Learnable Graph Matching: A Practical Paradigm for Data Association

Jiawei He, Zehao Huang, Naiyan Wang, and Zhaoxiang Zhang

Abstract—Data association is at the core of many computer vision tasks, e.g., multiple object tracking, image matching, and point cloud registration. Existing methods usually solve the data association problem by network flow optimization, bipartite matching, or end-to-end learning directly. Despite their popularity, we find some defects of the current solutions: they mostly ignore the intra-view context information; besides, they either train deep association models in an end-to-end way and hardly utilize the advantage of optimization-based assignment methods, or only use an off-the-shelf neural network to extract features. In this paper, we propose a general learnable graph matching method to address these issues. Especially, we model the intra-view relationships as an undirected graph. Then data association turns into a general graph matching problem between graphs. Furthermore, to make optimization end-to-end differentiable, we relax the original graph matching problem into continuous quadratic programming and then incorporate training into a deep graph neural network with KKT conditions and implicit function theorem. In MOT task, our method achieves state-of-the-art performance on several MOT datasets. For image matching, our method outperforms state-of-the-art methods with half training data and iterations on a popular indoor dataset, ScanNet. Code will be available at https://github.com/jiaweihe1996/GMTracker.

Index Terms—Graph matching, data association, multiple object tracking, image matching.

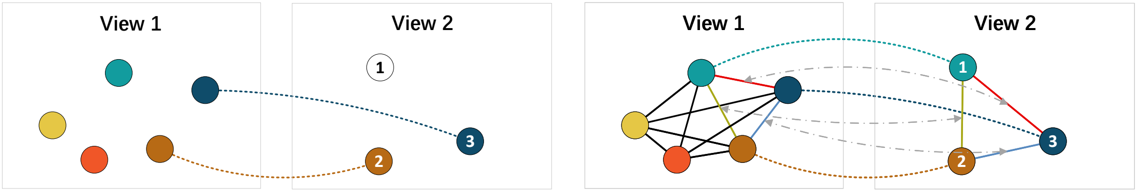
1 Introduction

DATA association is at the core of many computer vision tasks, for example, instances with the same identity are associated between different frames in *Multiple Object Tracking*, 2D or 3D keypoints are associated between different views for *Image Matching* and *Point Cloud Registration*. These tasks can be described as detecting entities (objects/keypoints/points) in different views, then establishing the correspondences between entities in these views. In this paradigm, the latter process is called *Data Association* and becomes the important part of these tasks. The traditional methods define data association task as a bipartite matching problem, ignoring the context information in each view, i.e., the pairwise relationship between entities. In this paper, we argue that the relationship between the entities within the same view is also crucial for some challenging cases in data association. Interestingly, these pairwise relationships within the same view can be represented as edges in a general graph. To this end, the popular bipartite matching across views 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 [2], [3] 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 [4]. Finally, the assignment problem can be learned in synergy with the features. In Fig. 1, we show how pairwise relationship is used in our learnable graph matching method.

To reveal the effectiveness and universality of our learnable graph matching method, we apply our method to some important and popular compuer vision tasks, *Multiple Object Tracking* (MOT) and *Image Matching*. MOT is a fundamental computer vision 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 autonomous 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 [5], [6]. As for the second one, recent MOT work focuses 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 [7], [8], [9], [10], [11], [12]; (2) improving the appearance models by the power of deep learning [13], [14], [15], [16]. Although very recently, some work [10], [11], [17], [18] 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 this paper, we propose a learnable graph matching based online tracker, called GMTracker. We construct the tracklet graph and the detection graph, and put the detec-

- *Jiawei He and Zhaoxiang Zhang are with Center for Research on Intelligent Perception and Computing (CRIPAC), National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing 100190, China, and also with School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: {hejiawei2019, zhaoxiang.zhang}@ia.ac.cn.*
- *Zhaoxiang Zhang is also with Centre for Artificial Intelligence and Robotics, Hong Kong Institute of Science and Innovation, Chinese Academy of Sciences (HKISI_CAS), Hong Kong, China.*
- *Zehao Huang and Naiyan Wang are with Tusimple, Beijing 100020, China. E-mail: {zehaohuang18, winsty}@gmail.com.*



(a) Bipartite Matching [1]

(b) Graph Matching

Fig. 1: An illustration of intra-graph relationship used in our graph matching formulation. We utilize the second-order edge to model the pairwise relationship, which is more robust in challenging scenes, such as heavy occluded in MOT task. For example, in view 2, the entity with ID 1 can not be associated with the entity correctly in view 1. However, with graph matching, the pairwise relationship helps data association.

tions and tracklets on the vertices respectively. The edge features on each graph represent the pairwise relationship between two connected vertices. According to the general design of the learnable graph matching module mentioned above, the learnable vertex and edge features on the two graphs are the input of the differentiable GM layer. The supervision acts on these features constrained by graph matching solutions. So the learning process converges to a more robust and reasonable solution than the traditional approach.

However, when the quantity of the objects increases, the graph matching method is not efficient enough. The time cost will become unbearable. So, we speed up the graph matching solver by introducing the gate mechanism, called Gated Search Tree (GST) for MOT. The feasible region is limited much smaller than the original quadratic programming formulation, so that the running time of solving the graph matching problem can be substantially reduced.

In Image Matching task, recently, learning-based methods have been popular. SuperGlue [19] is one of the representative work. Taking the keypoints from traditional methods, e.g., SIFT [20], or learning-based methods, e.g., SuperPoint [21] as the input, SuperGlue proposes transformer-based network to aggregate the keypoint features and matching the corresponding keypoints by a Sinkhorn layer. However, Sinkhorn is only a kind of learnable bipartite matching method, and no intra-frame relationship has been utilized explicitly. Based on SuperGlue [19], we construct the graph in each frame and replace the Sinkhorn matching module with our differentiable graph matching network, called GMatcher. On the widely used indoor image matching dataset ScanNet [22], the result fully embodies the characteristics of fast convergence and good performance of our learnable graph matching method.

In summary, our work has the following contributions:

- Instead of only focusing on data association across views, we emphasize the importance of intra-view relationships. Particularly, we propose to represent the relationships as a general graph, and formulate the data 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 design the Gated Search Tree (GST) algorithm, greatly accelerating the process of solving quadratic assignment problem in data association. Utilizing the new GST algorithm, the association stage is about 21× faster than the original Quadratic Programming solver.
- In MOT task, 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 metrics.
- In image matching task, compared with SOTA method SuperGlue, we use about half training data and training iterations and obtain the gain of about 1 camera pose estimation AUC.

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 [23] and MHT [24]. 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 [25], [26]. For combinatorial optimization, traditional approach include bipartite matching [8], dynamic programming [27], min-cost flow [7], [28] and conditional random field [29]. Follow-up work tried to adopt more complex optimization methods [30], [31], reduce the computational cost [32], [33] or promote an online setting from them [34], [35]. Early work of deep learning-based association in MOT such as [15], [16], [36], [37] mostly focus on learning a better appearance model for each object. More recently, several work tried to bridge the graph optimization and end-to-end deep learning [10], [11], [12], [37], [38]. [38] adopts Graph Neural Network (GNN) to learn an affinity matrix in a data-driven way. MPNTrack [10] introduces a message passing network to learn high-order information between vertices from different frames. [37] constructs two graph networks to model appearance and motion features, respectively. LifT [12] proposes a lifted disjoint path formulation for MOT, which introduces lifted edges to capture long term temporal interactions. [39] 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 [39] 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.

Data association in image matching. Image matching is a traditional computer vision task, and the main procedures include keypoint detection on the image, keypoint feature extraction, and keypoint matching. Earlier researches mainly focus on the robust keypoint detectors and feature extractors, especially for the local feature descriptor with scale-invariance, rotation-invariance and translation-invariance, such as SIFT [20], SURF [40] and ORB [41]. And the matching algorithms are based on the Nearest Neighbor (NN) search with different outlier filtering methods [20], [42], [43]. In the era of deep learning, the CNN-based feature detectors and extractors have emerged, such as D2-net [44] and SuperPoint [21]. However, the exploration of end-to-end learnable matching algorithms has just begun. SuperGlue [19] utilizes the attentional graph neural network to model the long-range relationship in the image, and the matching module is designed as a differentiable Sinkhorn layer, which is the approximation of graph matching without explicitly modeling the edge in the graph. LoFTR [45] is a kind of dense matching algorithm, from patch matching to pixel matching in coarse-to-fine style.

Graph matching and combinatorial optimization. Pairwise graph matching, or more generally Quadratic Assignment Problem (QAP), has wide applications in various computer vision tasks [46]. Compared with the linear assignment problem that only considers vertex-to-vertex relationship, 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 [47], this problem is an NP-hard problem. There is no polynomial solver like Hungarian algorithm [1] for the linear assignment problem. In the past decades, many work engages in making the problem tractable by relaxing the original QAP problem [48], [49], [50]. Lagrangian decomposition [51] and factorized graph matching [52] are two representative ones. 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 [53], [54]. 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 [55]. Inspired by it, in this paper, we propose a learnable graph matching layer to solve the challenging graph matching problem in data association.

3 Graph Matching Formulation for Data Association

In this section, we will formulate data association 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 Basic Graph Matching Formulation for Data Association

We define the aim of data association is to match the vertices in graph $\mathcal{G}_1$ and $\mathcal{G}_2$ constructd in view 1 and view 2 respectively. So, it can be seen as a graph matching problem, which is to maximize the similarities between the matched vertices and corresponding edges connected by these vertices. As defined in [2], the graph matching problem is a Quadratic Assignment Problem (QAP) . A practical mathematical form is named *Koopmans-Beckmann's* QAP [3]:

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

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.

3.2 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 [52], Eq. 1 can be rewritten as

$$\mathbf{\Pi}^* = \arg\min_{\mathbf{\Pi}} \frac{1}{2}||\mathbf{A_1}\mathbf{\Pi} - \mathbf{\Pi}\mathbf{A_2}||_F^2 - \text{tr}(\mathbf{B}^\top\mathbf{\Pi}). \tag{2}$$

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 [56], the QAP (Eq. 2) can be relaxed to its tightest convex relaxation by only constraining the permutation matrix $\mathbf{\Pi}$ to be a double stochastic matrix $\mathbf{X}$, formed as the following QP problem:

$$\mathbf{X}^* = \arg\min_{\mathbf{X}\in\mathcal{D}} \frac{1}{2}||\mathbf{A_1}\mathbf{X} - \mathbf{X}\mathbf{A_2}||_F^2 - \text{tr}(\mathbf{B}^\top\mathbf{X}), \tag{3}$$

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

4 Graph Matching for MOT

In this section, we introduce the problem definition of MOT and our graph matching formulation for the data association in MOT task.

4.1 Detection and Tracklet Graphs Construction

As an online tracker, we track objects frame by frame. In frame t, we define $\mathcal{D}^t = \{D_1^t, D_2^t, \cdots, D_{n_d}^t\}$ as the set of detections in current frame and $\mathcal{T}^t = \{T_1^t, T_2^t, \cdots, T_{n_t}^t\}$ 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.

4.2 From Edge Weights to Edge Features

In the general formulation of graph matching, 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 $\mathbf{H}^{\cdot,i,i'} = \mathbf{h}_{i,i'}$, 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 $\mathbf{H_D}$ and $\mathbf{H_T}$ are the weighted adjacency tensors for $\mathcal{G}_D$ and $\mathcal{G}_T$, 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:

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

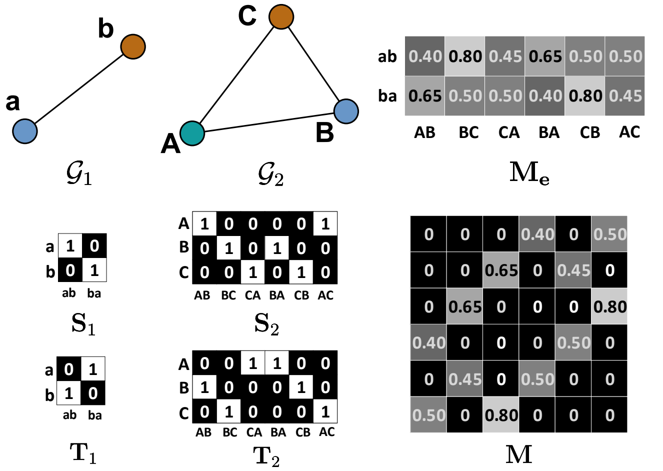


Fig. 2: An example of the derivation from edge affinity matrix $\mathbf{M_e}$ to quadratic affinity matrix $\mathbf{M}$.

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

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

where $\boldsymbol{\pi} = \text{vec}(\mathbf{\Pi})$, $\mathbf{b} = \text{vec}(\mathbf{B})$ 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 3.2, the formulation Eq. 5 using edge features can be relaxed to a QP:

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

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

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

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 [57], we map each element of matrix $\mathbf{M_e}$ to the *symmetric* quadratic affinity matrix $\mathbf{M}$:

$$\mathbf{M} = (\mathbf{S_D} \otimes \mathbf{S_T}) \text{diag}(\text{vec}(\mathbf{M_e})) (\mathbf{T_D} \otimes \mathbf{T_T})^\top, \tag{8}$$

where 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:

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

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

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

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 $j \in \mathcal{V}_T$:

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

5 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.

5.1 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.

5.2 Cross-Graph GCN

Similar to [10], [58], [59], 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:

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

where $\cos(\cdot,\cdot)$ means the cosine similarity between input features and $\mathrm{IoU}(\cdot,\cdot)$ 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 4.1. As for a tracklet vertex j, we estimate the bounding box $\mathbf{g}_j$ in current frame t by Kalman Filter [60] 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:

$$\mathbf{h}_i^{(l+1)} = \mathrm{MLP}(\mathbf{h}_i^{(l)} + \frac{\|\mathbf{h}_i^{(l)}\|_2 \mathbf{m}_i^{(l)}}{\|\mathbf{m}_i^{(l)}\|_2}), \tag{13}$$

where we adopt message normalization proposed in [61] 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.

5.3 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 [4]. In our implementation, we adopt the qpth library [4] 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 [62] only for forward operation.

For training, we use weighted binary cross entropy Loss:

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

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:

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

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

5.4 Gradients of the Graph Matching Layer

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 of a standard QP optimization.

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

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2} x^\top Q(\theta) x + q(\theta)^\top x \\ \text{subject to} \quad & G(\theta) x \le h(\theta) \\ & A(\theta) x = b(\theta). \end{aligned} \tag{16}$$

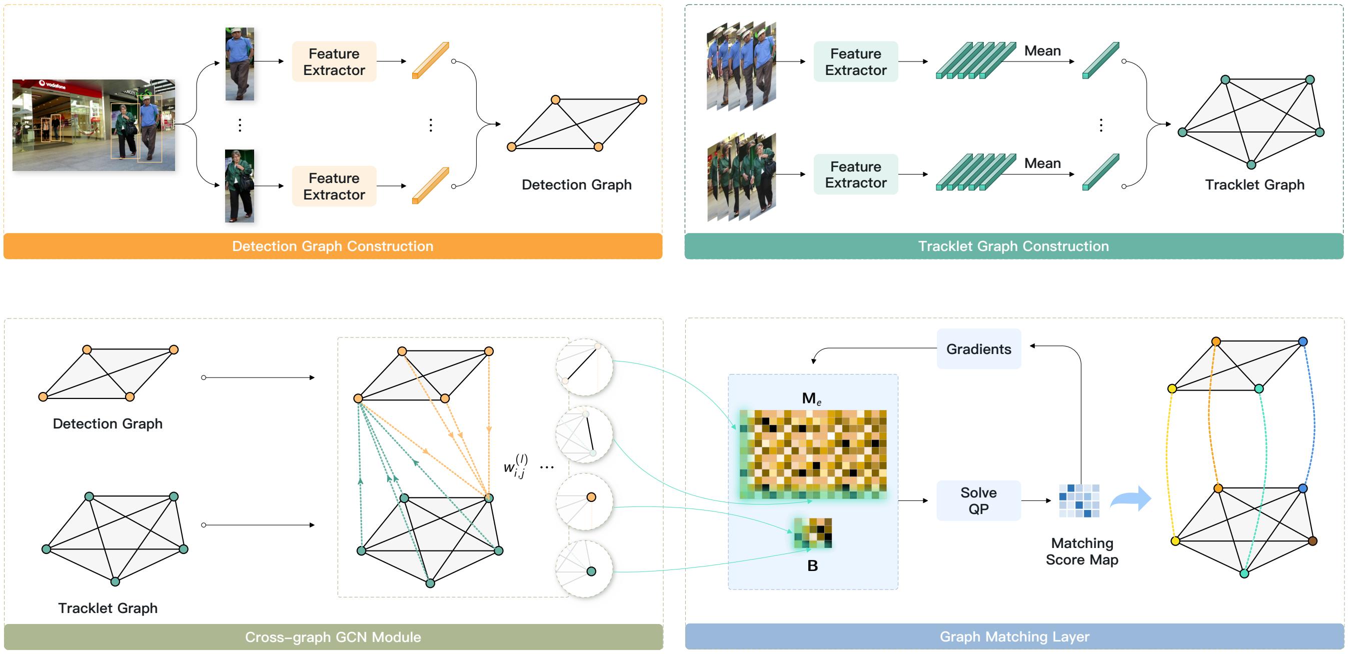


Fig. 3: Overview of our GMTracker method. We first extract features from detections and construct the detection graph using these features. The tracklet graph construction step is similar to the detection graph, but we average the features in a tracklet. Then the cross-graph GCN is adopted to enhance 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 $\mathbf{M}_e$ and $\mathbf{B}$ in the graph matching layer denote the edge affinity matrix from Eq. 7 and the vertex affinity matrix from Eq. 11 respectively.

So the Lagrangian is given by

$$L(x,\nu,\lambda) = \frac{1}{2}x^\top Qx + \lambda^\top(Gx-h) + q^\top x + \nu^\top(Ax-b), \quad (17)$$

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:

$$\begin{aligned} \nabla_x L(x^*, \lambda^*, \nu^*) &= 0 \\ Qx^* + q + A^\top \nu^* + G^\top \lambda^* &= 0 \\ Ax^* - b &= 0 \\ \text{diag}(\lambda^*)(Gx^* - h) &= 0 \\ Gx^* - h &\leq 0 \\ \lambda^* &\geq 0. \end{aligned} \quad (18)$$

We define the function

$$g(x,\lambda,\nu,\theta) = \begin{bmatrix} \nabla_x L(x,\lambda,\nu,\theta) \\ \text{diag}(\lambda)\lambda^\top(G(\theta)x - h(\theta)) \\ A(\theta)x - b(\theta) \end{bmatrix}, \quad (19)$$

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 [55], the gradients where the primal variable x and the dual variables ν and λ are the optimal solution, can be formulated as

$$J_\theta x^* = -J_x g(x^*, \lambda^*, \nu^*, \theta)^{-1} J_\theta g(x^*, \lambda^*, \nu^*, \theta), \quad (20)$$

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.

5.5 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 [16], we need to handle the born and death of tracklets. We keep the matching between detection and tracklet only if it satisfies all following constraints: 1) The appearance similarity between a detection and tracklet is above the threshold σ. 2) The detection is not far away from the tracklet. 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 pixel coordinates, called motion gate. 3) The detection bounding box overlaps with the position of tracklet predicted by the motion model. The constraints above can be written as

$$\begin{cases} \mathbf{B}_{i,j} > \sigma, \\ \texttt{KF}_{i,j} > \kappa, \\ \texttt{iou}_{i,j} > 0, \end{cases} \quad (21)$$

where, $(i.j) \in (\mathcal{V}_D, \mathcal{V}_T)$.

Here, besides the Kalman Filter adopted to estimate the geometric information in Section 5.2, we apply an Enhanced

Correlation Coefficient (ECC) [63] 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*.

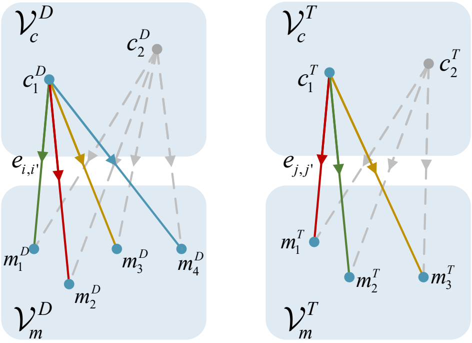


Fig. 4: An illustration of edge matching. Here, for matched pair (c_1^D, c_1^T) in $\boldsymbol{\pi}_c$, we find best matching between edge $e_{1,i'}$ and $e_{1,j'}$, drawn in the same color.

5.6 GST: A Practical Algorithm for Quadratic Assignment

However, due to the process of solving the quadratic programming, the inference speed is relatively slow compared with other mainstream MOT algorithms. To speed up, we design the gated search tree (GST) algorithm. Utilizing constraints Eq. 21, the feasible region is limited much smaller than the original quadratic programming, which greatly accelerates the process of solving the quadratic assignment problem.

The GST algorithm (Alg. 1) contains three main steps. Firstly, we construct a bipartite graph $\mathcal{G} = (\mathcal{V}_D, \mathcal{V}_T, \mathcal{E}_{DT})$, in which the edges are only between tracklets and detections meeting the constraints, i.e., $\mathcal{E}_{DT} = \{(i,j) | i \in \mathcal{V}_D, j \in \mathcal{V}_T, \mathbf{B}_{i,j} > \sigma, \texttt{KF}_{i,j} > \kappa, \texttt{iou}_{i,j} > 0\}$. Secondly, we use depth-first search to find all the connected components $\{\mathcal{G}_c\} = \{\mathcal{G}_c = (\mathcal{V}_c^D, \mathcal{V}_c^T, \mathcal{E}_c^{DT}) | c = (1, 2, \cdots, k), |\mathcal{V}_1^D| + |\mathcal{V}_1^T| \geq |\mathcal{V}_2^D| + |\mathcal{V}_2^T| \geq \cdots \geq |\mathcal{V}_k^D| + |\mathcal{V}_k^T|\}$. Last, we calculate the matching cost $\mathcal{L}(\boldsymbol{\pi}_c)$ for all matching candidates in each independent connected component $\mathcal{G}_c$ parallelly.

The matching cost follows the objective function of the quadratic programming Eq. 6. However, as a searching algorithm, the convex relaxation in the objective function shows no advantage. So, the matching cost can be denoted back to the objective function of original QAP, as

$$\mathcal{L}(\boldsymbol{\pi}) = -\boldsymbol{\pi}^\top \mathbf{M} \boldsymbol{\pi} - \mathbf{b}^\top \boldsymbol{\pi}. \tag{22}$$

To calculate the matching cost parallelly and reduce the computation in each independent connected component, we partition the quadratic affinity matrix $\mathbf{M}$ and vertex affinity matrix $\mathbf{B}$. We denote $\boldsymbol{\pi}^\top = [\boldsymbol{\pi}_c^\top, \boldsymbol{\pi}_m^\top]$, where $\boldsymbol{\pi}_c$ is in current connected component $\mathcal{G}_c$ and $\boldsymbol{\pi}_m$ is between the complement of the vertex sets, i.e., $\mathcal{G}_m = (\mathcal{V}_D \backslash \mathcal{V}_c^D, \mathcal{V}_T \backslash \mathcal{V}_c^T, \mathcal{E}_m^{DT})$. Then, the matching cost is

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\pi}_c) =& - \boldsymbol{\pi}^\top \mathbf{M} \boldsymbol{\pi} - \mathbf{b}^\top \boldsymbol{\pi} \\
=& - [\boldsymbol{\pi}_c^\top, \boldsymbol{\pi}_m^{*\top}] \begin{bmatrix} \mathbf{M}_c & \mathbf{M}_r \\ \mathbf{M}_l & \mathbf{M}_m \end{bmatrix} \begin{bmatrix} \boldsymbol{\pi}_c \\ \boldsymbol{\pi}_m^* \end{bmatrix} \\
& - [\mathbf{b}_c^\top, \mathbf{b}_m^\top] \begin{bmatrix} \boldsymbol{\pi}_c \\ \boldsymbol{\pi}_m^* \end{bmatrix}, \\
=& - [\boldsymbol{\pi}_c^\top, \boldsymbol{\pi}_m^{*\top}] \begin{bmatrix} \mathbf{M}_c & \mathbf{M}_r \\ \mathbf{M}_l & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\pi}_c \\ \boldsymbol{\pi}_m^* \end{bmatrix} - \mathbf{b}_c^\top \boldsymbol{\pi}_c, \\
=& - \boldsymbol{\pi}_c^\top \mathbf{M}_c \boldsymbol{\pi}_c - 2\boldsymbol{\pi}_m^{*\top} \mathbf{M}_l \boldsymbol{\pi}_c - \mathbf{b}_c^\top \boldsymbol{\pi}_c, \\
=& - \boldsymbol{\pi}_c^\top \mathbf{M}_c \boldsymbol{\pi}_c + 2\mathcal{L}_e(\boldsymbol{\pi}_c, \boldsymbol{\pi}_m^*) - \mathbf{b}_c^\top \boldsymbol{\pi}_c,
\end{aligned} \tag{23}$$

where $\mathbf{M}_c \in \mathbb{R}^{|\mathcal{V}_c^D| \times |\mathcal{V}_c^T|}, \mathbf{b}_c \in \mathbb{R}^{|\mathcal{V}_c^D||\mathcal{V}_c^T|}$. Here, the matching cost contains the pairwise cost $\mathcal{L}_e(\boldsymbol{\pi}_c, \boldsymbol{\pi}_m^*)$ that depends on the optimal solution $\boldsymbol{\pi}_m^*$, not available from connected component $\mathcal{G}_c$. To make it independent of the other components, we only consider the optimal solution $\widetilde{\boldsymbol{\pi}}_m^*$ given $\boldsymbol{\pi}_c$ instead the global optimal solution $\boldsymbol{\pi}_m^*$, i.e.,

$$\mathcal{L}_e(\boldsymbol{\pi}_c, \boldsymbol{\pi}_m^*) \approx \mathcal{L}_e(\widetilde{\boldsymbol{\pi}}_m^* | \boldsymbol{\pi}_c) = -\max_{\boldsymbol{\pi}_m} \boldsymbol{\pi}_m^\top \mathbf{M}_l \boldsymbol{\pi}_c. \tag{24}$$

Intuitively, it is to find the best matching between the edges in detection graph and tracklet graph with the start vertices fixed to an existing set of matches $\boldsymbol{\pi}_c$. As shown in Fig. 4, for each matched pair (c_i^D, c_j^T) in $\boldsymbol{\pi}_c$, we adopt bipartite matching between edge set $\{e_{i,i'}\}$ and $\{e_{j,j'}\}$ that start from c_i^D and c_j^T respectively.

Algorithm 1: Gated Search Tree (GST)

```
Input: $\mathbf{M}, \mathbf{B}, \texttt{iou}, \texttt{KF}, \sigma, \kappa$
Output: $\mathbf{\Pi}$
// Construct graph
for $(i.j) \in \texttt{range}(n_d, n_t)$ do
    $\mathbf{A}_{i.j} \leftarrow \mathbb{I}\{\texttt{iou}_{i,j} > 0 \wedge \texttt{KF}_{i,j} > \kappa \wedge \mathbf{B}_{i,j} > \sigma\}$
// Find Independent Connected Components (Alg. 2)
$\{\mathcal{G}_k\} \leftarrow \texttt{FICC}(\mathcal{G}(\mathbf{A}))$
// Find Best Matching
for $\mathcal{G}_c \in \{\mathcal{G}_k\}$ do
    $\mathbf{M}_c, \mathbf{b}_c = \mathbf{M}[\{c\}, \{c\}], \texttt{vec}(\mathbf{B}[\{c\}, \{c\}])$
    $\mathcal{L}(\boldsymbol{\pi}_c) = -\boldsymbol{\pi}_c^\top \mathbf{M}_c \boldsymbol{\pi}_c + 2\mathcal{L}_e(\widetilde{\boldsymbol{\pi}}_m^* | \boldsymbol{\pi}_c) - \mathbf{b}_c^\top \boldsymbol{\pi}_c,$
    $\boldsymbol{\pi}_c^* \leftarrow \arg\min_{\boldsymbol{\pi}_c} \mathcal{L}(\boldsymbol{\pi}_c)$
$\mathbf{\Pi} \leftarrow \bigcup_{c=1}^k \boldsymbol{\pi}_c^*$
return $\mathbf{\Pi}$
```

Then, we discuss the time cost of the original QP solver and the GST algorithm:

Proposition 5.1 (Original complexity). The quadratic programming Eq. 6 can be solved in $O(n_d^3 n_t^3)$ arithmetic operations.

Proposition 5.2. The running time of Algorithm 1 in parallel mode is

$$T = c \cdot n_m |\mathcal{V}_1^T||\mathcal{V}_1^D| + \epsilon, \tag{25}$$

where c is a constant factor, ϵ represents low-order terms of n and communication overhead between threads, $n_m = \max\{n_d, n_t\}$.

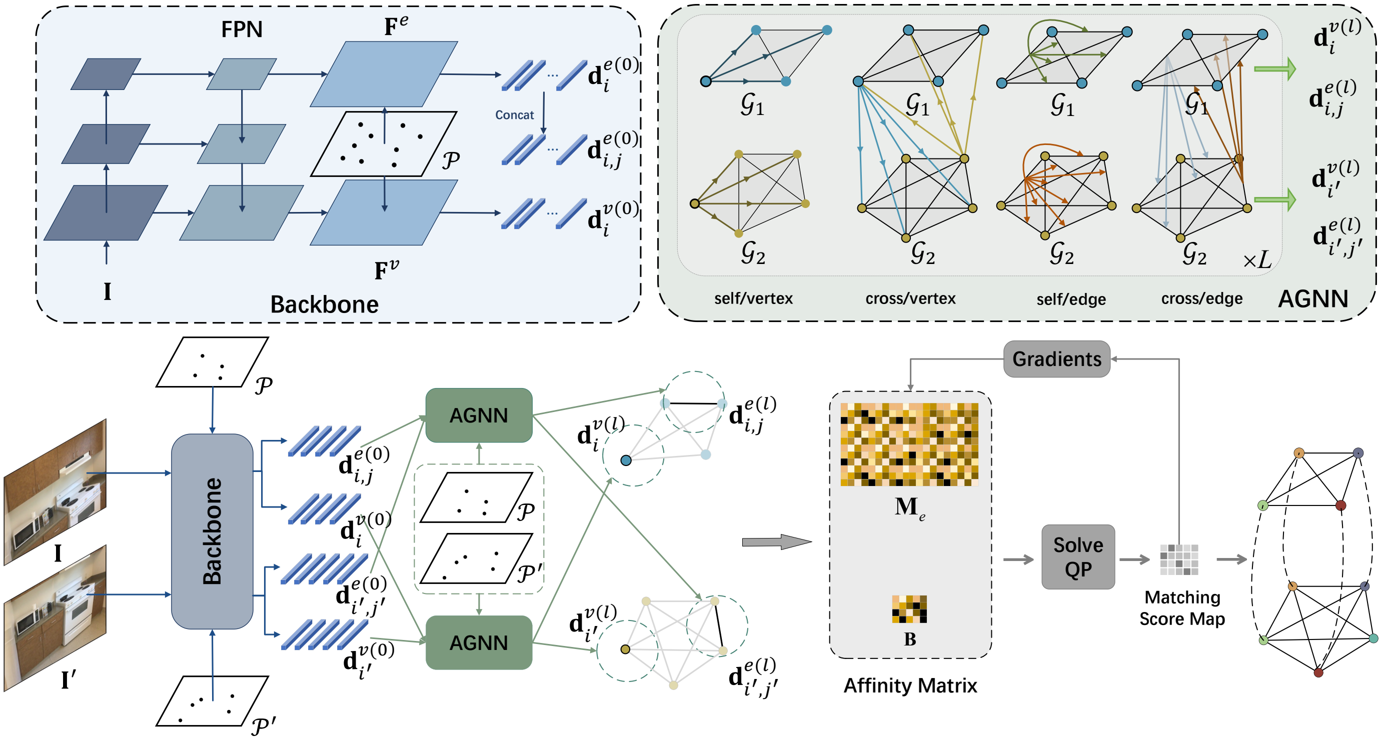


Fig. 5: Pipeline of our image matching network, GMatcher. The backbone is an FPN-like module. The edge and vertex features are from stride-8 and stride-2 feature map respectively. Edge and vertex AGNN are operated independently. The learnable graph matching layer replaces the Sinkhorn layer in SuperGlue.

Algorithm 2: Find Independent Connected Components (FICC)

```
Input: $\mathcal{G} = (\mathcal{V}_D, \mathcal{V}_T, \mathcal{E}_{DT})$
Output: $\{\mathcal{G}_c\}$
$c \leftarrow 0$; $\{\mathcal{G}_c\} \leftarrow \varnothing$
for $v_p \in \mathcal{V}_D \cup \mathcal{V}_T$ do
    visited$[v_p] \leftarrow$ False
for $v_p \in \mathcal{V}_D \cup \mathcal{V}_T$ do
    if $\neg$visited$[v_p]$ then
        $\mathcal{E}_c^{DT}, \mathcal{V}_c^D, \mathcal{V}_c^T \leftarrow \varnothing$
        if $v_p \in \mathcal{V}_D$ then
            $\mathcal{V}_c^D \leftarrow \mathcal{V}_c^D \cup \{v_p\}$
        else if $v_p \in \mathcal{V}_T$ then
            $\mathcal{V}_c^T \leftarrow \mathcal{V}_c^T \cup \{v_p\}$
        visit$(v_p)$
        $c \leftarrow c + 1$
def visit$(v_p)$:
    visited$[v_p] \leftarrow$ True
    for $e_{p,q} \in \mathcal{E}_{DT}$ do
        $\mathcal{E}_c^{DT} \leftarrow \mathcal{E}_c^{DT} \cup \{e_{p,q}\}$
        if $\neg$visited$[v_q]$ then
            visit$(v_q)$
            if $v_q \in \mathcal{V}_D$ then
                $\mathcal{V}_c^D \leftarrow \mathcal{V}_c^D \cup \{v_q\}$
            else if $v_i \in \mathcal{V}_T$ then
                $\mathcal{V}_c^T \leftarrow \mathcal{V}_c^T \cup \{v_q\}$
return $\{\mathbf{G_c}\}$
```

6 Learnable Graph Matching for Image Matching Task

Besides the MOT task, our learnable graph matching method can be easily adapted to other data association tasks with slight modifications. In this section, we take the image matching task as an example. We formulate the image matching task as a graph matching problem between the keypoints in two images and utilize our learnable graph matching algorithm to build an end-to-end keypoint-based neural network.

6.1 Problem Formulation

Given keypoints $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_m\}$, $\mathcal{P}' = \{\mathbf{p}'_1, \mathbf{p}'_2, \cdots, \mathbf{p}'_n\}$ on the image I and I' of the same scene respectively, where $\mathbf{p}_i = (x_i, y_i)$ is the keypoint position in image coordinates, the image matching task is to find the best matching between $\mathcal{P}$ and $\mathcal{P}'$ and thus estimate the reletive camera pose $\mathbf{T} \in SE(3)$. The keypoints are often on the corners, textured areas, or the boundary of the objects, where the local features are relatively robust and less affected by illumination and viewing angle. They can be derived from traditional methods, like SIFT [20], or deep learning based methods, like SuperPoint [21]. And the descriptor $\mathbf{d}_i \in \mathbb{R}^c$ is a c-dimentional local discriminative feature, corresponding to the keypoint $\mathbf{p}_i$. In this paper, we extract the features in an end-to-end way, with only positions of the keypoints are from the off-the-shelf neural network.

In our end-to-end graph matching neural network, called *GMatcher*, we take two images and the keypoints on each image as the input, solving the graph matching problem (Eq. 6) from $\mathcal{P}$ and $\mathcal{P}'$, and we finally obtain the assignment matrix $\mathbf{\Pi} \in \mathbb{R}^{m \times n}$ to represent the matching between two keypoint sets.

6.2 End-to-end Graph Matching Network for Image Matching

Our method is mainly based on SuperGlue [19], which utilizes the attentional graph neural network (AGNN) module to aggregate long-range dependencies. However, compared with graph matching, although SuperGlue stacks self and cross attention module many times to fuse the intra- and inter-image information, it does not explicitly define the intra-image graph and consider the edge similarities between images in keypoint matching.

To better utilize the high-order information, i.e., edge in the intra-image graph, we use FPN-like [5] backbone to extract multiscale features. The stride-2 feature map and stride-8 feature map are used to extract vertex feature $\mathbf{d}_i^v$ and edge feature $\mathbf{d}_{i,j}^e$ respectively. Note that the edge feature $\mathbf{d}_{i,j}^e$ is the concatenation of the endpoints' feature $\mathbf{d}_i^e$ and $\mathbf{d}_j^e$ on stride-8 feature map. The feature $\mathbf{d}_i$ on each keypoint $\mathbf{p}_i$ is extracted from the feature map that is restored to the original image resolution using bilinear interpolation.

Like the position embedding in transformer, we use MLP to encode the position information into the vertex feature $\mathbf{d}_i^v$ and edge feature $\mathbf{d}_{i,j}^e$, i.e.,

$$\begin{aligned} \mathbf{f}_i^v &= \mathbf{d}_i^v + \mathrm{MLP}_{\mathrm{pos}}(\mathbf{p}_i), \\ \mathbf{f}_{i,j}^e &= \mathbf{d}_{i,j}^e + [\mathrm{MLP}_{\mathrm{pos}}(\mathbf{p}_i), \mathrm{MLP}_{\mathrm{pos}}(\mathbf{p}_j)], \end{aligned} \tag{26}$$

where, $[\cdot, \cdot]$ denotes concatenation.

Then, similar to the AGNN module in SuperGlue, we conduct self-attentional and cross-attentional message passing for l times in vertex features and edge features separately. The detailed design can be referred to SuperGlue [19]. We use the output vetex features and edge features of AGNN to construct the final graphs $\mathcal{G}_1 = (\mathcal{V} = \{\mathbf{d}_i^{v(l)}\}, \mathcal{E} = \{\mathbf{d}_{i,j}^{e(l)}\}))$ and $\mathcal{G}_2 = (\mathcal{V} = \{\mathbf{d}_{i'}^{v(l)}\}, \mathcal{E} = \{\mathbf{d}_{i',j'}^{e(l)}\}))$ for two images and matching with our differentiable graph matching layer mentioned in Sec. 5.3.

7 Experiments on MOT task

7.1 Datasets and Evaluation Metrics

We carry out the experiments on MOT16 [64] and MOT17 [64] 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) [65] and ID F1 Score (IDF1) [66] 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. Moreover, a newly proposed metric Higher Order Tracking Accuracy (HOTA) [67], emphasizing the balance of object detection and association, becomes one of the official metrics of MOT benchmarks.

7.2 Implementation Details

Training. Following other MOT methods [10], [12], we adopt Tracktor [68] to refine the public detections. We use a ResNet50 [69] backbone followed by a global average pooling layer and a fully connected layer with 512 channels, as the ReID network used for feature extraction. The output ReID features are further normalized with the l_2 normalization. We pre-train the ReID network on Market1501 [70], DukeMTMC [66] and CUHK03 [71] datasets jointly. 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. All the ReID network training settings follows MPNTrack [10]. Our implementation is based on PyTorch [72] framework. We train our model on an NVIDIA RTX 2080Ti GPU. Adam [73] 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 10^{-3}.

Inference. Our inference pipeline mostly follows DeepSORT [16], 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 [10], [12]. This effectively reduces the false negatives introduced by upstream object detection algorithm.

7.3 Ablation Study

We conduct ablation studies of the proposed components in our method on the MOT17 dataset. Following [10], 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 information (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 based on Hungarian Algorithm, our method without training gets a gain of 1.9 IDF1, and a gain of 2.7 IDF1 and 1.1 MOTA with the linear interpolation. The results show the effectiveness of the second-order information in the graph.

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

GM	App. Enc.	GCN	Geo	Inter.	IDF1 ↑	MOTA ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw. ↓
					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.

Methods	IDF1	MOTA
Last Frame	64.3	62.2
Moving Average $\alpha = 0.2$	69.8	62.4
Moving Average $\alpha = 0.5$	70.0	62.4
Moving Average $\alpha = 0.8$	70.6	62.4
Mean	71.5	62.3

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

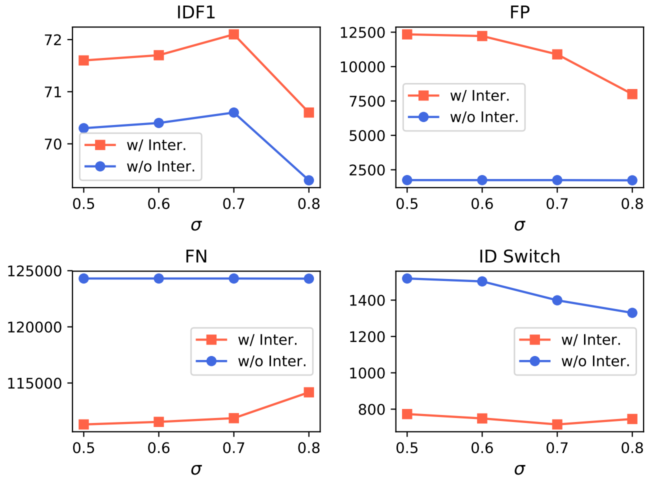


Fig. 6: Results on IDF1, FP, FN and ID Switch metrics under different threshold σ of the feature similarity to create a new tracklet.

and MOTA metric, respectively. With interpolation, the gain becomes even larger: about 4.1 improvements on IDF1 and 0.9 on MOTA.

Table 2 shows the effectiveness of our differentiable graph matching layer the importance of training all components in our tracker jointly. 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.

7.4 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. Besides, the oracle experiment shows the upper bound performance of our learnable graph matching method.

Max age (frames)	30	50	80	100	150
DeepSORT	67.2	67.9	68.1	68.1	67.3
Graph Matching	69.2	70.5	71.4	71.5	71.8

TABLE 4: The influence of *max age* δ on IDF1.

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

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 using the last frame of the tracklet. The results are shown in Table 3. The IDF1 is 7.2 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 born and death strategies. In most of the online tracking methods, one of the core strategies is how to create and delete a tracklet. In our GMTracker, we mostly follow DeepSORT, but we also make some improvements to make these strategies more suitable for our approach, as described in Section 5.5. Among the three criteria to create a new

Methods	FPS*	Solver (s)	IDF1	MOTA	IDS
GMTracker	0.987	8861	71.5	62.3	1017
GMTracker+GST	20.7 (21×)	93.1 (95×)	71.4	62.3	935

TABLE 6: Inference speed comparison between GMTracker and GMTracker+GST on MOT17 *val* set. * Running time of the off-the-shelf object detection algorithm is not considered.

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. 6. 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 in our method, that larger *max age* δ, 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 DeepSORT, which indicates our method can deal with long-term tracklet associations better.

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

7.5 Inference Time

We compare the running speed between our new GST algorithm and the original quadratic programming solved by CVXPY library, as shown in Table 6. Using the new GST algorithm, with the performance under main metrics almost the same, the speed is about 21× faster than the original GMTracker, and the solver running time is two orders of magnitude lower than before. Note that the ID switches are much fewer because the matching candidates are filtered before solving the quadratic assignment problem, and the phenomenon of early termination of the tracklet is somewhat alleviated.

7.6 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 8, when we apply Tracktor [68] 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 [79], 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 8, we also outperform the other *offline* state-of-the-art trackers on IDF1. With exactly the same visual inter- and extrapolation as LifT [12], called GMT_VIVE in Table 8, 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.

Keypoint detector	Matcher	Pose estimation AUC			P	MS
		@5°	@10°	@20°		
SuperPoint [21]	NN+mutual	9.43	21.53	36.40	50.4	18.8
	NN+GMS [74]	8.39	18.96	31.56	50.3	19.0
	NN+PointCN [75]	11.40	25.47	41.41	71.8	25.5
	NN+OANet [76]	11.76	26.90	43.85	74.0	25.7
	SuperGlue [19]	16.16	33.81	51.84	84.4	31.5
	SuperGlue* [19]	-	-	53.38	-	-
	GMatcher (Ours)	**17.43**	**35.75**	**54.54**	**84.9**	**31.6**

TABLE 7: Comparisons with SOTA keypoint-based methods on ScanNet. * denotes end-to-end training.

7.7 Qualitative results on MOT17

In Fig. 7, we show the hard cases, in which the baseline tracker *DeepSORT* has ID switches and our *GMTracker* tracks the objects with right IDs. For example, in Fig. 7a, DeepSORT fails to track the person with ID-2 and create a new tracklet ID-38, because the person is occluded by the streetlight and reappears. And in Fig. 7b, the people with ID-19 and ID-21 exchange their places. DeepSORT can not keep the IDs. The ID-19 drifts to the person with ID-21 and a new ID is assigned to the person with ID-21. However, our GMTracker tracks the objects with right IDs.

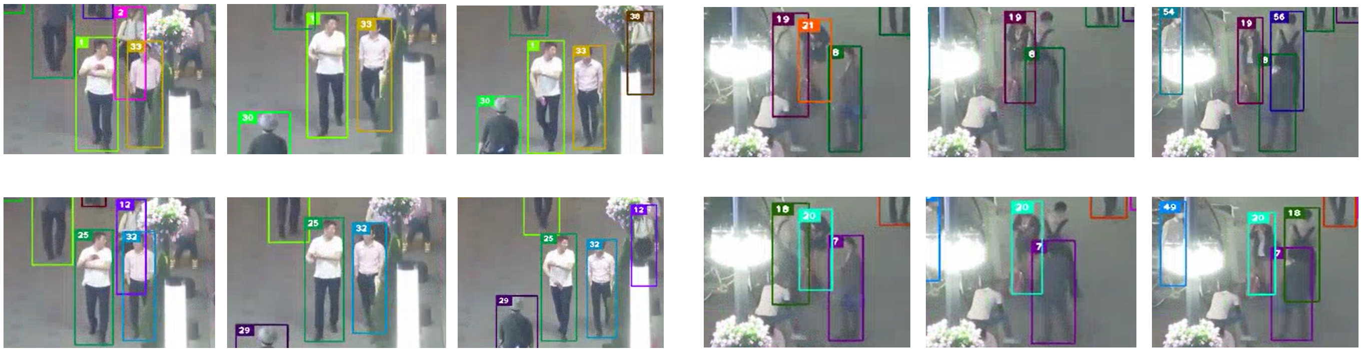
8 Experiments on Image Matching task

8.1 Datasets and Evaluation Metrics

We do the experiments about Image Matching task on the mainstream indoor camera pose estimation dataset ScanNet [22]. ScanNet is a large-scale indoor dataset collected using RGB-D cameras. Because it is an indoor dataset, local patterns are always similar in a large area, such as on the white wall and ceramic tile. In image matching task, the general evaluation metrics focus on the camera pose estimation performance, such as Area Under Curve (AUC) of the pose error with thresholds at $5^\circ, 10^\circ, 20^\circ$. Here, the pose error takes the maximum angle error of the rotation matrices and the transformation vectors. Besides, the match precision and the matching score are calculated, mainly reflecting the keypoint matching performance.

8.2 Implementation Details

We take state-of-the-art keypoint-based image matching pipeline SuperGlue [19] as our baseline. Our implementation uses PyTorch [72] framework, and we train our model on 24 NVIDIA RTX 2080Ti GPUs. Adam [73] optimizer is applied, and the base learning rate is 4×10^{-5} with Cosine Annealing as the learning rate scheduler. Like SuperGlue, we only sample the image pairs with overlap in $[0.4, 0.8]$ for training. The images and the depth maps are resized to the resolution of 640×480. The batch size is 24 and train



(a) Detection misses and resurfaces by occlusion.

(b) Two objects exchange their locations.

Fig. 7: Examples of tracking results on MOT17 dataset. The top line is from *DeepSORT*, and the bottom is from *GMTracker*. (a) and (b) are two typical hard cases, in which our method is better than the baseline *DeepSORT*.

Methods	Refined Det	IDF1 ↑	HOTA↑	MOTA ↑	MT↑	ML↓	FP ↓	FN ↓	IDS ↓	AssA↑	DetA↑	LocA↑
MOT17												
Tracktor++ (O) [68]	Tracktor	52.3	42.1	53.5	19.5	36.6	12201	248047	2072	41.7	42.9	80.9
Tracktor++v2 (O) [68]	Tracktor	55.1	44.8	56.3	21.1	35.3	8866	235449	1987	45.1	44.9	81.8
GNNMatch (O) [77]	Tracktor	56.1	45.4	57.0	23.3	34.6	12283	228242	1957	45.2	45.9	81.5
GSM_Tracktor (O) [78]	Tracktor	57.8	45.7	56.4	22.2	34.5	14379	230174	**1485**	47.0	44.9	80.9
CTTrackPub (O) [79]	CenterTrack	59.6	48.2	61.5	**26.4**	**31.9**	14076	200672	2583	47.8	49.0	81.7
BLSTM-MTP-T (O) [80]	Tracktor	60.5	-	55.9	20.5	36.7	**8663**	238863	1188	-	-	-
TADAM (O) [81]	Tracktor	58.7	-	59.7	-	-	9676	216029	1930	-	-	-
ArTIST-C (O) [82]	CenterTrack	59.7	48.9	**62.3**	29.1	34.0	19611	191207	2062	48.3	**50.0**	81.4
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**
MPNTrack [10]	Tracktor	61.7	49.0	58.8	28.8	33.5	17413	213594	1185	51.1	47.3	81.5
Lif_TsimInt [12]	Tracktor	65.2	50.7	58.2	28.6	33.6	16850	217944	**1022**	54.9	47.1	81.5
LifT [12]	Tracktor	65.6	51.3	60.5	27.0	33.6	14966	206619	1189	54.7	48.3	81.3
LPC_MOT [83]	Tracktor	66.8	51.5	59.0	29.9	33.9	23102	206948	1122	56.0	47.7	80.9
ApLift [84]	Tracktor	65.6	51.1	60.5	**33.9**	**30.9**	30609	190670	1709	53.5	49.1	80.7
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) [68]	Tracktor	54.9	44.6	56.2	20.7	35.8	2394	76844	617	44.6	44.8	82.0
GNNMatch (O) [77]	Tracktor	55.9	44.6	56.9	22.3	35.3	3235	74784	564	43.7	45.8	81.7
GSM_Tracktor (O) [78]	Tracktor	58.2	45.9	57.0	22.0	34.5	4332	73573	**475**	46.7	45.4	81.1
TADAM (O) [81]	Tracktor	59.1	-	59.5	-	-	2540	71542	529	-	-	-
ArTIST-C (O) [82]	CenterTrack	61.9	49.8	**63.0**	29.1	33.2	7,420	59,376	635	49.5	**50.6**	81.0
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
MPNTrack [10]	Tracktor	61.7	48.9	58.6	27.3	34.0	4949	70252	354	51.1	47.1	81.7
Lif_TsimInt [12]	Tracktor	64.1	49.6	57.5	25.4	34.7	**4249**	72868	**335**	53.3	46.5	**81.9**
LifT [12]	Tracktor	64.7	50.8	61.3	27.0	34.0	4844	65401	389	53.1	48.9	81.4
LPC_MOT [83]	Tracktor	67.6	51.7	58.8	27.3	35.0	6167	68432	435	56.4	47.6	81.3
ApLift [84]	Tracktor	66.1	51.3	61.7	**34.3**	31.2	9168	60180	495	53.2	49.8	80.7
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 8: Detailed comparison with state-of-the-art methods on MOT16 and MOT17 *test* set. (O) denotes online methods. (O*) denotes near-online methods.

the model for 1.2M iterations. In the training stage, only 200 pairs are sampled randomly for each scene in an epoch. During the inference stage, the mutual constraint and the matching score threshold are adopted to filter the mismatch and the threshold is set to 0.2. Other experiment settings not written out here are the same as SuperGlue.

8.3 Comparisons with SOTA keypoint-based methods

In Table 7, we show the comparison with other SOTA methods using the same SuperPoint [21] keypoint detector on ScanNet test set, which contains 1500 image pairs from [19].Note that we use about half training data and training iterations to outperform SOTA method SuperGlue [19] by 1 pose estimation AUC.

9 Conclusion

In this paper, we propose a novel learnable graph matching method for data association. Our graph matching method focuses on the pairwise relationship within the view. 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. We apply our method to Multiple Object Tracking task, called GMTracker. Taking the second-order edge-to-edge similarity into account, our tracker is more accurate and robust in the MOT task. To speed up our algorithm, we design GST to shrink the area of the feasible region. The experiments of ablation study and comparison with other state-of-the-art methods both show the efficiency and effectiveness of our method. Moreover, for Image Matching task, we propose the end-to-end learnable graph matching algorithm based on the SOTA method SuperGlue, caleed GMatcher, which shows our ability to solve data association tasks generally. The experiments show that we only use half training data and training iterations to outperform SuperGlue by about 1 AUC.

References

[1] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[2] E. L. Lawler, "The quadratic assignment problem," *Management Science*, vol. 9, no. 4, pp. 586–599, 1963.

[3] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica*, vol. 25, no. 1, pp. 53–76, 1957.

[4] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *ICML*, 2017.

[5] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Int. Conf. Comput. Vis.*, 2017.

[7] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1806–1819, 2011.

[8] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *IEEE Int. Conf. Image Process.*, 2016.

[9] S. Wang and C. C. Fowlkes, "Learning optimal parameters for multi-target tracking with contextual interactions," *Int. J. Comput. Vis.*, vol. 122, no. 3, pp. 484–501, 2017.

[10] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[11] Y. Xu, A. Osep, Y. Ban, R. Horaud, L. Leal-Taixé, and X. Alameda-Pineda, "How to train your deep multi-object tracker," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[12] A. Hornakova, R. Henschel, B. Rosenhahn, and P. Swoboda, "Lifted disjoint paths with application in multiple object tracking," in *ICML*, 2020.

[13] C.-H. Kuo and R. Nevatia, "How does person identity recognition help multi-person tracking?" in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2011.

[14] B. Yang and R. Nevatia, "An online learned CRF model for multi-target tracking," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2012.

[15] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler, "Learning by tracking: Siamese CNN for robust target association," in *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2016.

[16] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *IEEE Int. Conf. Image Process.*, 2017.

[17] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, "Online multi-object tracking with dual matching attention networks," in *Eur. Conf. Comput. Vis.*, 2018.

[18] S. Sun, N. Akhtar, H. Song, A. S. Mian, and M. Shah, "Deep affinity network for multiple object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 104–119, 2019.

[19] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[20] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[21] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2018.

[22] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[23] Y. Bar-Shalom, T. E. Fortmann, and P. G. Cable, "Tracking and data association," 1990.

[24] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[25] S. Hamid Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid, "Joint probabilistic data association revisited," in *Int. Conf. Comput. Vis.*, 2015.

[26] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *Int. Conf. Comput. Vis.*, 2015.

[27] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multicamera people tracking with a probabilistic occupancy map," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 267–282, 2007.

[28] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2008.

[29] B. Yang, C. Huang, and R. Nevatia, "Learning affinities and dependencies for multi-target tracking using a CRF model," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2011.

[30] A. R. Zamir, A. Dehghan, and M. Shah, "GMCP-Tracker: Global multi-object tracking using generalized minimum clique graphs," in *Eur. Conf. Comput. Vis.*, 2012.

[31] S. Tang, B. Andres, M. Andriluka, and B. Schiele, "Subgraph decomposition for multi-target tracking," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015.

[32] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2011.

[33] S. Tang, B. Andres, M. Andriluka, and B. Schiele, "Multi-person tracking by multicut and deep matching," in *Eur. Conf. Comput. Vis.*, 2016.

[34] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," in *Int. Conf. Comput. Vis.*, 2015.

[35] B. Wang, G. Wang, K. L. Chan, and L. Wang, "Tracklet association by online target-specific metric learning and coherent dynamics estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 3, pp. 589–602, 2016.

[36] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *Int. Conf. Comput. Vis.*, 2017.

[37] J. Li, X. Gao, and T. Jiang, "Graph networks for multiple object tracking," in *WACV*, 2020, pp. 719–728.

[38] X. Jiang, P. Li, Y. Li, and X. Zhen, "Graph neural based end-to-end data association framework for online multiple-object tracking," *arXiv preprint arXiv:1907.05315*, 2019.

[39] W. Hu, X. Shi, Z. Zhou, J. Xing, H. Ling, and S. Maybank, "Dual L1-normalized context aware tensor power iteration and its applications to multi-object tracking and multi-graph matching," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 360–392, 2020.

[40] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Eur. Conf. Comput. Vis.*, 2006.

[41] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to sift or surf," in *Int. Conf. Comput. Vis.*, 2011.

[42] T. Tuytelaars and L. Van Gool, "Wide baseline stereo matching based on local, affinely invariant regions," in *Brit. Mach. Vis. Conf.* Citeseer, 2000.

[43] T. Sattler, B. Leibe, and L. Kobbelt, "Scramsac: Improving ransac's efficiency with a spatial consistency filter," in *Int. Conf. Comput. Vis.*, 2009.

[44] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-net: A trainable cnn for joint detection and

description of local features," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[45] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "LoFTR: Detector-free local feature matching with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8922–8931.

[46] M. Vento and P. Foggia, "Graph matching techniques for computer vision," in *Image Processing: Concepts, Methodologies, Tools, and Applications*, 2013, pp. 381–421.

[47] J. Hartmanis, *Computers and intractability: a guide to the theory of NP-completeness*, 1982.

[48] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Int. Conf. Comput. Vis.*, 2005.

[49] C. Schellewald and C. Schnörr, "Probabilistic subgraph matching based on convex relaxation," in *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2005.

[50] P. H. Torr, "Solving markov random fields using semi definite programming." in *AISTATS*, 2003.

[51] P. Swoboda, C. Rother, H. Abu Alhaija, D. Kainmuller, and B. Savchynskyy, "A study of Lagrangean decompositions and dual ascent solvers for graph matching," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[52] F. Zhou and F. De la Torre, "Factorized graph matching," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2012.

[53] R. Wang, J. Yan, and X. Yang, "Learning combinatorial embedding networks for deep graph matching," in *Int. Conf. Comput. Vis.*, 2019.

[54] T. Yu, R. Wang, J. Yan, and B. Li, "Learning deep graph matching with channel-independent embedding and Hungarian attention," in *Int. Conf. Learn. Represent.*, 2020.

[55] S. Barratt, "On the differentiability of the solution to convex optimization problems," *arXiv preprint arXiv:1804.05098*, 2018.

[56] Y. Aflalo, A. Bronstein, and R. Kimmel, "On convex relaxation of graph isomorphism," *Proceedings of the National Academy of Sciences*, vol. 112, no. 10, pp. 2942–2947, 2015.

[57] A. Zanfir and C. Sminchisescu, "Deep learning of graph matching," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[58] C. Ma, Y. Li, F. Yang, Z. Zhang, Y. Zhuang, H. Jia, and X. Xie, "Deep association: End-to-end graph-based learning for multiple object tracking with conv-graph neural network," in *ICMR*, 2019.

[59] X. Weng, Y. Wang, Y. Man, and K. Kitani, "GNN3DMOT: Graph neural network for 3D multi-object tracking with 2D-3D multi-feature learning," *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[60] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[61] G. Li, C. Xiong, A. Thabet, and B. Ghanem, "DeeperGCN: All you need to train deeper GCNs," *arXiv preprint arXiv:2006.07739*, 2020.

[62] S. Diamond and S. Boyd, "CVXPY: A python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.

[63] G. D. Evangelidis and E. Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 10, pp. 1858–1865, 2008.

[64] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[65] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 319–336, 2008.

[66] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *Eur. Conf. Comput. Vis. Worksh.*, 2016.

[67] J. Luiten, A. Osep, P. Dendorfer, P. H. S. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "HOTA: A higher order metric for evaluating multi-object tracking," *Int. J. Comput. Vis.*, vol. 129, no. 2, pp. 548–578, 2021.

[68] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *Int. Conf. Comput. Vis.*, 2019.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[70] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Int. Conf. Comput. Vis.*, 2015.

[71] W. Li, R. Zhao, T. Xiao, and X. Wang, "DeepReID: Deep filter pairing neural network for person re-identification," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014.

[72] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inform. Process. Syst.*, 2019.

[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Int. Conf. Learn. Represent.*, 2014.

[74] J. Bian, W.-Y. Lin, Y. Matsushita, S.-K. Yeung, T.-D. Nguyen, and M.-M. Cheng, "GMS: Grid-based motion statistics for fast, ultra-robust feature correspondence," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[75] K. M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua, "Learning to find good correspondences," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[76] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, and H. Liao, "Learning two-view correspondences and geometry using order-aware network," in *Int. Conf. Comput. Vis.*, 2019.

[77] I. Papakis, A. Sarkar, and A. Karpatne, "GCNNMatch: Graph convolutional neural networks for multi-object tracking via sinkhorn normalization," *arXiv preprint arXiv:2010.00067*, 2020.

[78] Q. Liu, Q. Chu, B. Liu, and N. Yu, "GSM: Graph similarity model for multi-object tracking," in *IJCAI*, 2020.

[79] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *Eur. Conf. Comput. Vis.*, 2020.

[80] C. Kim, L. Fuxin, M. Alotaibi, and J. M. Rehg, "Discriminative appearance modeling with multi-track pooling for real-time multi-object tracking," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[81] S. Guo, J. Wang, X. Wang, and D. Tao, "Online multiple object tracking with cross-task synergy," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[82] F. Saleh, S. Aliakbarian, H. Rezatofighi, M. Salzmann, and S. Gould, "Probabilistic tracklet scoring and inpainting for multiple object tracking," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[83] P. Dai, R. Weng, W. Choi, C. Zhang, Z. He, and W. Ding, "Learning a proposal classifier for multiple object tracking," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[84] A. Hornakova, T. Kaiser, P. Swoboda, M. Rolinek, B. Rosenhahn, and R. Henschel, "Making higher order mot scalable: An efficient approximate solver for lifted disjoint paths," in *Int. Conf. Comput. Vis.*, 2021.

Jiawei He is a PhD student in BRAVE group of Center for Research on Intelligent Perception and Computing (CRIPAC), the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing, China, supervised by Prof. Zhaoxiang Zhang. Before this, he got his BS degree in automation from Xi'an Jiaotong University, China, in 2019. His research interests are in Computer Vision, Deep Learning, Learning-based Combinatorial Optimization, including video analysis, graph matching, multiple object tracking, 3D perception, etc.

Zehao Huang received the BS degree in automatic control from Beihang University, Beijing, China, in 2015. He is currently an algorithm engineer at TuSimple. His research interests include computer vision and image processing.

Naiyan Wang is currently the chief scientist of TuSimple. he leads the algorithm research group in the Beijing branch. Before this, he got his PhD degree from CSE department, HongKong University of Science and Technology in 2015. His supervisor is Prof. Dit-Yan Yeung. He got his BS degree from Zhejiang University, 2011 under the supervision of Prof. Zhihua Zhang. His research interest focuses on applying statistical computational model to real problems in computer vision and data mining. Currently, He mainly works on the vision based perception and localization part of autonomous driving. Especially He integrates and improves the cutting-edge technologies in academia, and makes them work properly in the autonomous truck.

Zhaoxiang Zhang received the bachelor's degree in circuits and systems from the University of Science and Technology of China (USTC) in 2004 and the Ph.D. degree from the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, China, in 2009. In October 2009, he joined the School of Computer Science and Engineering, Beihang University, and worked as an Assistant Professor from 2009 to 2011, an Associate Professor from 2012 to 2015, and the Vice-Director of the Department of Computer Application Technology from 2014 to 2015. In July 2015, he returned to the CASIA, to join as a Professor, where he is currently a Professor with the Center for Research on Intelligent Perception and Computing. He has published more than 200 papers in reputable conferences and journals. His major research interests include pattern recognition, computer vision, machine learning, and bio-inspired visual computing. He has won the best paper awards in several conferences and championships in international competitions. He has served as the Area Chair and a Senior PC for many international conferences, such as CVPR, ICCV, AAAI, and IJCAI. He has served or is serving as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, Pattern Recognition, and Neurocomputing.