 2D velocity of the object/detection which is computed from the displacement between the detection and the object.

Graph Networks. Afterwards, graph networks infer the similarity between each object and each detection. Each node is associated with the feature of the object/detection,

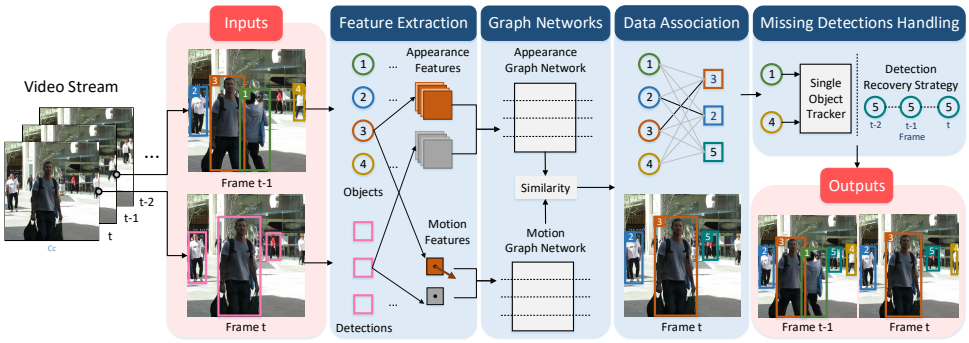

Figure 2. Pipeline of our MOT model. Inputs are the frame $t-1$, the frame t, the object set $\mathcal{O}_t$ from frame t and the detection set $\mathcal{D}_t$ at frame t. Outputs are tracking results. For example, there are four objects from frame $t-1$ and three detections from frame t, and their features are extracted and sent into the graph networks. Afterwards, data association and missing detections handling are performed.

and each edge that connects the object and the detection is associated with their similarity score.

Data Association. This procedure outputs the associations between the objects and the detections. The Hungarian algorithm [22] is used to find the optimal assignments. Note that we abandon those associations where the object is spatially very far from the detection.

Missing Detections Handling. After the data association procedure, there are still some missing detections. For those objects that have been missed in the current frame, the SOT strategy is used to track those missing objects in the current frame, and associate them with the recovered bounding boxes by SOT with high confidence score. For those detections that have been missed for a while, we use a detection recovery strategy, which applies a linear motion model to recover those missing detections.

We denote the i-th object in $\mathcal{O}_t$ as o_i, and the j-th detection in $\mathcal{D}_t$ as d_j. The state of assignment between o_i and d_j is denoted by $a_{i,j}$, where $a_{i,j}=1$ describes the situation that the detection d_j is associated with the object o_i, and $a_{i,j}=0$ describes the opposite situation. The assignment set is denoted by $\mathcal{A}_t=\{a_{i,j}\}^{|\mathcal{O}_t|\times|\mathcal{D}_t|}$, where $|\mathcal{O}_t|$ and $|\mathcal{D}_t|$ are the numbers of objects and detections respectively.

Then the optimal assignment set can be formulated by

$$\hat{\mathcal{A}}_t = \text{argmin}_{\mathcal{A}_t} \sum_{i=1}^{|\mathcal{O}_t|}\sum_{j=1}^{|\mathcal{D}_t|} a_{i,j}\mathcal{F}(o_i,d_j), \tag{1}$$

$$s.t. \sum_{i=1}^{|\mathcal{O}_t|} a_{i,j} \le 1 \text{ and } \sum_{j=1}^{|\mathcal{D}_t|} a_{i,j} \le 1, \tag{2}$$

where $\mathcal{F}(o_i,d_j)$ denotes the cost between the object o_i and the detection d_j. The constraints (2) describe that one object can be associated with at most one detection, and one detection can be associated with at most one object. Following the constraints, it is allowed that $\sum_{i=1}^{|\mathcal{O}_t|} a_{i,j}=0$ and $\sum_{j=1}^{|\mathcal{D}_t|} a_{i,j}=0$, which means that detections are associated with no objects, and objects are missing at current frame.

4. Graph Networks

The cost function in Eqn. (1) is calculated by

$$\mathcal{F}(o_i,d_j) = \alpha\text{AGN}(f_a^{o_i}, f_a^{d_j}) + (1-\alpha)\text{MGN}(f_m^{o_i}, f_m^{d_j}) \tag{3}$$

where $\text{AGN}(\cdot)$ and $\text{MGN}(\cdot)$ denote the appearance graph network and the motion graph network respectively, which are illustrated in Fig. 4. α is the hyperparameter. $f_a^{o_i}$ and $f_a^{d_j}$ denotes the appearance feature of the object o_i and the detection d_j respectively, and $f_m^{o_i}$ and $f_m^{d_j}$ denotes the motion feature of o_i and d_j respectively.

4.1. Preliminary

Battaglia *et al.* [3] conclude recent works on graph networks and present a new basic graph network (GN) framework, which is similar to Fig. 4 (a) but without the stage A. The unit of computation in the GN is the GN block, which takes a graph as input, computes over the structure, outputs an updated graph. During the computation, nodes, edges and the global variable are updated sequentially. Note that the update order can be tailored to the demands of the task. Inspired by [3], we propose two end-to-end graph networks to solve the MOT problem. Before introducing these two graph networks, we first describe the whole graph network.

4.2. Structures of the Graph Network

The graph network structure and the update order are designed carefully according to the unique characteristic of the MOT. We design a 4-step graph network, which moves the edge updating module to the end of [3]'s structure. The reason is that there is no ground truth of nodes and the global variable, so that this network is only supervised by edges for updating nodes and the global variable. Therefore, it is better to update edges in the end. Meanwhile,

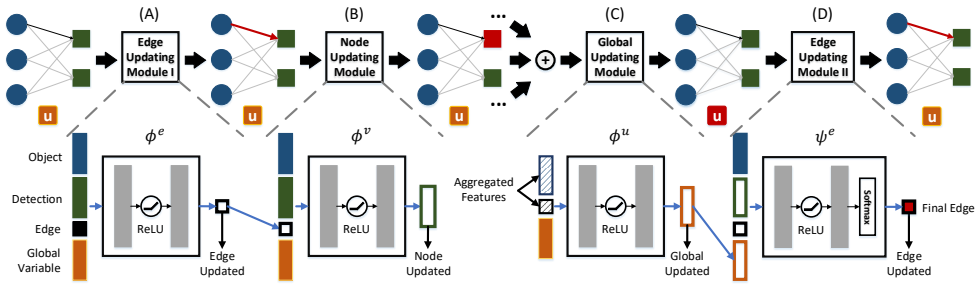

Figure 3. **Upper Part:** The structure of the 4-step graph network. Blue solid circles, green solid squares and orange rounded rectangles denote object set, detection set and the global variable respectively. Updated components of each module are marked in red. **Lower Part:** The corresponding networks for the four modules. For each module, there is a big square box denoting the neural network. Inside the box, the grey rectangles denote FC layers. The left side and right side denote the input and output. The blue, green, black, orange rectangles denote features of object nodes, detection nodes, edges and the global variable respectively. The hollow rectangles with black, green and orange edges denote updated features of edges, detection nodes and the global variable. The rectangles with textures are aggregated features. The red square with the black outline is the final edge feature.

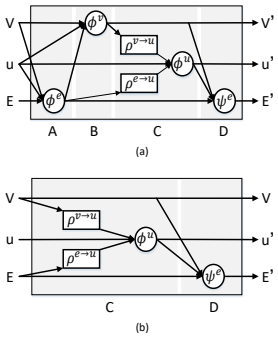

Figure 4. (a) The structure of our 4-step graph network. It is also the structure of the appearance graph network. Stage A, B, C, D denote the edge updating module I, the node updating module, the global updating module and the edge updating module II respectively. (b) The structure of the motion graph network, which only contains stage C and stage D.

one more edge updating module should be added to update edges at first, because the node updating module depends on updated edges rather than initialized edges. Hence, we design the following graph network structure, which contains four modules, including (A) edge updating module I ϕ^e, (B) node updating module ϕ^v, (C) global updating module ϕ^u, and (D) edge updating module II ψ^e, which are shown in Fig. 4 (a).

Denote V and E as the node set and the edge set respectively, and each node is represented by a feature. Here,

$$V = \{v_p^s, v_q^r | p = 1, ..., |\mathcal{O}_t|,\ q = 1, ..., |\mathcal{D}_t|\}, \quad (4)$$

$$E = \{e_k | k = 1, ..., K\}, \quad (5)$$

where v_p^s denotes the p-th object, v_q^r denotes the q-th detection, and e_k denotes the k-th edge/relationship between one object and one detection. $K = |\mathcal{O}_t| \times |\mathcal{D}_t|$ denotes the total number of object-detection pairs. Besides, the global variable u is expected to encode information of all objects, detections, and assignment states. In addition, it is necessary to consider all nodes and edges for updating the global variable. Hence, two aggregating functions $\rho^{v\to u}$ and $\rho^{e\to u}$ are used to aggregate all nodes and all edges respectively.

(A) Edge Updating Module I. Inputs are the object node, the detection node, the edge and the global variable. The output is the updated edge.

For simplicity, we denote the object node and the detection node that are connected by e_k as v_k^s and v_k^r respectively. Then, the updated edge e_k^* can be calculated by

$$e_k^* = \phi^e(v_k^s, v_k^r, e_k, u) = \text{NN}_\phi([v_k^s, v_k^r, e_k, u]), \quad (6)$$

where $\text{NN}_\phi(\cdot)$ is a neural network whose structure consists of two fully connected (FC) layers and a Leaky ReLU function in the middle. The input features are concatenated and sent into $\text{NN}_\phi(\cdot)$.

(B) Node Updating Module. This module merges historic features into detection nodes. Inputs are the object node, the detection node, the updated edge and the global variable. The output is the updated detection node. The updated detection node $\tilde{v}_k^r$ can be calculated by

$$\tilde{v}_k^r = \phi^v(v_k^s, v_k^r, e_k^*, u) = \text{NN}_\phi([v_k^s, v_k^r, e_k^*, u]), \quad (7)$$

where $\text{NN}_\phi(\cdot)$ has the same structure as the module (A).

(C) Global Updating Module. Inputs are the global variable, the aggregated node and the aggregated edge. Output is the updated global variable.

We aggregate all object and detection nodes and all updated edges. Denote $\overline{V}$ and $\overline{E}$ as the aggregated node and the aggregated edge respectively. These aggregated features can be calculated by

$$\overline{V} = \frac{1}{2K}\left(\sum_{k=1}^{K} v_k^s + \sum_{k=1}^{K} \tilde{v}_k^r\right), \overline{E} = \frac{1}{K}\sum_{k=1}^{K} e_k^*. \quad (8)$$

This aggregating process considers all associations.

$\overline{V}$ and $\overline{E}$ will be sent into the global updating module afterwards, together with the original global variable. Hence, the updated global variable $\tilde{u}$ can be calculated by

$$\tilde{u} = \phi^u(\overline{V}, \overline{E}, u) = \text{NN}_\phi([\overline{V}, \overline{E}, u]), \tag{9}$$

where $\text{NN}_\phi(\cdot)$ has the same structure as the module (A).

(D) Edge Updating Module II. Inputs are the object node, the updated detection node, the updated edge and the updated global variable. The output is the final edge. The final edge $\tilde{e}_k$ can be calculated by

$$\tilde{e}_k = \psi^u(v_k^s, \tilde{v}_k^r, e_k^*, \tilde{u}) = \text{NN}_\psi([v_k^s, \tilde{v}_k^r, e_k^*, \tilde{u}]), \tag{10}$$

where $\text{NN}_\psi(\cdot)$ has the similar structure as the module (A), but adding a softmax layer after the last FC layer to get the similarity score.

4.3. Appearance Graph Network

Appearance graph network measures the appearance similarity between each object and each detection. Inputs are appearance features of all objects and all detections, and outputs are the similarity scores of all object-detection pairs. The appearance feature uses the updated detection appearance feature from the last timestep, since it is updated at each timestep. Note that these updated nodes maintain the trajectory information at each timestep.

To get a robust appearance similarity, we design the 4-step graph network, which is illustrated in Fig. 4 (a). Each output edge is regarded as the appearance similarity, which is shown as $\text{AGN}(\cdot)$ in Eqn. (3).

4.4. Motion Graph Network

Motion graph network measures the motion similarity between each object and each detection. Inputs are motion features of all objects and all detections, and outputs are the similarity scores of all object-detection pairs. As mentioned in Sec. 3.2, the velocity of the object is estimated according to the trajectory, thus there is no need for node updating. Hence, we design a similar structure as the appearance graph network, but remove the first edge updating module and the node updating module, which is illustrated in Fig. 4 (b). Each output edge is regarded as the motion similarity, which is shown as $\text{MGN}(\cdot)$ in Eqn. (3).

4.5. Training Strategy

We use an online strategy to train these two graph networks, so that frames are selected sequentially while training. For training the appearance graph network, we design a 2-step training strategy. First, the edge updating module I is trained until convergence. Then, the latter three modules get training. For training the motion graph network, we train the whole network directly.

We train our graph networks with the cross entropy loss. Besides, we design a node cost for the node updating module. If the object-detection pair belongs to the same person, the detection node will be updated. Otherwise, the detection node is expected to be unchanged. Denote L as the final training loss, which can be formulated by

$$L = L_C + \lambda L_N, \tag{11}$$

where L_C and L_N denote the cross entropy loss and the node cost respectively. λ, which can adjust the weight of L_N, is set as 1. Here, L_C can be calculated by

$$L_C = -p \log \hat{p} - (1-p) \log(1-\hat{p}), \tag{12}$$

where p denotes the ground truth association between each object and each detection. $p = 1$ indicates that the detection is associated with the object, and $p = 0$ indicates the opposite situation. $\hat{p}$ denotes the predicted p. L_C is applied for the edge updating module I and the edge updating module II. Besides, the node cost L_N is designed as

$$L_N = (1-p) \times \text{MSE}(v, \tilde{v}), \tag{13}$$

where v is the original feature of the detection node, and $\tilde{v}$ is the updated v. $\text{MSE}(\cdot, \cdot)$ is the mean squared error function.

5. Experiments

5.1. Datasets and Evaluation Metrics

MOTChallenge is one of the public MOT benchmark platform, where many state-of-the-art methods have been evaluated. Specifically, MOT16 [28] and MOT17 are the most popular benchmarks in *MOTChallenge*. Both of these two benchmarks contain same sequences, including 7 training sequences and 7 testing sequences with crowd scenarios, different viewpoints and camera motions. The difference between these two benchmarks is that the MOT16 only provides detection results from DPM [13], while the MOT17 provides detection results from three different detectors, including DPM [13], Faster R-CNN [31] and SDP [45]. Moreover, the MOT17 provides more accurate ground truth for all the sequences.

In addition, a validation set is built for the ablation study and tuning hyperparameters. Specifically, we divide each training sequence from the MOT17 into two parts, named as Set A (first 4/5 frames) and Set B (last 1/5 frames). We use Set A as the training set and Set B as the validation set. Note that in the testing procedure, we use all training sequences from the MOT17 to train the model.

We use metrics proposed in [23] to evaluate the MOT performance, including MOTA (MOT Accuracy), MOTP (MOT Precision), IDF1 (ID F1-Measure), MT (Mostly Tracked Target Percentage), ML (Mostly Lost Target Percentage), FP (False Positives), FN (False Negatives), IDS (Identity Switch) and FM (Fragment). MOTA is the most important metric, since it combines FP, FN and IDS.

	Detection	Methods	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	FM↓
Offline Approach	Public	EDMT17[7], *CVPRW 2017*	45.3	47.9	17.0%	39.9%	11122	87890	639	946
		TLMHT[37], *TCSVT 2018*	48.7	55.3	15.7%	44.5%	6632	86504	413	642
		LMP[39], *CVPR 2018*	48.8	51.3	18.2%	40.1%	6654	86245	481	595
		HCC[26], *ACCV 2018*	49.3	50.7	17.8%	39.9%	5333	86795	391	535
Online Approach	Public	STAM[10], *CVPRW 2017*	46.0	50.0	14.6%	43.6%	6895	91117	473	1422
		DMAN[49], *ECCV 2018*	46.1	54.8	17.4%	42.7%	7909	89874	532	1616
		AMIR[33], *ICCV 2017*	47.2	46.3	14.0%	41.6%	2681	92856	774	1675
		JCSTD[40], *TITS 2019*	47.4	41.1	14.4%	36.4%	8076	86638	1266	2697
		KCF[9], *WACV 2019*	48.8	47.2	15.8%	38.1%	5875	86567	906	1116
Near-online Approach	Public	LINF1[11], *ECCV 2016*	41.0	45.7	11.6%	51.3%	7896	99224	430	963
		MHT_bLSTM*[20], *ECCV 2018*	42.1	47.8	14.9%	44.4%	11637	93172	753	1156
		NOMT[8], *ICCV 2015*	46.4	**53.3**	**18.3%**	41.4%	9753	87565	**359**	**504**
		Ours *without* SOT	47.4	42.6	14.5%	34.4%	**7795**	86178	1931	3389
		Ours	**47.7**	43.2	16.1%	**34.3%**	9518	**83875**	1907	3376
Near-online Approach	Private (POI[46])	**Ours *without* SOT**	**58.4**	**54.8**	**27.3%**	**23.2%**	**5731**	**68630**	**1454**	**1730**

Table 1. Experiments on MOT16 test set. For the near-online approach, the best result in each metric is highlighted in bold, and the second best result is underlined. * indicates the use of additional training data.

	Detection	Methods	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	FM↓
Offline Approach	Public	EDMT17*[7], *CVPRW 2017*	50.0	51.3	21.6%	36.3%	32279	247297	2264	3260
		MHT_DAM[19], *ICCV 2015*	50.7	47.2	20.8%	36.9%	22875	252889	2314	2865
		jCC[18], *TPAMI 2018*	51.2	54.5	20.9%	37.0%	25937	247822	1802	2984
		eHAF17[38], *TCSVT 2019*	51.8	54.7	23.4%	37.9%	33212	236772	1834	2739
Online Approach	Public	EAMTT[34], *ECCV 2016*	42.6	41.8	12.7%	42.7%	30711	288474	4488	5720
		PHD_GSDL17[15], *Acess 2018*	48.0	49.6	17.1%	35.6%	23199	265954	3998	8886
		AM_ADM17[24], *Acess 2018*	48.1	52.1	13.4%	39.7%	25061	265495	2214	5027
		DMAN[49], *ECCV 2018*	48.2	55.7	19.3%	38.3%	26218	263608	2194	5378
		MTDF[14], *TMM 2019*	49.6	45.2	18.9%	33.1%	37124	241768	5567	9260
Near-online Approach	Public	MHT_bLSTM*[20], *ECCV 2018*	47.5	**51.9**	18.2%	41.7%	25981	268042	**2069**	**3124**
		Ours *without* SOT	50.1	46.3	18.6%	33.3%	**25210**	250761	5470	8113
		Ours	**50.2**	47.0	**19.3%**	**32.7%**	29316	**246200**	5273	7850

Table 2. Experiments on MOT17 test set. For the near-online approach, the best result in each metric is highlighted in bold, and the second best result is underlined. * indicates the use of additional training data.

5.2. Implementation Details

All the experiments are conducted on Linux with the Intel 3.6GHz CPU and a NVIDIA GTX 1070 GPU.

Network Details. Input images are first resized into 224 $\times$ 224. Afterwards, the pre-trained ResNet-34 [17] is used to extract the appearance feature. The size of the FC layer in each updating module is set as 256, and the negative slop in Leaky ReLU is set as 10^{-2}. The global variable is randomly initialized between 0 and 1, and its size is set as 100. The edge feature is initialized with the Intersection over Union (IoU) between the object and the detection, and its size is set as 2. The size of the appearance feature and the motion feature are set as 512 and 6 respectively. Besides, we set $\alpha = 0.3$ in Eqn. (3) based on the results on validation set, where α is selected from 0.1 to 0.9 with step-size 0.1.

Training Details. We use the Adam [21] optimizer to train the appearance graph network and the motion graph network. Learning rates are set as 10^{-5} and 5×10^{-4} for the appearance and the motion graph network respectively. The batch training strategy is used, and we set the batchsize as 8. To simulate detections from the real detectors, we randomly sample bounding boxes around the ground truth, which has a large IoU with the ground truth. The IoU between the sample box and the ground truth is set as 0.85.

Occlusion Handling. When the object is mostly occluded, it is prone to getting lost, and ID switch may happen when the object reappears later. To deal with such a challenge, we store those missing objects from the current frame, and keep associating them with detections in the next timesteps. Therefore, those missing objects from previous frames are equally regarded as objects at the current frame, which will be sent into the graph network as well. Considering the computational efficiency, if the object is missing over 25 frames, it will be removed from the graph network. Intuitively, the probability for associating those missing objects with detections should decrease when it is missed for a long time. Hence, we update the cost function for those missing objects by $\tilde{\mathcal{F}}(o_i, d_j) = \mathcal{F}(o_i, d_j) \times \eta^{\Delta t-1}$, where Δt is the temporal gap between the missing object and the detection. η is set as 1.3 based on the results on the validation set, where η is selected from 1.1 to 1.9 with step-size 0.1. $\tilde{\mathcal{F}}(o_i, d_j)$ will replace $\mathcal{F}(o_i, d_j)$ in Eqn. (1).

Missing Detections Handling. Two strategies of han-

dling the missing detections have been described in Sec. 3.2. First, we use the SOT [50] for those objects that are not associated with any detection at the current frame. To keep the robustness of the SOT results, we abandon those SOT predictions which have the confidence score less than 0.98. For the detection recovery strategy, we recover those missing detections that have been missed within 16 frames.

5.3. Results on the MOT16 and MOT17 Benchmark

We evaluate our method on the MOT16 and MOT17 benchmark. Experimental results are shown in Table 1 and Table 2. On MOT16, we can see that our approach achieves the best result on MOTA, ML and FN on near-online MOT, and achieves the second best result on MT, which shows that our method can generate longer trajectories for targets. Compared with the four offline MOT methods, our method is competitive on MOTA and achieves better results on ML and FN. Compared with the five online MOT methods, our method achieves the second best result on MOTA and MT and the best result on ML and FN. Specifically, KCF [9] integrates SOT in MOT with an adaptive model refreshment strategy to achieve state-of-the-art performance, while our method only uses SOT to handle the missing detections. Since the near-online method uses more frames than the online method, the comparison with online methods is only for reference. On MOT17, our method has better performance on MOTA, MT, ML and FN than MHT_bLSTM [20]. Note that MHT_bLSTM is the only one near-online method which provides the results on the MOT17 benchmark. In addition, on MOT17, compared with our graph network without SOT, our method performs better in all metrics except the FP, which shows that SOT can improve the performance of our baseline method. In summary, our method achieves state-of-the-art performance on near-online MOT on the MOT16 and MOT17 benchmarks, and the single object tracker can improve the performance.

Since the detection results influence the tracking performance significantly, we evaluate our method with a better detection POI [46] on MOT16. To eliminate the impact of SOT on our method, we evaluate our method without SOT. As shown in Table 1, with the POI detection, our method achieves better performance than our method with public detection on MOT16. Hence, the better detection is, the better our method performs.

5.4. Ablation Study

Three ablation studies are conducted. The first one is to show why we choose different networks for the appearance graph network and the motion graph network. The second one demonstrates the effectiveness of the global variable. The third one validates the effectiveness of the node cost L_N. Note that we conduct these experiments on the validation set. First, we denote several models as follows:

Methods	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	FM↓
A	52.6	54.8	31.6	32.2	1529	28765	1226	884
A*	52.7	56.3	31.5	33.0	1455	28882	1161	913
M	53.9	61.4	31.9	32.2	1390	28570	690	772

Table 3. Performance of models based on single feature.

A - Appearance graph network with stage C and stage D.
A* - Appearance graph network with all four stages.
A*/g - A* without the global variable.
M - Motion graph network with stage C and stage D.
M/g - M without the global variable.
A*+M - Our method (Cost is calculated by Eqn. (3)).
A*/g+M/g - **A*+M** without the global variable.

(I) Effectiveness of graph networks. Table 3 shows results of three models, including **A**, **A*** and **M**. It is observed that **A*** has a better performance compared with **A**, which indicates that the node updating module in **A*** is likely to catch long term cue to update the appearance feature of objects. To be specific, **A*** improves 1.5 on IDF1 compared with **A**, and has the better FP and IDS than **A**. Hence, we choose **A*** as our appearance graph network.

Moreover, we can see that **M** outperforms both **A*** and A on MOTA, which indicates that the motion graph network is more effective than the appearance graph network. We find that there are some ID switch cases that can be only handled by the appearance graph network, and some can be only handled by the motion graph network. Fig. 5 (a)-(c) shows the effectiveness of graph network. The person with ID=1 is occluded by another person in (a). Due to the similar motion of these persons, it drifts to the woman with white clothes by the motion graph network in (c). However, it can be regarded as a missing object by the appearance graph network in (b). Fig. 5(d)-(f) shows the effectiveness of the motion graph network. Due to the fact that persons with ID=2 and ID=3 have similar clothes in (d), it is difficult to distinguish them by using the appearance graph network in (e). However, it can be handled by the motion graph network in (f). Hence, these two networks are complementary. More results are provided in the supplementary material.

Besides, our current method combines the appearance cue and the motion cue with a weighted strategy, whose costs are calculated by Eqn. (3). The parameter α is set as 0.3 (less than 0.5), which indicates that the motion cue is more important than the appearance cue in our method.

(II) Effectiveness of the global variable. We remove the global variable from **A*** and **M**, and re-train them to get **A*/g** and **M/g** respectively. We also get the model based on the weighted strategy.

Table 4 shows the results of these models. On MOTA, IDF1, MT, FP and IDS, the graph networks with the global variable outperform the graph networks without the global variable no matter what feature they ground. Specifically, on IDF1, **A*** improves 0.5 compared with **A*/g**, **M** im-

Methods	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	FM↓
A*	**52.7**	**56.3**	**31.5**	33.0	**1455**	28882	**1161**	913
A*/g	52.6	55.8	31.2	**32.9**	1545	**28819**	1174	**885**
M	**53.9**	**61.4**	**31.9**	**32.2**	**1390**	**28570**	**690**	**772**
M/g	52.6	60.0	31.6	32.8	1392	28621	1521	802
A*+M	**54.5**	**63.7**	**33.2**	32.3	**1525**	**28210**	**511**	**683**
A*/g+M/g	54.3	62.3	32.9	**32.0**	1622	28247	517	692

Table 4. Performance of models with/without the global variable. The best result is highlighted in bold.

Loss	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	FM↓
$L_C + \lambda L_N$	**52.7**	**56.3**	31.5	**33.0**	**1455**	28882	**1161**	**913**
L_C	52.5	56.0	**32.0**	33.9	1539	**28811**	1253	939

Table 5. Performance of **A*** trained with/without the node cost. The best result is highlighted in bold.

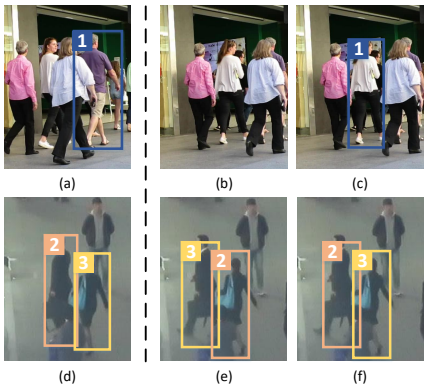

Figure 5. Results of **A*** and **M** in MOT17-09 and MOT17-03 with Faster R-CNN detections. (a) One person at frame 281 in MOT17-09. (b) The tracking results at frame 283 in MOT17-09 using **A***. (c) The tracking results at frame 283 in MOT17-09 using **M**. (d) Two people at frame 1211 in MOT17-03. (e) The tracking results at frame 1217 in MOT17-03 using **A***. (f) The tracking results at frame 1217 in MOT17-03 using **M**.

proves 1.4 compared with **M/g**, and **A*+M** improves 1.4 compared with **A*/g+M/g**. Hence, the global variable in the graph network can catch the global relationship to help tracking, such as, the interaction among objects and the characteristic of the video sequence.

(III) Effectiveness of node cost. We retrain the **A*** without the node cost. The results are shown in Table 5. As we can see, **A*** trained with $L_C + \lambda L_N$ performs better than **A*** trained with L_C, which indicates that the node cost is helpful for tracking. Hence, we train the appearance graph network **A*** with the additional node cost.

5.5. Robustness Study

We conduct experiments over different initializations to assess the robustness of our model. We train our model on five different initializations: default, xavier_uniform, xavier_normal, kaiming_uniform and kaiming_normal. Default initialization uses kaiming_uniform to initialize the weights and the biases of the fully connected layer. The remain four initializations are used to initialize the weights, and the biases are initialized with zeros. Specifically, we train our model with the default initialization to achieve the best performance. Details of different initializations can be found on [1]. We report the standard deviation and the mean of MOTA, IDF1, MT and ML on the validation set over these five initializations.

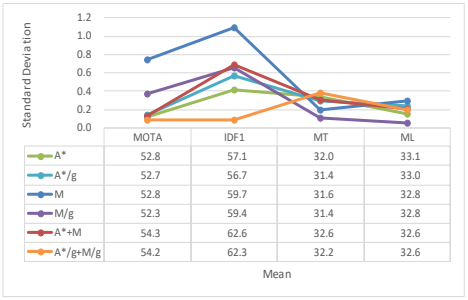

Figure 6. Standard deviation and mean of MOTA, IDF1, MT and ML over five initializations.

As shown in Fig 6, **M** is more sensitive to initialization than other models. Due to the weighted strategy, **A*+M** is more robust than **M**. **A*/g+M/g** is more robust than **A*/g** and **M/g**. Compared with Table 4, the average performance of **A*** and **A*/g** over five initializations is better than the performance of **A*** and **A*/g** trained with default initialization. Since the average performance of **M** over five initializations is worse than the performance of **M** trained with default initialization, the average performance of **A*+M** over five initializations is also worse than the performance of **A*+M** trained with default initialization.

6. Conclusion and Future Works

In this paper, we propose a near-online MOT method with an end-to-end graph network, which can achieve the global optimum in each timestep. Besides, the updating mechanism is included in our graph network. Experimental results show the effectiveness of our method.

In future works, we will focus on extending the graph network for the near-online MOT, which can build a bigger graph among recent frames and more reasoning will be included. Besides, the interaction among objects in the same frame can also be modeled with the graph network.

Acknowledgement

This work was partially supported the Natural Science Foundation of China under contracts 61572042, 61527804. This work was partially supported by Qualcomm. We also acknowledge the high-performance computing platform of Peking University for providing computational resources.

References

[1] https://pytorch.org/docs/stable/nn.init.html.

[2] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 4509–4517, USA, 2016. Curran Associates Inc.

[3] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

[4] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using K-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011.

[5] A. A. Butt and R. T. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1846–1853, 2013.

[6] V. Chari, S. Lacoste-Julien, I. Laptev, and J. Sivic. On pairwise costs for network flow multi-object tracking. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5537–5545, 2015.

[7] J. Chen, H. Sheng, Y. Zhang, and Z. Xiong. Enhancing detection model for multiple hypothesis tracking. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2143–2152, July 2017.

[8] W. Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3029–3037, 2015.

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

[10] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu. Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4846–4855, Oct 2017.

[11] L. Fagot-Bouquet, R. Audigier, Y. Dhome, and F. Lerasle. Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016: 14th European Conference on Computer Vision*, pages 774–790.

[12] K. Fang, Y. Xiang, X. Li, and S. Savarese. Recurrent autoregressive networks for online multi-object tracking. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 466–475, March 2018.

[13] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, Sept 2010.

[14] Z. Fu, F. Angelini, J. Chambers, and S. M. Naqvi. Multi-level cooperative fusion of gm-phd filters for online multiple human tracking. *IEEE Transactions on Multimedia*, pages 1–1, 2019.

[15] Z. Fu, P. Feng, F. Angelini, J. Chambers, and S. M. Naqvi. Particle PHD filter based multiple human tracking using online group-structured dictionary learning. *IEEE Access*, 6:14764–14778, 2018.

[16] X. Gao and T. Jiang. OSMO: Online specific models for occlusion in multiple object tracking under surveillance scene. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, pages 201–210, New York, NY, USA, 2018. ACM.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[18] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele. Motion segmentation multiple object tracking by correlation co-clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1):140–153, Jan 2018.

[19] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg. Multiple hypothesis tracking revisited. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4696–4704, 2015.

[20] C. Kim, F. Li, and J. M. Rehg. Multi-object tracking with neural gating using bilinear LSTM. In *Computer Vision – ECCV 2018: 15th European Conference on Computer Vision*, September 2018.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[22] H. W. Kuhn. Statement for Naval Research Logistics: "The Hungarian method for the assignment problem". *Naval Res. Logist.*, 52(1):6–21, 2005. Reprinted from Naval Res. Logist. Quart. **2** (1955), 83–97.

[23] L. Leal-Taixe, A. Milan, I. Reid, S. Roth, and K. Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. 2015.

[24] S. Lee, M. Kim, and S. Bae. Learning discriminative appearance models for online multi-object tracking with appearance discriminability measures. *IEEE Access*, pages 1–1, 2018.

[25] C. Ma, C. Yang, F. Yang, Y. Zhuang, Z. Zhang, H. Jia, and X. Xie. Trajectory factory: Tracklet cleaving and reconnection by deep siamese Bi-GRU for multiple object tracking. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, July 2018.

[26] L. Ma, S. Tang, M. J. Black, and L. Van Gool. Multi-person tracking with automatic customization. In *The 14th Asian Conference on Computer Vision (ACCV)*, December 2018.

[27] A. Milan, S. Hamid Rezatofighi, A. Dick, I. Reid, and K. Schindler. Online multi-target tracking using recurrent neural networks. In *2017 The Association for the Advance of Artificial Intelligence (AAAI)*, February 2017.

[28] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016.

[29] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1201–1208, 2011.

[30] L. Ren, J. Lu, Z. Wang, Q. Tian, and J. Zhou. Collaborative deep reinforcement learning for multi-object tracking. In *Computer Vision – ECCV 2018: 15th European Conference on Computer Vision*, September 2018.

[31] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, June 2017.

[32] E. Ristani and C. Tomasi. Features for multi-target multi-camera tracking and re-identification. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[33] A. Sadeghian, A. Alahi, and S. Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 300–311, Oct 2017.

[34] R. Sanchez-Matilla, F. Poiesi, and A. Cavallaro. Online multi-target tracking with strong and weak detections. In *Computer Vision – ECCV 2016: 14th European Conference on Computer Vision Workshops*, pages 84–99, 2016.

[35] S. Schulter, P. Vernaza, W. Choi, and M. Chandraker. Deep network flow for multi-object tracking. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2730–2739, July 2017.

[36] Y. Shen, H. Li, S. Yi, D. Chen, and X. Wang. Person re-identification with deep similarity-guided graph neural network. In *Computer Vision – ECCV 2018: 15th European Conference on Computer Vision*, pages 508–526, Cham, 2018. Springer International Publishing.

[37] H. Sheng, J. Chen, Y. Zhang, W. Ke, Z. Xiong, and J. Yu. Iterative multiple hypothesis tracking with tracklet-level association. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2018.

[38] H. Sheng, Y. Zhang, J. Chen, Z. Xiong, and J. Zhang. Heterogeneous association graph fusion for target association in multiple object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2019.

[39] S. Tang, M. Andriluka, B. Andres, and B. Schiele. Multiple people tracking by lifted multicut and person Re-identification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3701–3710, July 2017.

[40] W. Tian, M. Lauer, and L. Chen. Online multi-object tracking using joint domain information in traffic scenarios. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11, 2019.

[41] B. Wang, G. Wang, K. L. Chan, and L. Wang. Tracklet association with online target-specific metric learning. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1234–1241, 2014.

[42] B. Wang, G. Wang, K. L. Chan, and L. Wang. Tracklet association by online target-specific metric learning and coherent dynamics estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(3):589–602, March 2017.

[43] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4705–4713, 2015.

[44] B. Yang and R. Nevatia. An online learned CRF model for multi-target tracking. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2034–2041, 2012.

[45] F. Yang, W. Choi, and Y. Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2129–2137, June 2016.

[46] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan. POI: Multiple object tracking with high performance detection and appearance feature. In G. Hua and H. Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 36–42, Cham, 2016. Springer International Publishing.

[47] A. R. Zamir, A. Dehghan, and M. Shah. GMCP-tracker: Global multi-object tracking using generalized minimum clique graphs. In *Computer Vision – ECCV 2012: 12th European Conference on Computer Vision*, 2012.

[48] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[49] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang. Online multi-object tracking with dual matching attention networks. In *Computer Vision – ECCV 2018: 15th European Conference on Computer Vision*, September 2018.

[50] Z. Zhu, Q. Wang, L. Bo, W. Wu, J. Yan, and W. Hu. Distractor-aware siamese networks for visual object tracking. In *Computer Vision – ECCV 2018: 15th European Conference on Computer Vision*, September 2018.