

Learnable Graph Matching: Incorporating Graph Partitioning with Deep Feature Learning for Multiple Object Tracking

Jiawei He^{1,3} Zehao Huang² Naiyan Wang² Zhaoxiang Zhang^{1,3,4}

¹ Institute of Automation, Chinese Academy of Sciences (CASIA) ² TuSimple

³ School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS)

⁴ Centre for Artificial Intelligence and Robotics, HKISI.CAS

{hejiawei2019, zhaoxiang.zhang}@ia.ac.cn {zhaohuang18, winsty}@gmail.com

Abstract

Data association across frames is at the core of Multiple Object Tracking (MOT) task. This problem is usually solved by a traditional graph-based optimization or directly learned via deep learning. Despite their popularity, we find some points worth studying in current paradigm: 1) Existing methods mostly ignore the context information among tracklets and intra-frame detections, which makes the tracker hard to survive in challenging cases like severe occlusion. 2) The end-to-end association methods solely rely on the data fitting power of deep neural networks, while they hardly utilize the advantage of optimization-based assignment methods. 3) The graph-based optimization methods mostly utilize a separate neural network to extract features, which brings the inconsistency between training and inference. Therefore, in this paper we propose a novel learnable graph matching method to address these issues. Briefly speaking, we model the relationships between tracklets and the intra-frame detections as a general undirected graph. Then the association problem turns into a general graph matching between tracklet graph and detection graph. Furthermore, to make the optimization end-to-end differentiable, we relax the original graph matching into continuous quadratic programming and then incorporate the training of it into a deep graph network with the help of the implicit function theorem. Lastly, our method GMTracker, achieves state-of-the-art performance on several standard MOT datasets. Our code will be available at <https://github.com/jiaweihe1996/GMTracker>.

1. Introduction

Multiple Object Tracking (MOT) is a fundamental task that aims at associating the same object across successive frames in a video clip. A robust and accurate MOT algorithm is indispensable in broad applications, such as au-

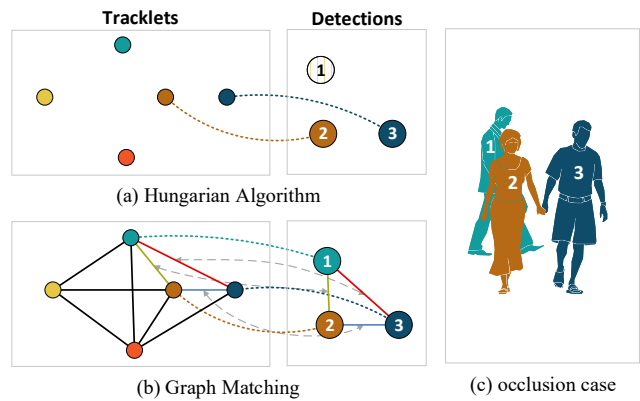


Figure 1: An illustration of intra-graph relationship used in our graph matching formulation. We utilize the second-order edge-to-edge similarity to model the group activity, which is more robust under heavy occlusion. Note that not all vertices in tracklet graph can be matched to the detection graph because they disappear in the current frame.

tonomous driving and video surveillance. The *tracking-by-detection* is currently the dominant paradigm in MOT. This paradigm consists of two steps: (1) obtaining the bounding boxes of objects by detection frame by frame; (2) generating trajectories by associating the same objects between frames. With the rapid development of deep learning based object detectors, the first step is largely solved by the powerful detectors such as [37, 38]. As for the second one, recent MOT works focus on improving the performance of data association mainly from the following two aspects: (1) formulating the association problem as a combinatorial graph partitioning problem and solve it by advanced optimization techniques [5, 7, 62, 8, 65, 18]; (2) improving the appearance models by the power of deep learning [28, 67, 31, 64]. Although very recently, there are works [77, 53, 8, 65] trying to unify feature learning and data association into an end-to-end trained neural network, these two directions are

almost isolated so that these recent attempts hardly utilize the progress from the combinatorial graph partitioning.

In the graph view of MOT, each vertex represents a detection bounding box or a tracklet, while the edges are constructed between the vertices across different frames to represent the similarities between them. Then the association problem can be formulated as a min-cost flow problem [72]. The most popular used method is to construct a bipartite graph between two frames and adopt Hungarian algorithm [27] to solve it. This online strategy is widely used in practice because of its simplicity [7, 64]. Nevertheless, these methods are not robust to occlusions due to the lack of history trajectories and long term memory. This problem is traditionally solved by constructing graph from multiple frames, and then deriving the best association based on the optimal solution of this min-cost flow problem [72].

All these existing works focus on finding the best matching across frames, but ignoring the context within the frame. In this paper, we argue that the relationship between the vertices within the same frame is also crucial for some challenging cases in MOT. For example, we can match an occluded object to the correct tracklet solely by the past relationships with neighborhood objects. Fig. 1 just shows such an example. Interestingly, these pairwise relationships within the same frame can be represented as edges in a general graph. To this end, the popular bipartite matching across frames can be updated to general graph matching between them. To further integrate this novel assignment formulation with powerful feature learning, we first relax the original formulation of graph matching [30, 26] to a quadratic programming, and then derive a differentiable QP layer based on the KKT conditions and the implicit function theorem for the graph matching problem, inspired by the OptNet [2]. Finally, the assignment problem can be learned in synergy with the features.

Overall, our work has the following contributions:

- Instead of only focusing on the association across frames, we emphasize the importance of intra-frame relationships. Particularly, we propose to represent the relationships as a general graph, and formulate the association problem as general graph matching.
- To solve this challenging assignment problem, and further incorporate it with deep feature learning, we derive a differentiable quadratic programming layer based on the continuous relaxation of the problem, and utilize implicit function theorem and KKT conditions to derive the gradient w.r.t the input features during back-propagation.
- We evaluate our proposed GMTracker on the large scale open benchmark. Our method could remarkably advance the state-of-the-art performance in terms of association metric such as IDF1.

2. Related Work

Data association in MOT. The data association step in *tracking-by-detection* paradigm is generally solved by probabilistic filter or combinatorial optimization techniques. Classical probabilistic approach includes JPDA [3] and MHT [48]. The advantage of this approach is to keep all the possible candidates for association, and remain the chance to recover from failures. Nevertheless, their costs are prohibitive if no approximation is applied [14, 24]. For combinatorial optimization, traditional approach include bipartite matching [7], dynamic programming [13], min-cost flow [72, 5] and conditional random field [66]. Follow-up works tried to adopt more complex optimization methods [70, 55], reduce the computational cost [47, 56] or promote an online setting from them [9, 59].

Deep learning in MOT. Early works of deep learning in MOT such as [64, 31, 51, 34] mostly focus on learning a better appearance model for each object. Then by the advance of object detection and multi-task learning, several works [6, 40, 76, 73] combine detection and tracking in the same framework. More recently, several works tried to bridge the graph optimization and end-to-end deep learning [21, 8, 65, 34, 18]. [21] adopts Graph Neural Network (GNN) to learn an affinity matrix in a data-driven way. MPNTrack [8] introduces a message passing network to learn high-order information between vertices from different frames. [34] constructs two graph networks to model appearance and motion features, respectively. Lift [18] proposes a lifted disjoint path formulation for MOT, which introduces lifted edges to capture long term temporal interactions.

Neighborhood and context information in MOT. Pedestrians usually walk in a group, so the motion of them are highly clustered. Modeling neighborhood and context relationships may provide important clues for the MOT task. Several early works [50, 19] consider the group model as a prior in motion model for the crowd scenes. [68] considers the distance between detections as the neighborhood relationship during the data association. However, these hand-crafted priors can be quite limited in complicated scenes. Recently, many methods [36, 46, 39, 29] learn appearance and geometric information by passing messages among neighbors. However, their goal is still to enhance the appearance features. They do not consider them explicitly in the association across frames. In this work, we propose to explicitly consider the neighborhood context in data association via differentiable graph matching and use it to guide the feature learning in an end-to-end manner.

Graph matching and Combinatorial Optimization. Pairwise graph matching, or more generally Quadratic Assignment Problem (QAP), has wide applications in various computer vision tasks [58]. Compared with the linear assignment problem that only considers vertex-to-vertex relation-

ship, pairwise graph matching also considers the second-order edge-to-edge relationship in graphs. The second-order relationship makes matching more robust. However, as shown in [15], this problem is an NP-hard problem. There is no polynomial solver like Hungarian algorithm [27] for the linear assignment problem. In the past decades, many works focus on making the problem tractable by relaxing the original QAP problem [32, 52, 57]. Lagrangian decomposition [54] and factorized graph matching [75] are two representative ones.

In MOT task, the application of graph matching is very limited. To the best of our knowledge, [20] is the first to formulate the MOT task as a graph matching problem and use dual L1-normalized tensor power iteration method to solve it. Different from [20] that directly extracts the features from an off-the-shelf neural network, we propose to guide the feature learning by the optimization problem, which can both enjoy the power of deep feature learning and combinatorial optimization. This joint training manner of representation and optimization problem also eliminate the inconsistencies between the training and inference.

To incorporate graph matching into deep learning, one stream of work is to treat the assignment problem as a supervised learning problem directly, and use the data fitting power of deep learning to learn the projection from input graphs to output assignment directly [61, 69]. Another more theoretically rigorous is to relax the problem to a convex optimization problem first, and then utilize the KKT condition and implicit function theorem to derive the gradients w.r.t all variables at the optimal solution [4]. As shown in [2], the universality and transferability of the latter approach are much better than the first one. Thus, in this paper, we derive a graph matching layer based on this spirit to solve the challenging graph matching problem in MOT.

3. Graph Matching Formulation for MOT

In this section, we will formulate the multiple object tracking problem as a graph matching problem. Instead of solving the original Quadratic Assignment Problem (QAP), we relax the graph matching formulation as a convex quadratic programming (QP) and extend the formulation from the edge weights to the edge features. The relaxation facilitates the differentiable and joint learning of feature representation and combinatorial optimization.

3.1. Detection and Tracklet Graphs Construction

As an online tracker, we track objects frame by frame. In frame t , we define $\mathcal{D}^t = \{D_1^t, D_2^t, \dots, D_{n_d}^t\}$ as the set of detections in current frame and $\mathcal{T}^t = \{T_1^t, T_2^t, \dots, T_{n_t}^t\}$ as the set of tracklets obtained from past frames. n_d and n_t denote the number of detected objects and tracklet candidates. A detection is represented by a triple $D_p^t = (\mathbf{I}_p^t, \mathbf{g}_p^t, t)$, where \mathbf{I}_p^t contains the image pixels in the detected area,

$\mathbf{g}_p^t = (x_p^t, y_p^t, w_p^t, h_p^t)$ is a geometric vector including the central location and size of the detection bounding box. Each tracklet contains a series of detected objects with the same tracklet id. With a bit abuse of notations, the generation of T_{id}^t can be represented as $T_{id}^t \leftarrow T_{id}^{t-1} \cup \{D_{(id)}^{t-1}\}$, which means we add $D_{(id)}^{t-1}$ to the tracklet T_{id}^{t-1} .

Then we define the detection graph in frame t as $\mathcal{G}_D^t = (\mathcal{V}_D^t, \mathcal{E}_D^t)$ and the tracklet graph up to the frame t as $\mathcal{G}_T^t = (\mathcal{V}_T^t, \mathcal{E}_T^t)$. Each vertex $i \in \mathcal{V}_D^t$ and vertex $j \in \mathcal{V}_T^t$ represents the detection D_i^t and the tracklet T_j^t , respectively. The $e_u = (i, i')$ is the edge in \mathcal{E}_D^t and $e_v = (j, j')$ is the edge in \mathcal{E}_T^t . Both of these two graphs are complete graphs. Then the data association in frame t can be formulated as a graph matching problem between \mathcal{G}_D^t and \mathcal{G}_T^t . For simplicity, we will ignore t in the following sections.

3.2. Basic Formulation of Graph Matching

Given the detection graph \mathcal{G}_D and the tracklet graph \mathcal{G}_T , the graph matching problem is to maximize the similarities between the matched vertices and corresponding edges connected by these vertices. In the following derivation, we use the general notation \mathcal{G}_1 and \mathcal{G}_2 to obtain a general graph matching formulation.

As defined in [30], the graph matching problem is a Quadratic Assignment Problem (QAP). A practical mathematical form is named *Koopmans-Beckmann's* QAP [26]:

$$\begin{aligned} \underset{\mathbf{\Pi}}{\text{maximize}} \quad & \mathcal{J}(\mathbf{\Pi}) = \text{tr}(\mathbf{A}_1 \mathbf{\Pi} \mathbf{A}_2 \mathbf{\Pi}^\top) + \text{tr}(\mathbf{B}^\top \mathbf{\Pi}), \\ \text{s.t.} \quad & \mathbf{\Pi} \mathbf{1}_n = \mathbf{1}_n, \mathbf{\Pi}^\top \mathbf{1}_n = \mathbf{1}_n, \end{aligned} \quad (1)$$

where $\mathbf{\Pi} \in \{0, 1\}^{n \times n}$ is a permutation matrix that denotes the matching between the vertices of two graphs, $\mathbf{A}_1 \in \mathbb{R}^{n \times n}$, $\mathbf{A}_2 \in \mathbb{R}^{n \times n}$ are the weighted adjacency matrices of graph \mathcal{G}_1 and \mathcal{G}_2 respectively, and $\mathbf{B} \in \mathbb{R}^{n \times n}$ is the vertex affinity matrix between \mathcal{G}_1 and \mathcal{G}_2 . $\mathbf{1}_n$ denotes an n -dimensional vector with all values to be 1.

3.3. Reformulation and Convex Relaxation

For *Koopmans-Beckmann's* QAP, as $\mathbf{\Pi}$ is a permutation matrix, i.e., $\mathbf{\Pi}^\top \mathbf{\Pi} = \mathbf{\Pi} \mathbf{\Pi}^\top = \mathbf{I}$. Following [75], Eq. 1 can be rewritten as

$$\mathbf{\Pi}^* = \arg \min_{\mathbf{\Pi}} \frac{1}{2} \|\mathbf{A}_1 \mathbf{\Pi} - \mathbf{\Pi} \mathbf{A}_2\|_F^2 - \text{tr}(\mathbf{B}^\top \mathbf{\Pi}). \quad (2)$$

This formulation is more intuitive than that in Eq. 1. For two vertices $i, i' \in \mathcal{G}_1$ and their corresponding vertices $j, j' \in \mathcal{G}_2$, the first term in Eq. 2 denotes the difference of the weight of edge (i, i') and (j, j') , and the second term denotes the vertex affinities between i and j . Then the goal of the optimization is to maximize the vertex affinities between all matched vertices, and minimize the difference of edge weights between all matched edges.

It can be proven that the convex hull of the permutation matrix lies in the space of the doubly-stochastic matrix. So, as shown in [1], the QAP (Eq. 2) can be relaxed to its tightest convex relaxation by only constraining the permutation matrix Π to be a double stochastic matrix \mathbf{X} , formed as the following QP problem:

$$\mathbf{X}^* = \arg \min_{\mathbf{X} \in \mathcal{D}} \frac{1}{2} \|\mathbf{A}_1 \mathbf{X} - \mathbf{X} \mathbf{A}_2\|_F^2 - \text{tr}(\mathbf{B}^\top \mathbf{X}), \quad (3)$$

where $\mathcal{D} = \{\mathbf{X} : \mathbf{X} \mathbf{1}_n = \mathbf{1}_n, \mathbf{X}^\top \mathbf{1}_n = \mathbf{1}_n, \mathbf{X} \geq 0\}$.

3.4. From Edge Weights to Edge Features

In the formulation of graph matching above, the element $a_{i,i'}$ in the weighted adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a scalar denoting the weight on the edge (i, i') . To facilitate the application in our MOT problem, we expand the relaxed QP formulation by using an l_2 -normalized edge feature $\mathbf{h}_{i,i'} \in \mathbb{R}^d$ instead of the scalar-formed edge weight $a_{i,i'}$ in \mathbf{A} . We build a weighted adjacency tensor $\mathbf{H} \in \mathbb{R}^{d \times n \times n}$ where $\mathbf{H}^{i,i'} = \mathbf{h}_{i,i'}$, i.e., we consider the each dimension of $\mathbf{h}_{i,i'}$ as the element $a_{i,i'}$ in \mathbf{A} and concatenate them along channel dimension. The \mathbf{H}_1 and \mathbf{H}_2 are the weighted adjacency tensors for \mathcal{G}_1 and \mathcal{G}_2 , respectively. Then the optimization objective in Eq. 2 can be further expanded to consider the l_2 distance between two corresponding n - d edge features other than the scalar differences:

$$\begin{aligned} \Pi^* &= \arg \min_{\Pi} \sum_{c=1}^d \frac{1}{2} \|\mathbf{H}_1^c \Pi - \Pi \mathbf{H}_2^c\|_F^2 - \text{tr}(\mathbf{B}^\top \Pi) \\ &= \arg \min_{\Pi} \sum_{i=1}^n \sum_{i'=1}^n \sum_{j=1}^n \sum_{j'=1}^n \frac{1}{2} \|\mathbf{h}_{ii'} \pi_{ij} - \mathbf{h}_{jj'} \pi_{i'j'}\|_2^2 \\ &\quad - \text{tr}(\mathbf{B}^\top \Pi) \\ &= \arg \min_{\Pi} \sum_{i=1}^n \sum_{i'=1}^n \sum_{j=1}^n \sum_{j'=1}^n \frac{1}{2} (\pi_{ij}^2 - 2\pi_{ij}\pi_{i'j'} \mathbf{h}_{ii'}^\top \mathbf{h}_{jj'}) \\ &\quad + \pi_{i'j'}^2 - \text{tr}(\mathbf{B}^\top \Pi), \end{aligned} \quad (4)$$

where n is the number of vertices in graph \mathcal{G}_1 and \mathcal{G}_2 , the subscript i and i' are the vertices in graph \mathcal{G}_1 and j and j' are in graph \mathcal{G}_2 . We reformulate Eq. 4 as:

$$\pi^* = \arg \min_{\pi} \pi^\top ((n-1)^2 \mathbf{I} - \mathbf{M}) \pi - \mathbf{b}^\top \pi, \quad (5)$$

where $\pi = \text{vec}(\Pi)$, $\mathbf{b} = \text{vec}(\mathbf{B})$ and $\mathbf{M} \in \mathbb{R}^{n^2 \times n^2}$ is the symmetric quadratic affinity matrix between all the possible edges in two graphs.

Following the relaxation in Section 3.3, the formulation Eq. 5 using edge features can be relaxed to a QP:

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{D}'} \mathbf{x}^\top ((n-1)^2 \mathbf{I} - \mathbf{M}) \mathbf{x} - \mathbf{b}^\top \mathbf{x}, \quad (6)$$

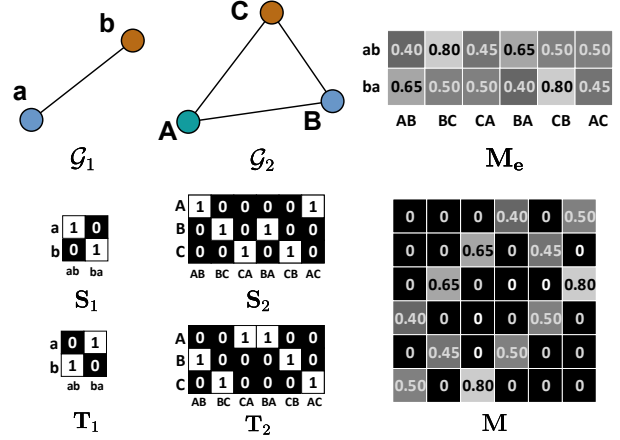


Figure 2: An example of the derivation from edge affinity matrix \mathbf{M}_e to quadratic affinity matrix \mathbf{M} .

where $\mathcal{D}' = \{\mathbf{x} : \mathbf{R}\mathbf{x} = \mathbf{1}, \mathbf{U}\mathbf{x} \leq \mathbf{1}, \mathbf{x} \geq 0, \mathbf{R} = \mathbf{1}_{n_2}^\top \otimes \mathbf{I}_{n_1}, \mathbf{U} = \mathbf{I}_{n_2}^\top \otimes \mathbf{1}_{n_1}\}$, \otimes denotes Kronecker product.

In the implementation, we first compute the cosine similarity between the edges in \mathcal{G}_D and \mathcal{G}_T to construct the matrix $\mathbf{M}_e \in \mathbb{R}^{|\mathcal{E}_D| \times |\mathcal{E}_T|}$. The element of the matrix \mathbf{M}_e is the cosine similarity between edge features $\mathbf{h}_{i,i'}$ and $\mathbf{h}_{j,j'}$ in two graphs:

$$\mathbf{M}_e^{u,v} = \mathbf{h}_{i,i'}^\top \mathbf{h}_{j,j'}, \quad (7)$$

where $e_u = (i, i')$ is the edge in \mathcal{G}_D and $e_v = (j, j')$ is the edge in \mathcal{G}_T .

And following [71], we map each element of matrix \mathbf{M}_e to the symmetric quadratic affinity matrix \mathbf{M} :

$$\mathbf{M} = (\mathbf{S}_D \otimes \mathbf{S}_T) \text{diag}(\text{vec}(\mathbf{M}_e)) (\mathbf{T}_D \otimes \mathbf{T}_T)^\top, \quad (8)$$

where $\text{diag}(\cdot)$ means constructing a diagonal matrix by the given vector, $\mathbf{S}_D \in \{0, 1\}^{|\mathcal{V}_D| \times |\mathcal{E}_D|}$ and $\mathbf{S}_T \in \{0, 1\}^{|\mathcal{V}_T| \times |\mathcal{E}_T|}$, whose elements are an indicator function:

$$\mathbb{I}_s(i, u) := \begin{cases} 1 & \text{if } i \text{ is the start vertex of edge } e_u, \\ 0 & \text{if } i \text{ is not the start vertex of edge } e_u, \end{cases} \quad (9)$$

$\mathbf{T}_D \in \{0, 1\}^{|\mathcal{V}_D| \times |\mathcal{E}_D|}$ and $\mathbf{T}_T \in \{0, 1\}^{|\mathcal{V}_T| \times |\mathcal{E}_T|}$, whose elements are another indicator function:

$$\mathbb{I}_t(i', u) := \begin{cases} 1 & \text{if } i' \text{ is the end vertex of edge } e_u, \\ 0 & \text{if } i' \text{ is not the end vertex of edge } e_u. \end{cases} \quad (10)$$

An example of the derivation from \mathbf{M}_e to \mathbf{M} is illustrated in Fig. 2.

Besides, each element in the vertex affinity matrix \mathbf{B} is the cosine similarities between feature \mathbf{h}_i on vertex $i \in \mathcal{V}_D$ and feature \mathbf{h}_j on vertex $j \in \mathcal{V}_T$:

$$\mathbf{B}_{i,j} = \mathbf{h}_i^\top \mathbf{h}_j \quad (11)$$

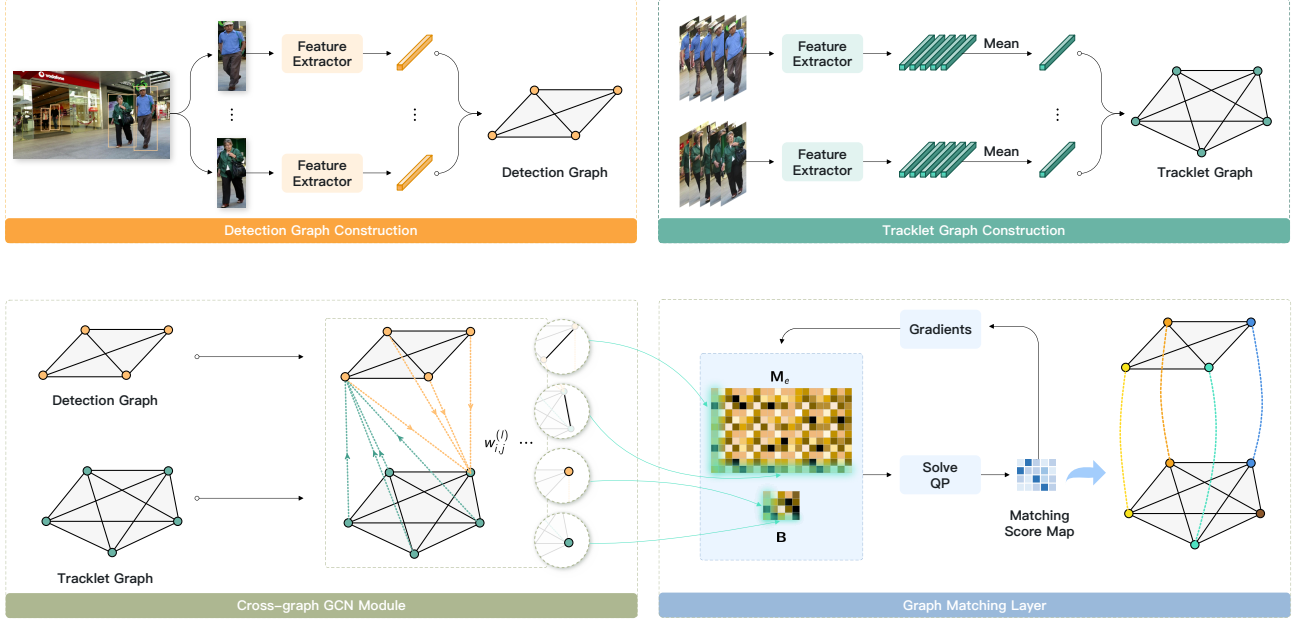


Figure 3: Overview of our method. We first extract features from detections and construct the detection graph using these features. The tracklet graph construction step is similar to the detection graph, but we average the features in a tracklet. Then the cross-graph GCN is adopted to enhance the features. The weight $w_{i,j}$ is from the feature similarity and geometric information. The core of our method is the differentiable graph matching layer built as a QP layer from the formulation in Eq. 6. The M_e and B in the graph matching layer denote the edge affinity matrix from Eq. 7 and the vertex affinity matrix from Eq. 11 respectively.

4. Graph Matching Network and GMTracker

In this section, we will describe the details of our Graph Matching Network and our GMTracker. As shown in Fig. 3, the pipeline of our Graph Matching Network consists of three parts: (1) feature encoding in detection and tracklet graphs; (2) feature enhancement by cross-graph Graph Convolutional Network (GCN) and (3) differentiable graph matching layer. We will describe these three parts step by step and show how we integrate them into a tracker (GMTracker) in the following.

4.1. Feature Encoding in Two Graphs

We utilize a pre-trained ReIdentification (ReID) network followed by a multi-layer perceptron (MLP) to generate the appearance feature \mathbf{a}_D^i for each detection D_i . The appearance feature \mathbf{a}_T^j of the tracklet T_j is obtained by averaging all the appearance features of detections before.

4.2. Cross-Graph GCN

Similar to [8, 42, 63], we only adopt a GCN module between the graph \mathcal{G}_D and graph \mathcal{G}_T to enhance the feature, and thus it is called Cross-Graph GCN.

The initial vertex features on detection graph and tracklet graph are the appearance features on the vertices, i.e., let $\mathbf{h}_i^{(0)} = \mathbf{a}_D^i$ and $\mathbf{h}_j^{(0)} = \mathbf{a}_T^j$. Let $\mathbf{h}_i^{(l)}$ and $\mathbf{h}_j^{(l)}$ be the feature

of vertex $i \in \mathcal{G}_D$ and vertex $j \in \mathcal{G}_T$ in the l -th propagation, respectively. We define the aggregation weight coefficient $w_{i,j}^{(l)}$ in GCN as the appearance and geometric similarity between vertex i and vertex j :

$$w_{i,j}^{(l)} = \cos(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}) + \text{IoU}(\mathbf{g}_i, \mathbf{g}_j) \quad (12)$$

where $\cos(\cdot, \cdot)$ means the cosine similarity between input features and $\text{IoU}(\cdot, \cdot)$ denotes the Intersection over Union of two bounding boxes. For a detection vertex i , \mathbf{g}_i is the corresponding detection bounding box defined in Section 3.1. As for a tracklet vertex j , we estimate the bounding box \mathbf{g}_j in current frame t by Kalman Filter [22] motion model with a constant velocity. Note that we only consider the appearance feature similarity in weight $w_{i,j}$ when the camera moves, since the motion model cannot predict reliable future positions in these complicated scenes.

We use summation as the aggregation function, i.e., $\mathbf{m}_i^{(l)} = \sum_{j \in \mathcal{G}_T} w_{i,j}^{(l)} \mathbf{h}_j^{(l)}$ and the vertex features are updated by:

$$\mathbf{h}_i^{(l+1)} = \text{MLP}\left(\mathbf{h}_i^{(l)} + \frac{\|\mathbf{h}_i^{(l)}\|_2 \mathbf{m}_i^{(l)}}{\|\mathbf{m}_i^{(l)}\|_2}\right), \quad (13)$$

where we adopt message normalization proposed in [33] to stabilize the training.

We apply l_2 normalization to the final features after cross-graph GCN and denote it as \mathbf{h}_i . Then we use \mathbf{h}_i as

the feature of vertex i in graph \mathcal{G}_D , and construct the edge feature for edge (i, i') with $\mathbf{h}_{i,i'} = l_2([\mathbf{h}_i, \mathbf{h}_{i'}])$, where $[\cdot]$ denotes concatenation operation. The similar operation is also applied to the tracklet graph \mathcal{G}_T . In our implementation, we only apply GCN once.

4.3. Differentiable Graph Matching Layer

After enhancing the vertex features and constructing the edge features on graph \mathcal{G}_D and \mathcal{G}_T , we meet the core component of our method: the differentiable graph matching layer. By optimizing the QP in Eq. 6 from quadratic affinity matrix \mathbf{M} and vertex affinity matrix \mathbf{B} , we can derive the optimal matching score vector \mathbf{x} and reshape it back to the shape $n_d \times n_t$ to get the matching score map \mathbf{X} .

Since we finally formulate the graph matching problem as a QP, we can construct the graph matching module as a differentiable QP layer in our neural network. Since KKT conditions are the necessary and sufficient conditions for the optimal solution \mathbf{x}^* and its dual variables, we could derive the gradient in backward pass of our graph matching layer based on the KKT conditions and implicit function theorem, which is inspired by OptNet [2]. Please refer to the appendix for the detailed derivation of the gradients in graph matching layer. In our implementation, we adopt the qpth library [2] to build the graph matching module. In the inference stage, to reduce the computational cost and accelerate the algorithm, we solve the QP using the CVXPY library [11] only for forward operation.

For training, we use weighted binary cross entropy Loss:

$$\mathcal{L} = \frac{-1}{n_d n_t} \sum_{i=1}^{n_d} \sum_{j=1}^{n_t} k y_{i,j} \log(\hat{y}_{i,j}) + (1 - y_{i,j}) \log(1 - \hat{y}_{i,j}), \quad (14)$$

where $\hat{y}_{i,j}$ denotes the matching score between detection D_i and tracklet T_j , and $y_{i,j}$ is the ground truth indicating whether the object belongs to the tracklet. $k = (n_t - 1)$ is the weight to balance the loss between positive and negative samples. Besides, due to our QP formulation of graph matching, the distribution of matching score map \mathbf{X} is relatively smooth. We adopt softmax function with temperature τ to sharpen the distribution of scores before calculating the loss:

$$\hat{y}_{i,j} = \text{Softmax}(x_{i,j}, \tau) = \frac{e^{x_{i,j}/\tau}}{\sum_{j=1}^{n_t} e^{x_{i,j}/\tau}}, \quad (15)$$

where $x_{i,j}$ is the original matching score in score map \mathbf{X} .

4.4. Inference Details

Due to the continuous relaxation, the output of the QP layer may not be binary. To get a valid assignment, we use the greedy rounding strategy to generate the final permutation matrix from the predicted matching score map, i.e., we match the detection with the tracklet with the maximum

score. After matching, like DeepSORT [64], we need to handle the born and death of tracklets. We filter out the detection if it meets one of the three criteria: 1) All the appearance similarities between a detection and existing tracklets are below a threshold σ . 2) It is far away from all tracklets. We set a threshold κ as the Mahalanobis distance between the predicted distribution of the tracklet bounding box by the motion model and the detection bounding box in pixels, called motion gate. 3) The detection bounding box has no overlap with any tracklets. Here, besides the Kalman Filter adopted to estimate the geometric information in Section 4.2, we apply an Enhanced Correlation Coefficient (ECC) [12] in our motion model additionally to compensate the camera motion. Besides, we apply the IoU association between the filtered detections and the unmatched tracklets by Hungarian algorithm to compensate some incorrect filtering. Then the remaining detections are considered as a new tracklet. We delete a tracklet if it has not been updated since δ frames ago, called *max age*.

5. Experiments

5.1. Datasets

We carry out all experiments on MOT16 [43] and MOT17 [43] benchmark. The videos in this benchmark were taken under various scenes, light conditions and frame rates. Occlusion, motion blur, camera motion and distant pedestrians are also crucial problems in this benchmark. Among all the evaluation metrics, Multiple Object Tracking Accuracy (MOTA) [23] and ID F1 Score (IDF1) [49] are the most general metrics in the MOT task. Since MOTA is mostly dominated by the detection metrics false positive and false negative, and our graphing matching method mainly tries to tackle the associations between detected objects, we pay more attention to IDF1 than the MOTA metric.

5.2. Implementation Details

Training. Following other MOT methods [8, 18], we adopt Tracktor [6] to refine the public detections. For the ReID network used for feature extraction, we use a ResNet50 [16] backbone followed by a global average pooling layer and a fully connected layer with 512 channels. We further normalize the output feature with the l_2 normalization. We pre-train the ReID network on Market1501 [74], DukeMTMC [49] and CUHK03 [35] datasets jointly, following the setting of [8]. The parameters of the ReID network will be frozen after pre-training. Then we add two trainable fully connected layers with 512 channels to get appearance features. Our implementation is based on PyTorch [45] framework. We train our model on an NVIDIA RTX 2080Ti GPU. Adam [25] optimizer is applied with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is 5×10^{-5} and weight decay is 10^{-5} . The temperature τ in Eq. 15 is

GM	App. Enc.	GCN	Geo	Inter.	IDF1 \uparrow	MOTA \uparrow	MT \uparrow	ML \downarrow	FP \downarrow	FN \downarrow	ID Sw. \downarrow
					68.1	62.1	556	371	1923	124480	1135
✓					70.0	62.3	555	374	1735	124292	1128
✓			✓		70.2	62.2	555	374	1744	124301	1140
✓	✓				70.4	62.3	554	375	1741	124298	1058
✓	✓	✓			70.6	62.2	556	374	1748	124305	1399
✓	✓	✓	✓		71.5	62.3	555	375	1741	124298	1017
				✓	68.9	62.9	678	361	11440	112853	723
✓				✓	71.6	64.0	669	365	7095	113392	659
✓			✓	✓	71.7	64.0	666	364	6816	113778	724
✓	✓			✓	72.0	64.2	671	368	7701	112370	627
✓	✓	✓		✓	72.1	63.3	676	364	10888	111869	716
✓	✓	✓	✓	✓	73.0	63.8	672	361	9579	111683	570

Table 1: Ablation studies on different proposed components on MOT17 *val* set.

Train w/ GM	Inference w/ GM	IDF1	MOTA
		69.5	62.1
	✓	70.2	62.3
✓	✓	71.5	62.3

Table 2: Ablation study on the graph matching layer.

10^{-3} .

Inference. Our inference pipeline mostly follows DeepSORT [64], except that we use general graphing matching instead of bipartite matching for association. As in DeepSORT, we set the motion gate κ as 9.4877, which is at the 0.95 confidence of the inverse χ^2 distribution. The feature similarity threshold σ is set to 0.6 in the videos taken by the moving camera, and 0.7 when we use geometric information in the cross-graph GCN module for videos taken by the static camera. The *max age* δ is 100 frames.

Post-Processing. To compare with other state-of-the-art offline methods, we perform a linear interpolation within the tracklet as post-processing to compensate the missing detections, following [8, 18]. This effectively reduces the false negatives introduced by upstream object detection algorithm.

5.3. Ablation Study

We conduct ablation studies of the proposed components in our method on the MOT17 dataset. Following [8], we divide the training set into three parts for three-fold cross-validation, called MOT17 *val* set, and we conduct the experiments under this setting both in the ablation study section and the discussions section. We ablate each component we propose: (i) graph matching module built as a QP layer (GM); (ii) MLP trained on MOT dataset to refine the appearance features (App. Enc.); (iii) the cross-graph GCN module (GCN) with and without using geometric informa-

tion (Geo); (iv) the linear interpolation method between the same object by the time (Inter.).

As shown in Table 1, compared with the DeepSORT baseline (the first row), which associates the detections and the tracklets by Hungarian Algorithm, the graph matching method gets a gain of 1.9 IDF1 without interpolation, and a gain of 2.7 IDF1 and 1.1 MOTA with the linear interpolation. The results show the effectiveness of the second-order edge-to-edge information.

Appearance feature refinement and GCN improve about 0.6 IDF1 compared to the untrained model. Geometric information provides about 1.0 additional gain on IDF1, which highlights the importance of geometric information in the MOT task. Finally, compared with the baseline, our method achieves about 3.4 and 0.2 improvements on IDF1 metric and MOTA metric, respectively. With interpolation, the gain becomes even larger: about 4.1 improvements on IDF1 and 0.9 on MOTA.

As shown in Table 2, we get the gain of 1.3 and 2.0 IDF1 compared with only removing the graph matching layer in training stage and in both training and inference stage, respectively. The results demonstrate the effectiveness of our differentiable graph matching layer and the importance of training all components in our tracker jointly.

5.4. Discussions

In this part, we discuss two main design choices of our method on MOT17 *val* set. When we construct the tracklet graph, there are some different intra-tracklet feature aggregation methods. Moreover, how to create and delete a tracklet is important for an online tracker.

Intra-tracklet feature aggregation. In the tracklet graph \mathcal{G}_T , each vertex represents a tracklet. And the vertex feature \mathbf{a}_T^j is the aggregation of the appearance features of all detections in tracklet T_j . Here, we compare several aggregation methods, including mean, moving average and only

Methods	IDF1	MOTA
Last Frame	69.1	62.3
Moving Average $\alpha = 0.5$	69.9	62.3
Moving Average $\alpha = 0.8$	70.1	62.3
Mean	70.0	62.3

Table 3: Ablation studies on different intra-tracklet feature aggregation methods.

Max age (frames)	30	50	80	100	150
Hungarian Algorithm	67.2	67.9	68.1	68.1	67.3
Graph Matching	68.0	69.3	69.7	70.0	70.0

Table 4: The influence of $max\ age\ \delta$ on IDF1 .

using the last frame of the tracklet. The results are shown in Table 3. The IDF1 is 0.9 lower when only using the last frame of the tracklet. The results also reveal that when we utilize all the frame information, no matter using the simple average or the moving average, their impact is not significant. To make our method simple and effective, we finally use the simple average method to aggregate the appearance features within a tracklet.

Tracklet death strategies. As for removing a trajectory from association candidates, our basic strategy is that if the tracklet has not been associated with any detections in δ frames, the tracklet will be removed and not be matched any more. Table 4 shows that larger $max\ age\ \delta$, which means more tracklet candidates, yields better IDF1 score. It shows the effectiveness of our method from another aspect that our GMTracker can successfully match the tracklets disappeared about five seconds ago. On the contrary, when the $max\ age$ increases to 150 frames, the IDF1 will drop 0.8 using Hungarian algorithm, which indicates our graph matching can deal with long-term tracklet associations better.

5.5. Comparison with State-of-the-Art Methods

We compare our GMTracker with other state-of-the-art methods on MOT16 and MOT17 test sets. As shown in Table 5, when we apply Tracktor [6] to refine the public detection, the *online* GMTracker achieves 63.8 IDF1 on MOT17 and 63.9 IDF1 on MOT16, outperforming the other online trackers. To compare with CenterTrack [76], we use the same detections, called GMT_CT, and the IDF1 is 66.9 on MOT17 and 68.6 on MOT16. With the simple linear interpolation, called GMT_simInt in Table 5, we also outperform the other *offline* state-of-the-art trackers on IDF1. With exactly the same visual inter- and extrapolation as Lift [18], called GMT_VIVE in Table 5, the MOTA is comparable with Lift. After utilizing the CenterTrack detections and linear interpolation, the GMTCT_simInt improves the SOTA on both MOT16 and MOT17 datasets. In appendix, we report more detailed performance on other

Methods	Refined Det	IDF1 \uparrow	MOTA \uparrow	FP \downarrow	FN \downarrow	IDS \downarrow
MOT17						
GNMOT (O*) [34]	-	47.0	50.2	29316	246200	5273
FAMNet (O) [10]	-	48.7	52.0	14138	253616	3072
JBNOT (O*) [17]	-	50.8	52.6	31572	232659	3050
Tracktor++ (O) [6]	Tracktor	52.3	53.5	12201	248047	2072
Tracktor++v2 (O) [6]	Tracktor	55.1	56.3	8866	235449	1987
GNNMatch (O) [44]	Tracktor	56.1	57.0	12283	228242	1957
GSM_Tracktor (O) [39]	Tracktor	57.8	56.4	14379	230174	1485
CTTrackPub (O) [76]	CenterTrack	59.6	61.5	14076	200672	2583
GMTracker(Ours) (O)	Tracktor	63.8	56.2	8719	236541	1778
GMT_CT(Ours) (O)	CenterTrack	66.9	61.5	14059	206655	2415
TPM [46]	-	52.6	54.2	13739	242730	1824
eTC17 [60]	-	58.1	51.9	36164	232783	2288
MPNTrack [8]	Tracktor	61.7	58.8	17413	213594	1185
Lif_TsimInt [18]	Tracktor	65.2	58.2	16850	217944	1022
LifT [18]	Tracktor	65.6	60.5	14966	206619	1189
GMT_simInt (Ours)	Tracktor	65.9	59.0	20395	209553	1105
GMT_VIVE (Ours)	Tracktor	65.9	60.2	13142	209812	1675
GMTCT_simInt (Ours)	CenterTrack	68.7	65.0	18213	177058	2200
MOT16						
Tracktor++v2 (O) [6]	Tracktor	54.9	56.2	2394	76844	617
GNNMatch (O) [44]	Tracktor	55.9	56.9	3235	74784	564
GSM_Tracktor (O)[39]	Tracktor	58.2	57.0	4332	73573	475
GMTracker(Ours) (O)	Tracktor	63.9	55.9	2371	77545	531
GMT_CT (Ours) (O)	CenterTrack	68.6	62.6	5104	62377	787
TPM [46]	-	47.9	51.3	2701	85504	569
eTC [60]	-	56.1	49.2	8400	83702	606
MPNTrack [8]	Tracktor	61.7	58.6	4949	70252	354
Lif_TsimInt [18]	Tracktor	64.1	57.5	4249	72868	335
LifT [18]	Tracktor	64.7	61.3	4844	65401	389
GMT_simInt (Ours)	Tracktor	66.2	59.1	6021	68226	341
GMT_VIVE (Ours)	Tracktor	66.6	61.1	3891	66550	503
GMTCT_simInt (Ours)	CenterTrack	70.6	66.2	6355	54560	701

Table 5: Comparison with state-of-the-art methods on MOT16 and MOT17 *test* set. (O) denotes online methods. (O*) denotes near-online methods.

metrics, e.g., HOTA [41].

6. Conclusion

In this paper, we propose a novel learnable graph matching method for multiple object tracking task, called GMTracker. Our graph matching method focuses on the relationship between tracklets and detections. Taking the second-order edge-to-edge similarity into account, our tracker is more accurate and robust in the MOT task, especially in crowded videos. To make the graph matching module end-to-end differentiable, we relax the QAP formulation into a convex QP and build a differentiable graph matching layer in our Graph Matching Network. The experiments of ablation study and comparison with other state-of-the-art methods both show the effectiveness of our method.

Acknowledgements

This work was supported in part by the National Key R&D Program of China(No. 2018YFB1004602), the National Natural Science Foundation of China (No. 61836014, No. 61773375). The authors would like to thank Roberto Henschel for running their post-processing code for us.

Appendix

A. Gradients of the Graph Matching Layer

As described in Section 4.3 of our main paper, the gradients of the graph matching layer we need for backward can be derived from the KKT conditions with the help of the implicit function theorem. Here, we show the details of deriving the gradients.

For a quadratic programming (QP), the standard formulation is as

$$\begin{aligned} & \underset{x}{\text{minimize}} && \frac{1}{2}x^\top Q(\theta)x + q(\theta)^\top x \\ & \text{subject to} && G(\theta)x \leq h(\theta) \\ & && A(\theta)x = b(\theta). \end{aligned} \quad (\text{A})$$

So the Lagrangian is given by

$$L(x, \nu, \lambda) = \frac{1}{2}x^\top Qx + \lambda^\top (Gx - h) + q^\top x + \nu^\top (Ax - b), \quad (\text{B})$$

where, ν and λ are the dual variables.

The (x^*, λ^*, ν^*) are the optimal solution if and only if they satisfy the KKT conditions:

$$\begin{aligned} \nabla_x L(x^*, \lambda^*, \nu^*) &= 0 \\ Qx^* + q + A^\top \nu^* + G^\top \lambda^* &= 0 \\ Ax^* - b &= 0 \\ \text{diag}(\lambda^*)(Gx^* - h) &= 0 \\ Gx^* - h &\leq 0 \\ \lambda^* &\geq 0. \end{aligned} \quad (\text{C})$$

We define the function

$$g(x, \lambda, \nu, \theta) = \begin{bmatrix} \nabla_x L(x, \lambda, \nu, \theta) \\ \text{diag}(\lambda)\lambda^\top (G(\theta)x - h(\theta)) \\ A(\theta)x - b(\theta) \end{bmatrix}, \quad (\text{D})$$

and the optimal solution x^*, λ^*, ν^* satisfy the equation $g(x^*, \lambda^*, \nu^*, \theta) = 0$.

According to the implicit function theorem, as proven in [4], the gradients where the primal variable x and the dual variables ν and λ are the optimal solution, can be formulated as

$$J_\theta x^* = -J_x g(x^*, \lambda^*, \nu^*, \theta)^{-1} J_\theta g(x^*, \lambda^*, \nu^*, \theta), \quad (\text{E})$$

where, $J_x g(x^*, \lambda^*, \nu^*, \theta)$ and $J_\theta g(x^*, \lambda^*, \nu^*, \theta)$ are the Jacobian matrices. Each element of them is the partial derivative of function g with respect to variable x and θ , respectively.

	IDF1	MOTA	MT	ML	FP	FN	ID Sw.
Baseline	68.1	62.1	556	371	1923	124480	1135
Ours	71.5	62.3	555	375	1741	124298	1017
Oracle	77.2	62.6	545	368	1730	124287	14

Table A: Comparison between the baseline, our GMTracker and the Oracle tracker on MOT17 *val* set.

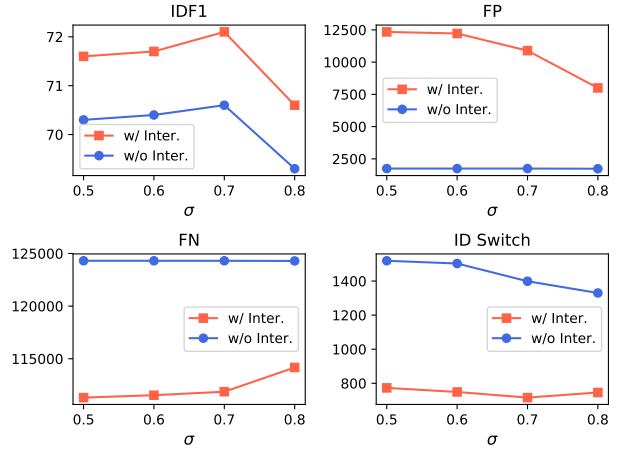


Figure A: Results on IDF1, FP, FN and ID Switch metrics under different threshold σ of the feature similarity to create a new tracklet.

B. Pseudo-code of Our Algorithm

To make our algorithm clear and easy to understand, we show the pseudo code of our GMTracker algorithm in Alg. A. The input of the algorithm is the detection set $\mathcal{D}^t = \{D_1^t, D_2^t, \dots, D_{n_d}^t\}$ and tracklet set $\mathcal{T}^t = \{T_1^t, T_2^t, \dots, T_{n_t}^t\}$, defined in Section 4.1 of our main paper. And the output is the new tracklet set \mathcal{T}^{t+1} to be associated in the next frame. The motion gate κ is 9.4877. The feature similarity threshold σ is 0.6 in the videos taken by the moving camera, and 0.7 in the videos taken by the static camera. The max age δ is 100 frames.

C. Additional Experiments and Analyses

C.1. Comparison with the Oracle Tracker

To explore the upper bound of the association method, we compare our method with the ground truth association, called the Oracle tracker. The results on MOT17 *val* set are shown in Table A. There is a gap of 5.7 IDF1 and about 1000 ID Switches between our online GMTracker and the Oracle tracker.

Another observation is that on some metrics, which are extremely relevant to detection results, like MOTA, FP and FN, the gaps between the baseline, our method and the Or-

acle tracker are relatively small. That is why we mainly concern with the metrics reflecting the association results, such as IDF1 and ID Switch.

C.2. Discussions

Tracklet born strategies. In our GMTracker, the tracklet born strategies mostly follow DeepSORT, but we also make some improvements to make these strategies more suitable for our approach, as described in Section 4.4 in our main paper. Among the three criteria to create a new tracklet, we find that the threshold σ is the most sensitive hyperparameter in our method. We conduct experiments with different σ , and its influence on IDF1, FP, FN and ID Switch is shown in Fig. A.

C.3. Detailed Performance

As shown in Table B, the results on more metrics, such as HOTA, AssA, DetA, LocA, MT, ML are provided for better comparison.

Algorithm A: GMTracker Algorithm

```

Input:  $\mathcal{D}^t, \mathcal{T}^t$ 
Output:  $\mathcal{T}^{t+1}$ 
for  $D_i^t \in \mathcal{D}^t$  do
     $\mathbf{a}_D^{i,t} \leftarrow \text{MLP}_a(\text{ReID}(\mathbf{I}_i^t))$ 
     $\mathbf{h}_i^{(0)} \leftarrow \mathbf{a}_D^{i,t}$ 
for  $T_j^t \in \mathcal{T}^t$  do
    for  $D_{(j)}^k \in T_j^t$  do
         $\mathbf{a}_D^{(j),k} \leftarrow \text{MLP}_a(\text{ReID}(\mathbf{I}_{(j)}^k))$ 
     $\mathbf{a}_T^{j,t} \leftarrow \text{mean}(\mathbf{a}_D^{(j),k})$ 
     $\mathbf{h}_j^{(0)} \leftarrow \mathbf{a}_T^{j,t}$ 
for  $l \leq l_{max}$  do
    for  $D_i^t \in \mathcal{D}^t$  do
         $\mathbf{m}_i^{(l)} \leftarrow \mathcal{A}(\{w_{i,j}^{(l)} \mathbf{h}_j^{(l)} \mid j \in \mathcal{G}_T\})$ 
         $\mathbf{h}_i^{(l+1)} \leftarrow \mathcal{F}(\mathbf{h}_i^{(l)}, \mathbf{m}_i^{(l)})$ 
    for  $T_j^t \in \mathcal{T}^t$  do
         $\mathbf{m}_j^{(l)} \leftarrow \mathcal{A}(\{w_{i,j}^{(l)} \mathbf{h}_i^{(l)} \mid i \in \mathcal{G}_D\})$ 
         $\mathbf{h}_j^{(l+1)} \leftarrow \mathcal{F}(\mathbf{h}_j^{(l)}, \mathbf{m}_j^{(l)})$ 
for  $D_i^t, D_{i'}^t \in \mathcal{D}^t$  do
     $\mathbf{h}_i \leftarrow \mathbf{h}_i^{(l+1)}, \mathbf{h}_{i'} \leftarrow \mathbf{h}_{i'}^{(l+1)}$ 
     $\mathbf{h}_{i,i'} \leftarrow l_2([\mathbf{h}_i, \mathbf{h}_{i'}])$ 
for  $T_i^t, T_{j'}^t \in \mathcal{T}^t$  do
     $\mathbf{h}_j \leftarrow \mathbf{h}_j^{(l+1)}, \mathbf{h}_{j'} \leftarrow \mathbf{h}_{j'}^{(l+1)}$ 
     $\mathbf{h}_{j,j'} \leftarrow l_2([\mathbf{h}_j, \mathbf{h}_{j'}])$ 
for  $D_i^t, T_j^t \in \mathcal{D}^t, \mathcal{T}^t$  do
     $\mathbf{M}_e^{u,v} \leftarrow \mathbf{h}_{i,i'}^\top \mathbf{h}_{j,j'}$ 
     $\mathbf{B}_{i,j} \leftarrow \mathbf{h}_i^\top \mathbf{h}_j$ 
match  $\leftarrow \text{graph\_matching}(\mathbf{M}_e, \mathbf{B})$ 
for  $D_i^t, T_j^t \in \mathcal{D}^t, \mathcal{T}^t$  do
    if  $\text{IoU}(D_i^t, T_j^t) \leq 0$  or  $d(D_i^t, T_j^t) >$ 
         $\kappa$  or  $\cos(D_i^t, T_j^t) < \sigma$  then
         $\text{delete}(\text{match}(i, j))$ 
for  $D_i^t, T_j^t \in \mathcal{D}_{unmatch}^t, \mathcal{T}_{unmatch}^t$  do
    if  $\text{IoU}(D_i^t, T_j^t) \geq 0.3$  then
         $\text{match}_{add} \leftarrow \text{Hungarian}(\text{IoU}(D_i^t, T_j^t))$ 
for  $D_i^t, T_j^t \in \mathcal{D}^t, \mathcal{T}^t$  do
    if  $\text{match}(i, j)$  or  $\text{match}_{add}(i, j)$  then
         $T_j^{t+1} \leftarrow T_j^t + \{D_i^t\}$ 
         $\text{motion}(T_j^{t+1}).\text{update}()$ 
    if  $D_i^t \in \mathcal{D}_{unmatch}^t$  then
         $T_{new}^{t+1} \leftarrow \{D_i^t\}$ 
    if  $T_j^t.\text{last\_update} > \delta$  then
         $\text{delete}(T_j^t)$ 
return  $\mathcal{T}^{t+1}$ 

```

Methods	Refined Det	IDF1 \uparrow	HOTA \uparrow	MOTA \uparrow	MT \uparrow	ML \downarrow	FP \downarrow	FN \downarrow	IDS \downarrow	AssA \uparrow	DetA \uparrow	LocA \uparrow
MOT17												
GNMOT (O*) [34]	-	47.0	-	50.2	19.3	32.7	29316	246200	5273	-	-	-
FAMNet (O) [10]	-	48.7	-	52.0	19.1	33.4	14138	253616	3072	-	-	-
JBNOT (O*) [17]	-	50.8	41.3	52.6	19.7	35.8	31572	232659	3050	39.8	43.3	80.2
Tracktor++ (O) [6]	Tracktor	52.3	42.1	53.5	19.5	36.6	12201	248047	2072	41.7	42.9	80.9
Tracktor++v2 (O) [6]	Tracktor	55.1	44.8	56.3	21.1	35.3	8866	235449	1987	45.1	44.9	81.8
GNNMatch (O) [44]	Tracktor	56.1	45.4	57.0	23.3	34.6	12283	228242	1957	45.2	45.9	81.5
GSM_Tracktor (O) [39]	Tracktor	57.8	45.7	56.4	22.2	34.5	14379	230174	1485	47.0	44.9	80.9
CTTrackPub (O) [76]	CenterTrack	59.6	48.2	61.5	26.4	31.9	14076	200672	2583	47.8	49.0	81.7
GMTracker(Ours) (O)	Tracktor	63.8	49.1	56.2	21.0	35.5	8719	236541	1778	53.9	44.9	81.8
GMT_CT(Ours) (O)	CenterTrack	66.9	52.0	61.5	26.3	32.1	14059	200655	2415	55.1	49.4	81.8
TPM [46]	-	52.6	41.5	54.2	22.8	37.5	13739	242730	1824	40.9	42.5	80.0
eTC17 [60]	-	58.1	44.9	51.9	23.1	35.5	36164	232783	2288	47.0	43.3	79.4
MPNTrack [8]	Tracktor	61.7	49.0	58.8	28.8	33.5	17413	213594	1185	51.1	47.3	81.5
Lif_TsimInt [18]	Tracktor	65.2	50.7	58.2	28.6	33.6	16850	217944	1022	54.9	47.1	81.5
LifT [18]	Tracktor	65.6	51.3	60.5	27.0	33.6	14966	206619	1189	54.7	48.3	81.3
GMT_simInt (Ours)	Tracktor	65.9	51.1	59.0	29.0	33.6	20395	209553	1105	55.1	47.6	81.2
GMT_VIVE (Ours)	Tracktor	65.9	51.2	60.2	26.5	33.2	13142	209812	1675	55.1	47.8	81.3
GMTCT_simInt (Ours)	CenterTrack	68.7	54.0	65.0	29.4	31.6	18213	177058	2200	56.4	52.0	81.5
MOT16												
Tracktor++v2 (O) [6]	Tracktor	54.9	44.6	56.2	20.7	35.8	2394	76844	617	44.6	44.8	82.0
GNNMatch (O) [44]	Tracktor	55.9	44.6	56.9	22.3	35.3	3235	74784	564	43.7	45.8	81.7
GSM_Tracktor (O)[39]	Tracktor	58.2	45.9	57.0	22.0	34.5	4332	73573	475	46.7	45.4	81.1
GMTracker(Ours) (O)	Tracktor	63.9	48.9	55.9	20.3	36.6	2371	77545	531	53.7	44.6	82.1
GMT_CT (Ours) (O)	CenterTrack	68.6	53.1	62.6	26.7	31.0	5104	62377	787	56.3	50.4	81.8
TPM [46]	-	47.9	36.7	51.3	18.7	40.8	2701	85504	569	34.6	39.3	79.1
eTC [60]	-	56.1	42.0	49.2	17.3	40.3	8400	83702	606	44.5	39.9	78.8
MPNTrack [8]	Tracktor	61.7	48.9	58.6	27.3	34.0	4949	70252	354	51.1	47.1	81.7
Lif_TsimInt [18]	Tracktor	64.1	49.6	57.5	25.4	34.7	4249	72868	335	53.3	46.5	81.9
LifT [18]	Tracktor	64.7	50.8	61.3	27.0	34.0	4844	65401	389	53.1	48.9	81.4
GMT_simInt (Ours)	Tracktor	66.2	51.2	59.1	27.5	34.4	6021	68226	341	55.1	47.7	81.5
GMT_VIVE (Ours)	Tracktor	66.6	51.6	61.1	26.7	33.3	3891	66550	503	55.3	48.5	81.5
GMTCT_simInt (Ours)	CenterTrack	70.6	55.2	66.2	29.6	30.4	6355	54560	701	57.8	53.1	81.5

Table B: Detailed comparison with state-of-the-art methods on MOT16 and MOT17 *test* set. (O) denotes online methods. (O*) denotes near-online methods.

References

- [1] Yonathan Aflalo, Alexander Bronstein, and Ron Kimmel. On convex relaxation of graph isomorphism. *Proceedings of the National Academy of Sciences*, 112(10):2942–2947, 2015. 4
- [2] Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *ICML*, 2017. 2, 3, 6
- [3] Yaakov Bar-Shalom, Thomas E Fortmann, and Peter G Cable. Tracking and data association, 1990. 2
- [4] Shane Barratt. On the differentiability of the solution to convex optimization problems. *arXiv preprint arXiv:1804.05098*, 2018. 3, 9
- [5] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011. 1, 2
- [6] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. In *ICCV*, 2019. 2, 6, 8, 11
- [7] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, 2016. 1, 2
- [8] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *CVPR*, 2020. 1, 2, 5, 6, 7, 8, 11
- [9] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *ICCV*, 2015. 2
- [10] Peng Chu and Haibin Ling. FAMNet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking. In *ICCV*, 2019. 8, 11
- [11] Steven Diamond and Stephen Boyd. CVXPY: A python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016. 6
- [12] Georgios D Evangelidis and Emmanouil Z Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1858–1865, 2008. 6
- [13] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, 2007. 2
- [14] Seyed Hamid Rezatofighi, Anton Milan, Zhen Zhang, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint probabilistic data association revisited. In *ICCV*, 2015. 2
- [15] Juris Hartmanis. *Computers and intractability: a guide to the theory of NP-completeness*. 1982. 3
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6
- [17] Roberto Henschel, Yunzhe Zou, and Bodo Rosenhahn. Multiple people tracking using body and joint detections. In *CVPR Workshops*, 2019. 8, 11
- [18] Andrea Hornakova, Roberto Henschel, Bodo Rosenhahn, and Paul Swoboda. Lifted disjoint paths with application in multiple object tracking. In *ICML*, 2020. 1, 2, 6, 7, 8, 11
- [19] Min Hu, Saad Ali, and Mubarak Shah. Detecting global motion patterns in complex videos. In *ICPR*, 2008. 2
- [20] Weiming Hu, Xinchu Shi, Zongwei Zhou, Junliang Xing, Haibin Ling, and Stephen Maybank. Dual L1-normalized context aware tensor power iteration and its applications to multi-object tracking and multi-graph matching. *International Journal of Computer Vision*, 128(2):360–392, 2020. 3
- [21] Xiaolong Jiang, Peizhao Li, Yanjing Li, and Xiantong Zhen. Graph neural based end-to-end data association framework for online multiple-object tracking. *arXiv preprint arXiv:1907.05315*, 2019. 2
- [22] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 82(1):35–45, 1960. 5
- [23] Rangachar Kasturi, Dmitry Goldgof, Padmanabhan Soundararajan, Vasant Manohar, John Garofolo, Rachel Bowers, Matthew Boonstra, Valentina Korzhova, and Jing Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):319–336, 2008. 6
- [24] Chanh Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *ICCV*, 2015. 2
- [25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 6
- [26] Tjalling C. Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25(1):53–76, 1957. 2, 3
- [27] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 2, 3
- [28] Cheng-Hao Kuo and Ram Nevatia. How does person identity recognition help multi-person tracking? In *CVPR*, 2011. 1
- [29] Long Lan, Dacheng Tao, Chen Gong, Naiyang Guan, and Zhigang Luo. Online multi-object tracking by quadratic pseudo-boolean optimization. In *IJCAI*, 2016. 2
- [30] Eugene L. Lawler. The quadratic assignment problem. *Management Science*, 9(4):586–599, 1963. 2, 3
- [31] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese CNN for robust target association. In *CVPR Workshops*, 2016. 1, 2
- [32] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005. 3
- [33] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. DeeperGCN: All you need to train deeper GCNs. *arXiv preprint arXiv:2006.07739*, 2020. 5
- [34] Jiahe Li, Xu Gao, and Tingting Jiang. Graph networks for multiple object tracking. In *WACV*, pages 719–728, 2020. 2, 8, 11
- [35] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. DeepReID: Deep filter pairing neural network for person re-identification. In *CVPR*, 2014. 6
- [36] Tianyi Liang, Long Lan, and Zhigang Luo. Enhancing the association in multi-object tracking via neighbor graph. *arXiv preprint arXiv:2007.00265*, 2020. 2

- [37] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1
- [38] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 1
- [39] Qiankun Liu, Qi Chu, Bin Liu, and Nenghai Yu. GSM: Graph similarity model for multi-object tracking. In *IJCAI*, 2020. 2, 8, 11
- [40] Zhichao Lu, Vivek Rathod, Ronny Votel, and Jonathan Huang. RetinaTrack: Online single stage joint detection and tracking. In *CVPR*, 2020. 2
- [41] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip H. S. Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. HOTA: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, 129(2):548–578, 2021. 8
- [42] Cong Ma, Yuan Li, Fan Yang, Ziwei Zhang, Yueqing Zhuang, Huizhu Jia, and Xiaodong Xie. Deep association: End-to-end graph-based learning for multiple object tracking with conv-graph neural network. In *ICMR*, 2019. 5
- [43] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. 6
- [44] Ioannis Papakis, Abhijit Sarkar, and Anuj Karpatne. GC-NNMatch: Graph convolutional neural networks for multi-object tracking via sinkhorn normalization. *arXiv preprint arXiv:2010.00067*, 2020. 8, 11
- [45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 6
- [46] Jinlong Peng, Tao Wang, Weiyao Lin, Jian Wang, John See, Shilei Wen, and Erui Ding. TPM: Multiple object tracking with tracklet-plane matching. *Pattern Recognition*, 107:107480, 2020. 2, 8, 11
- [47] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011. 2
- [48] Donald Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979. 2
- [49] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCV Workshops*, 2016. 6
- [50] Mikel Rodriguez, Saad Ali, and Takeo Kanade. Tracking in unstructured crowded scenes. In *ICCV*, 2009. 2
- [51] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *ICCV*, 2017. 2
- [52] Christian Schellewald and Christoph Schnörr. Probabilistic subgraph matching based on convex relaxation. In *CVPR Workshops*, 2005. 3
- [53] ShiJie Sun, Naveed Akhtar, HuanSheng Song, Ajmal S Mian, and Mubarak Shah. Deep affinity network for multiple object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):104–119, 2019. 1
- [54] Paul Swoboda, Carsten Rother, Hassan Abu Alhaija, Dagmar Kainmuller, and Bogdan Savchynskyy. A study of Lagrangean decompositions and dual ascent solvers for graph matching. In *CVPR*, 2017. 3
- [55] Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Subgraph decomposition for multi-target tracking. In *CVPR*, 2015. 2
- [56] Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Multi-person tracking by multicut and deep matching. In *ECCV*, 2016. 2
- [57] Philip HS Torr. Solving markov random fields using semi definite programming. In *AISTATS*, 2003. 3
- [58] Mario Vento and Pasquale Foggia. Graph matching techniques for computer vision. In *Image Processing: Concepts, Methodologies, Tools, and Applications*, pages 381–421. 2013. 2
- [59] Bing Wang, Gang Wang, Kap Luk Chan, and Li Wang. Tracklet association by online target-specific metric learning and coherent dynamics estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(3):589–602, 2016. 2
- [60] Gaoang Wang, Yizhou Wang, Haotian Zhang, Renshu Gu, and Jenq-Neng Hwang. Exploit the connectivity: Multi-object tracking with trackletnet. In *ACM MM*, 2019. 8, 11
- [61] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *ICCV*, 2019. 3
- [62] Shaofei Wang and Charless C Fowlkes. Learning optimal parameters for multi-target tracking with contextual interactions. *International Journal of Computer Vision*, 122(3):484–501, 2017. 1
- [63] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris Kitani. GNN3DMOT: Graph neural network for 3D multi-object tracking with 2D-3D multi-feature learning. *CVPR*, 2020. 5
- [64] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, 2017. 1, 2, 6, 7
- [65] Yihong Xu, Aljosa Osep, Yutong Ban, Radu Horaud, Laura Leal-Taixé, and Xavier Alameda-Pineda. How to train your deep multi-object tracker. In *CVPR*, 2020. 1, 2
- [66] Bo Yang, Chang Huang, and Ram Nevatia. Learning affinities and dependencies for multi-target tracking using a CRF model. In *CVPR*, 2011. 2
- [67] Bo Yang and Ram Nevatia. An online learned CRF model for multi-target tracking. In *CVPR*, 2012. 1
- [68] Ju Hong Yoon, Chang-Ryeol Lee, Ming-Hsuan Yang, and Kuk-Jin Yoon. Online multi-object tracking via structural constraint event aggregation. In *CVPR*, 2016. 2
- [69] Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. Learning deep graph matching with channel-independent embedding and Hungarian attention. In *ICLR*, 2020. 3
- [70] Amir Roshan Zamir, Afshin Dehghan, and Mubarak Shah. GMCP-Tracker: Global multi-object tracking using generalized minimum clique graphs. In *ECCV*, 2012. 2

- [71] Andrei Zanfir and Cristian Sminchisescu. Deep learning of graph matching. In *CVPR*, 2018. [4](#)
- [72] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, 2008. [2](#)
- [73] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. FairMOT: On the fairness of detection and re-identification in multiple object tracking. *arXiv preprint arXiv:2004.01888*, 2020. [2](#)
- [74] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *ICCV*, 2015. [6](#)
- [75] Feng Zhou and Fernando De la Torre. Factorized graph matching. In *CVPR*, 2012. [3](#)
- [76] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *ECCV*, 2020. [2](#), [8](#), [11](#)
- [77] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. In *ECCV*, 2018. [1](#)