Tracklet Association Tracker: An End-to-End Learning-based Association Approach for Multi-Object Tracking

Han Shen
IIIS, Tsinghua Univeristy
Beijing, China
shenh15@mails.tsinghua.edu.cn

Lichao Huang
Horizon Robotics
Beijing, China
lichao.huang@hobot.cc

Chang Huang
Horizon Robotics
Beijing, China
chang.huang@hobot.cc

Wei Xu
IIIS, Tsinghua Univeristy
Beijing, China
weixu@mail.tsinghua.edu.cn

Abstract

Traditional multiple object tracking methods divide the task into two parts: affinity learning and data association. The separation of the task requires to define a hand-crafted training goal in affinity learning stage and a hand-crafted cost function of data association stage, which prevents the tracking goals from learning directly from the feature. In this paper, we present a new multiple object tracking (MOT) framework with data-driven association method, named as Tracklet Association Tracker (TAT). The framework aims at gluing feature learning and data association into a unity by a bi-level optimization formulation so that the association results can be directly learned from features. To boost the performance, we also adopt the popular hierarchical association and perform the necessary alignment and selection of raw detection responses. Our model trains over $20\times$ *faster than a similar approach, and achieves the state-of-the-art performance on both MOT2016 and MOT2017 benchmarks.*

1. Introduction

Multiple Object Tracking (MOT) is one of the most critical middle-level computer vision tasks with wide-range applications such as visual surveillance, sports events, and robotics. Owing to the great success of object detection techniques, detection based paradigm dominates the community of MOT. The critical components of the paradigm include an *affinity model* telling how likely two objects belong to a single identity, and a *data association method* that links objects across frames, based on their affinities, so as to form a complete trajectory for each identity.

Tracklet-based association is a well-accepted approach in detection-based MOT [16, 36, 34, 31]. It is usually constructed by two stages: In stage I, we link detection responses in the adjacent frame using straightforward strategies to form short tracklets. In stage II, we mainly perform two tasks: extract much finer features from the tracklets, including temporal and spatial, appearance and motion data to construct a tracklet-level affinity model, and then perform graph-based association across all of them, and conduct necessary post-processing. There are two advantages of this approach, compared to associations on detection responses directly. With tracklet-based association, the number of connected components is significantly brought down so that investigating detection dependency across distant frames is computationally affordable. Besides, it is capable of extracting high-level information, while reducing bounding box noises brought by bad detectors [16].

There are various ways to define the affinity model in stage I, like bounding box intersection-over-union(IOU), spatial-temporal distance, appearance similarity, *etc*. The harder part exists in stage II. For the affinity model, traditional hand-crafted features or individually learned affinities do not work well [16, 37], due to the lack of data-driven properties in joint consideration of multiple correlated association choices. For the association, it is regular to use a global optimization algorithm, such as linear programming or network flow, to link these short tracklets. However, it is non-trivial to define a proper cost function for these approaches. Earlier trackers use hand-crafted cost functions and perform an inference afterward. Sometimes, they have to use grid search and empirical tuning to find the hyper-parameters producing the best outcome.

Recently, deep learning has shown its powerful learning capability in feature extraction. It outperforms almost all hand-crafted feature descriptors, such as HOG, SIFT, *etc*. Large-scale data provides nutrients for the learning of large models, and data-driven approaches are becoming rather important. However, for MOT, the ultimate goal, like MOTA, is not directly related to the features of objects. Thus, it is necessary to model the connectivity across frames into an association method, so that we can build a bridge from the features to the association goal, and perform learning using optimization method. Because network flow is an approach which can be solved in polynomial time, it has a great potential for data-driven learning comparing to other NP-hard formulations [31, 32, 33, 36]. Schulter *et al.* [29] propose a network flow based novel framework to handle both tasks of stage II in an *end-to-end* fashion: by back-propagating a constrained linear programming objective function. While the framework allows learning from detection features and auto-tuning of costs, the drawbacks are clear: 1) the large number of detection responses limit the expansion of window size; 2) the unbounded costs are easily diverging, and training is slow; 3) High-level motion information is not considered.

We propose Tracklet Association Tracker (TAT), an improved bi-level optimization framework compared to work of Schulter *et al.* [29] in three key aspects. First, we use deep metric learning to extract the appearance embedding for each detection response; Second, we introduce tracklet to the framework, not only accelerating the computation but also provides motion dependency. Last but not the least, we adopt an approximate gradient that significantly improves the association model training process. By clarifying the boundary of cost values, the framework ensures convergence can always be achieved and includes all cost parameters into the end-to-end training process while retaining high accuracy.

All in all, our contributions include:

- We introduce tracklet association into the bi-level optimization framework. By exploiting tracklets, our system improves the performance on long time occlusion.

- We implement TAT, an approximate network flow learning approach that provides a more stable and faster(over $20\times$) solution of similar method [29]. The method achieves the state-of-the-art performance on MOT2016 and MOT2017 [22].

- We conduct comprehensive discussions on the impact of each component we introduce. Besides, we give a quantitative evaluation on the importance of alignment and noisy outlier removal, which shows both ancient and modern detectors can benefit from these strategies.

2. Related Work

Since the tracking-by-detection approach becomes the mainstream in multi-target tracking, data association is regarded as the core part of MOT [21]. There are a lot of ways to solve data associations. Widely adopted probabilistic inference methods like Kalman filter [26], extended Kalman filter [23] and particle filter [2] rely on the first-order Markov assumption to estimate position in the new frame based on previous states and present observations. New detections can be assigned locally between adjacent or nearby frames using bipartite algorithms such as the optimal Hungarian algorithm [16, 35], or k-partite graph matching [8]. Multiple Hypothesis Tracking (MHT) [1, 17] postpones determinating ambiguous matches until enough information obtained. It results in a combinatorially increasing search space so that the hypothesis tree is pruned regularly. These local data association based tracking are sensitive to occlusion and noisy detections.

Tracking algorithms with global or delayed optimization [1, 17] try to produce longer and more robust trajectories by considering more frames or even the entire sequence in one shot. A popular paradigm is to formulate the task of MOT as an equivalence of solving an extremum of graph [31, 32, 33, 36, 3]. For instance, multi-cut-based approaches[31, 32, 33] try to decompose graph into isolated components with multi-cut algorithm, so that detections in same component belong to same identity. Generalized Minimum Clique Graph[36] treats detection association as to find the minimum clique in a corresponding graph. However, these graph-based approaches are NP-hard, indicating that only sub-optimal solution can be achieved even with expensive approximate methods.

Exceptionally, network-flow-based tracking [37, 24, 10, 4] uses graph formulation and can be solved in polynomial time. It restricts the cost function to contain only unary and pairwise terms to achieve efficient inference. For instance, the work of Zhang *et al.* [37] and Pirsiavash *et al.* [24] both assume logarithm cost functions and solve min-cost max-flow by push-relabel or successive shortest path algorithms. Dehghan *et al.* [10] use network flow to simultaneously predict detections and associate identities. Butt and Collins [4] encode neighboring connections differently so they leverage network flow to capture the relationships among three consecutive frames. Our work is most similar to the work of Schulter *et al.* [29] in that we both formulate the MOT problem in network flow paradigm and solve it by a bi-level optimization problem. Our work differs in the hierarchical design and efficiency.

Recently, people use deep learning to improve the tracking performance. DeepMatching [32] applies a convolutional neural network (CNN) to yield non-rigid matching between image pairs. Sadeghian *et al.* [27] encode history trajectory into embeddings by long short-term mem-

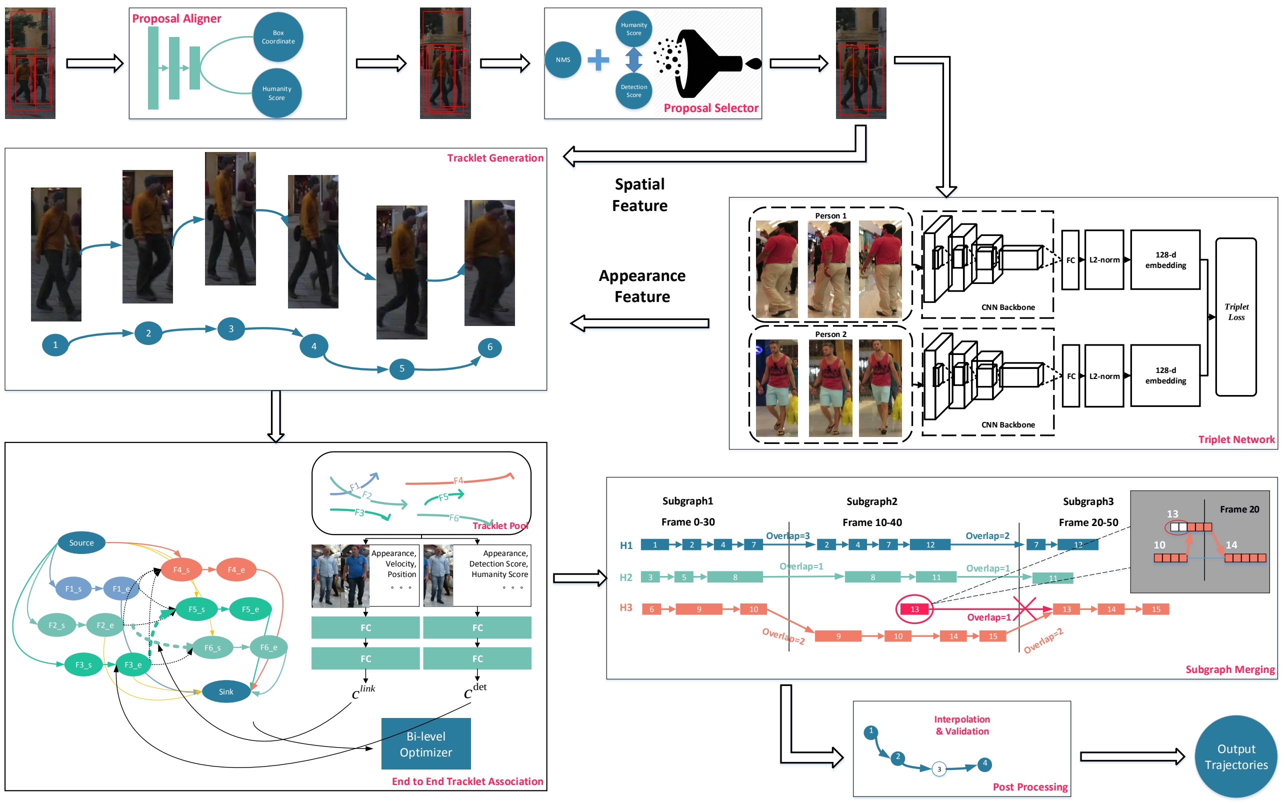


Figure 1: TAT workflow. 1) The detection responses are processed by the "Proposal Aligner" and "Proposal Selector" to produce aligned and high-quality bounding boxes. 2) They are fed into "Triplet Network" to get the corresponding appearance embeddings. 3) Then, we use a simple Multilayer Perceptron (MLP) to link neighboring boxes into tracklets according to the appearance and spatial features, a.k.a "Tracklet Generation". 4) As the core step of TAT, we propose an end-to-end network flow association approach (bridged on bi-level optimization) to learn costs $\boldsymbol{c}^{det}, \boldsymbol{c}^{link}$ from the tracklet feature, a.k.a "Tracklet Association". The mappings from $\boldsymbol{c}^{det}, \boldsymbol{c}^{link}$ to the edges of a network flow graph are shown by the black arrows in the "End to End Tracklet Association" module. 5) To reduce graph size, the sequences are divided into subgraphs, so there are strategies to merge results of subgraphs into long trajectories, as shown in the "Subgraph Merging" module. 6) Finally, in the post-processing step, we do interpolation and post-validation to generate the final trajectories.

ory(LSTM) [15] and compute embedding affinity with present detections. Quadruplet CNN [30] simultaneously train a multi-task loss to jointly learn object appearance and bounding box regression, and adopt minimax label propagation for matching. Instead, our work uses triplet loss and solves the association problem using a learnable association framework.

3. MOT Framework

Fig. 1 illustrates the fundamental steps in TAT. The components include 1) a proposal aligner, a proposal selector and a triplet network to achieve accurate appearance model; 2) a tracklet generation module to connect neighboring bounding boxes; 3) an end-to-end bi-level optimization tracklet association module to associate tracklets. 4) subgraph merging and post-processing module to propose final trajectories. In this section, we will elaborate on each step.

3.1. Appearance Model

Proposal aligner. To train an appearance model with high discriminative ability, it is essential to have bounding boxes aligned with targets; otherwise, the appearance model is prune to obscurity. Ancient detectors suffers from localization accuracy [12], while modern detectors are limited by the variance of preset anchor sizes and aspect ratios [25]. Thus, a secondary alignment is beneficial for that detections can be treated as better anchors compared to preset ones. Hence, we adopt a region proposal aligner with convolutional neural network architecture. It takes the slightly padded image patches of the corresponding detection responses as input, the aligned bounding box offset $\Delta\boldsymbol{p}$ and their respective classification scores $\boldsymbol{h}$ as output. It allows

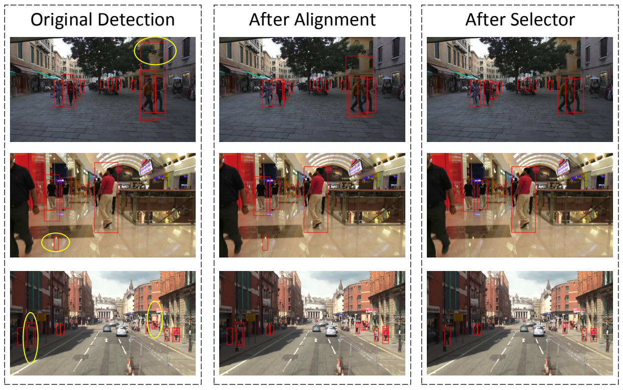


Figure 2: The proposal aligner and selector demonstration. The yellow ovals indicate false positives of DPM [12].

us to treat the raw detections as anchors, and to perform a regression enhancement based on the accurate baseline.

Proposal selector. After aligning the detection boxes, there are cases where two boxes of a single target overlap larger than before. We use non-maximum-suppression (NMS) to remove duplicates. Further, we use the classification score of the proposal aligner as a new indicator, naming it *humanity* for that it reflects the probability of the target being a human w.r.t. the new box coordinates. We filter out the boxes with both low humanity and detection score because true positives are unlikely to perform badly simultaneously in two measures, hence we have high confidence to regard the removed boxes as false positives. Through proposal selection, the remaining proposals are cleaner and are less likely to result in redundant overlapping trajectories.

Triplet Network. In a surveillance video, it is common to assume that human appearances do not change much in a short period. Based on the cleaned proposals processed by the proposal aligner and selector, we use metric learning [28, 6, 20, 34, 30] to learn an embedding $\boldsymbol{z}$ for each tracking target candidate so that the distance between two targets with the same identity is smaller than that with different identities. We use the triplet loss as the training goal:

$$L_{triplet} = \sum_{(\boldsymbol{z}_a,\boldsymbol{z}_p,\boldsymbol{z}_n)\in\mathcal{Z}} max(0, \boldsymbol{d}_a(\boldsymbol{z}_a,\boldsymbol{z}_p) - \boldsymbol{d}_a(\boldsymbol{z}_a,\boldsymbol{z}_n) + \theta), \tag{1}$$

where $(\boldsymbol{z}_a,\boldsymbol{z}_p,\boldsymbol{z}_n)$ denotes an instance of triplet where $\boldsymbol{z}_a$ is the anchor, $\boldsymbol{z}_p$ is a candidate from the same trajectory (positive sample), and $\boldsymbol{z}_n$ is a candidate from a different one (negative example). $\boldsymbol{d}_a(\boldsymbol{z}_1,\boldsymbol{z}_2)$ denotes the Euclidean distance between $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, which is called the *appearance distance*.

We apply a convolutional neural network to learn the embedding $\boldsymbol{z}$, with the architecture illustrated in "Triplet Network" module of Fig. 1. The convolutional feature maps of original target images are flattened, fed into the fully connected layers and finally normalized by an L2-Normalization (L2-Norm) layer. The output of L2-Norm is the 128-dimensional *appearance embedding*, $\boldsymbol{z}$.

We adopt online sampling to generate more instances. The sampling strategy includes: 1) we sample N persons and l instances of each per batch; 2) we divide each trajectory into segments so that the temporal distance within a segment is no longer than τ; 3) in each batch, we select one segment to draw samples of a person, with target detection boxes randomly shifted around at each frame as data augmentation; 4) we add in the targets that the detector has missed (but in the labeled ground truth) into the training set.

Similar to FaceNet [28], we make full use of every positive sample pair, but only use the most violating K negative samples selected by hard negative mining, together with K randomly drawn violating samples to construct the triplet set $\mathcal{Z}$. Consequently, the process produces a total number of $|\mathcal{Z}| = N \times l \times (l-1) \times 2 \times K$ triplets per batch.

3.2. Two-level Association

The difficulty of data association is different in sparse and dense scenes. Inspired by Huang *et al.* [16], we use a two-level association paradigm. The method connects neighboring boxes into tracklets at the low level and performs the end-to-end tracklet association at the high level.

3.2.1 Tracklet Generation

Performing low-level association with simple model helps us cut down the number of nodes and candidate edges. To learn a robust low-level affinity model, we take both appearance and spatial features into consideration as the input of the affinity model. Then we use the Hungarian algorithm [19] based on the output of the affinity model to decide whether candidate pairs between adjacent frames should be matched. The track formed by the matched boxes, a.k.a. *tracklets*, is used in the second stage association.

Appearance feature. We use the embedding distance between two candidates of the pair as appearance feature, denoted as $\boldsymbol{d}_a(\boldsymbol{z}_1,\boldsymbol{z}_2)$ (from Section 3.1). The metric reflects how similar two candidates look alike in visual.

Spatial feature. For low-level association, we only take the candidate pairs in adjacent frames into consideration. We define the *relative position distance* as

$$\boldsymbol{d}_p(P_1,P_2) = \left(\frac{|P_{x_1}-P_{x_2}|}{\sqrt{P_{w_1}P_{w_2}}}, \frac{|P_{y_1}-P_{y_2}|}{\sqrt{P_{h_1}P_{h_2}}}, \frac{|P_{w_1}-P_{w_2}|}{P_{w_2}}, \frac{|P_{h_1}-P_{h_2}|}{P_{h_2}}\right), \tag{2}$$

where $(P_{x_i},P_{y_i},P_{w_i},P_{h_i})_{i=1,2}$denotes point-size coordinates of the two bounding boxes of the candidate pair.

Feature fusion. Concatenating the appearance distance $\boldsymbol{d}_a(\boldsymbol{z}_1, \boldsymbol{z}_2)$, the relative position distance, along with humanity score for the two patches as feature input, we train an MLP classifier to predict the affinity score $y \in [0, 1]$ between candidate pairs. A score close to 1 indicates the pair bounding boxes belong to same trajectory and score close to 0 otherwise. Assuming there are r_t detections in frame t and r_{t+1} detections in frame $t+1$, we construct a matching matrix $\mathcal{Y} \in R^{r_t \times r_{t+1}}$ from affinity score of the candidate pairs. Then we use the Hungarian algorithm to calculate matching pairs. From a conservative perspective, we only keep matching results with very high confidence at this stage by applying the Eq.(3) of [16], and leave uncertain matchings to next association phase where we consider more extended sequence information.

3.2.2 Tracklet Associaton

Given tracklets from Section 3.2.1, we train an *association model* based on an end-to-end learnable network flow formulation to associate the tracklets into trajectories.

Problem Formulation. Fig 1 illustrates the structure of our network flow paradigm in "End-to-end Tracklet Association" module. We use nodes to represent tracklets, and edges to represent candidate tracklet pairs that may be associated. The *source* and *sink* are two auxiliary nodes to indicate the initialization and termination of trajectories.

The bi-level optimization problem for MOT is formulated as follows [28]. We solve for the parameter $\boldsymbol{\theta}$ with:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{x}^{gt}, \boldsymbol{x}^{\star}) = \sum_{\kappa \in \{det, init, term\}} \sum_i w_i (x_i^{\kappa, gt} - x_i^{\kappa, \star})^2 \tag{3}$$

$$\boldsymbol{x}^{\star} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_i c_i^{det}(\boldsymbol{f}_u, \boldsymbol{\theta}_1) x_i^{det} + \sum_{i,j} c_{i,j}^{link}(\boldsymbol{f}_p, \boldsymbol{\theta}_2) x_{i,j}^{link} + \sum_i c_i^{init} x_i^{init} + \sum_i c_i^{term} x_i^{term} \tag{4}$$

s.t.,

$$\boldsymbol{x} = [\cdots, x_i^{det}, x_i^{init}, x_i^{term}, \cdots, x_{i,j}^{link}, \cdots] \in \mathcal{R}^{M \times 1} \tag{5a}$$

$$\boldsymbol{C}\boldsymbol{x} = \boldsymbol{0}, \tag{5b}$$

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{5c}$$

where $\boldsymbol{x}^{\{det, link, init, term\}} \in \{0, 1\}$ represents whether the unary, pairwise, source-to-node and node-to-sink edges are connected, and $\boldsymbol{c}^{\{det, link, init, term\}}$ are their corresponding costs. The weights w is a hyper-parameter balancing the importance of each edge. For more details on definition of $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, please refer to the work of Schulter *et al.* [29]. The earlier work [29] uses log-barrier and basis substitution to eliminate constraints, and achieves the expected partial derivative of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}}$(Eq. (10) of [29]). We find the solution to be unnecessary and defective, so we give our solution which can achieve the goal without the complicated matrix multiplication, as stated later in this section.

Based on the problem formulation, we make the following three significant improvements:

Improvement 1: Using tracklet-level features. We define the tracklet set as $\{F_i, i = 1, \cdots, n\}$. We model the unary cost $\boldsymbol{c}^{det}$ and pairwise cost $\boldsymbol{c}^{link}$ as $\boldsymbol{c}^{det}(\boldsymbol{f}_u, \boldsymbol{\theta}_1)$ and $\boldsymbol{c}^{link}(\boldsymbol{f}_p, \boldsymbol{\theta}_2)$ that are fit by multi-layer perceptrons (MLP). $\boldsymbol{f}_u$ is the unary tracklet feature, and $\boldsymbol{f}_p$ is the feature extracted from tracklet pairs. $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ are MLP parameters w.r.t. unary and pairwise functions. For a pair of connected tracklets F_i and F_j, t_i and t_j denote the frame at the tail of F_i and the head of F_j, respectively. Variables $\vec{h}_i$, $\vec{d}_i$, and $\overline{\boldsymbol{z}}_i$ are the humanity scores, detection scores, and the average embedding of tracklet F_i. $\mathcal{S}$ and $\mathcal{A}$ denote the area and aspect ratio of the bounding box. Furthermore, we denote the time gap between two connected tracklets F_i and F_j as $\Delta t = |t_i - t_j|$. We use $\hat{P}_i^t$ and $\hat{P}_j^h$ to denote the forward and backward Kalman filter estimated position for F_i (from t_i to t_j) and F_j (from t_j to t_i), respectively. We also include in the absolute position distance between F_i and F_j, denoted as $\boldsymbol{d}_p(P_i^t, P_j^h)$, in case that Kalman Filter does not work robustly on short tracklets. We divide the position distance terms by Δt to support association across a long time period. With these notations, we can write $\boldsymbol{f}_u$ and $\boldsymbol{f}_p$ as

$$\begin{aligned}
&\boldsymbol{f}_u(i) = (\boldsymbol{median}(\vec{h}_i), \boldsymbol{median}(\vec{d}_i), |F_i|) \\
&\boldsymbol{f}_p(i,j) = (\boldsymbol{d}_a(i,j), \boldsymbol{d}_A(i,j), \boldsymbol{d}_S(i,j), \boldsymbol{d}_p^U(i,j), |F_i|, |F_j|, \Delta t), \\
&\text{where} \quad \boldsymbol{d}_a = \boldsymbol{d}_a(\overline{\boldsymbol{z}}_i, \overline{\boldsymbol{z}}_j), \quad \boldsymbol{d}_{\mathcal{A}} = \frac{\mathcal{A}_i^t}{\mathcal{A}_j^h}, \quad \boldsymbol{d}_{\mathcal{S}} = \frac{\mathcal{S}_i^t}{\mathcal{S}_j^h}, \\
&\boldsymbol{d}_p^U = (\frac{\boldsymbol{d}_p(\hat{P}_i^t, P_j^h)}{\Delta t}, \frac{\boldsymbol{d}_p(P_i^t, \hat{P}_j^h)}{\Delta t}, \frac{\boldsymbol{d}_p(P_i^t, P_j^h)}{\Delta t})
\end{aligned} \tag{6}$$

Improvement 2: Fixing training deficiency with approximate gradient. To solve the bi-level optimization problem Eq. 3- 4, we find it is unnecessary to calculate the partial derivative $\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}}$ precisely [29], not to mention the defect of the gradient formula they provide. The explaination is three-fold:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}} = -t \cdot \boldsymbol{B}[\boldsymbol{B}^T \frac{\partial^2 P}{\partial \boldsymbol{x}^{\star 2}} \boldsymbol{B}]^{-1} \boldsymbol{B}^T \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}^{\star}} \tag{7}$$

$$\frac{\partial^2 P}{\partial \boldsymbol{x}^2} = Diag(\frac{1}{(1 - x_i^{\star})^2} + \frac{1}{x_i^{\star 2}}). \tag{8}$$

1) When the temperature $t = 10M$, Eq. 8 results in an unstable gradient of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}}$. when $\boldsymbol{x}$ approaches $\{0, 1\}$, $\frac{\partial^2 P}{\partial \boldsymbol{x}^2}$ is large, and the corresponding gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}}$ becomes small, as a result the loss descends slowly at the beginning. On the other hand, when $\boldsymbol{x}$ approaches 0.5, $\frac{\partial^2 P}{\partial \boldsymbol{x}^2}$ is relative small, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}}$ increases sharply because of the large temperature. These extreme gradient values harms the training paradigm of deep learning, which usually uses fixed learning rates that are independent to the gradient order of each iteration. It's easily to get stuck or start oscillating during

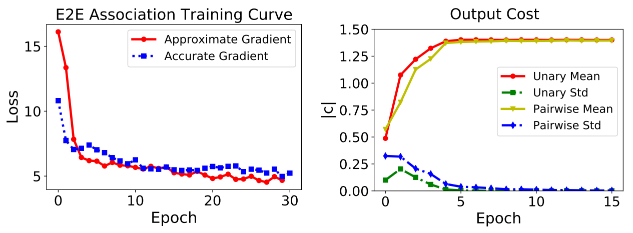
training. 2) Eq. 7 does not guarantee the movement smoothness of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}}$ because $B[B^T \frac{\partial^2 P}{\partial \boldsymbol{x}^2} B]^{-1} B^T \in \mathcal{R}^{M \times M}$ is not invertible. It indicates that even though $\boldsymbol{c}$ is moving towards a dedicated direction temporarily, the impact on $\boldsymbol{x}$ is not predictable. 3) Moreover, When moving to a combination of values of $\boldsymbol{c}$ and $\boldsymbol{x}$, the original constrained linear programming becomes hard to solve as it takes a long time to jump out of the bad point.

Thus, we doubt the plausibility of the chain rule after performing the basis substitution in Eq. 4. However, the formulation is still valuable, because we can use an approximate gradient rather than the accurate gradient to solve the above problems. Our idea is quite straightforward but unexpectedly works well: intuitively, if $\frac{\partial \mathcal{L}}{\partial x_i} > 0$, it means x_i is larger than the expected value x_i^{gt} (in our case, it also indicates that $x_i^{gt} = 0$). Then x_i is supposed to be turned down towards 0 so that its corresponding edge is not chosen into the final trajectory. To achieve this, the corresponding cost c_i should be enlarged. In contrast, if $\frac{\partial \mathcal{L}}{\partial x_i} < 0$, we should reduce the value of c_i so that x_i is increasing towards 1. To save computation, we simply set $\frac{\partial L}{\partial \boldsymbol{c}} = -\frac{\partial L}{\partial \boldsymbol{x}}$.

Improvement 3: Bound the range of cost parameters. Also, we find Schulter *et al.* [29] do not constrict the value of the cost vector $\boldsymbol{c}$. It is risky because we never know what value the cost $\boldsymbol{c}$ will converge to. In our experiments, the value of $\boldsymbol{c}$ sometimes diverges to infinity if the learning rate is not carefully set. Even converged, the final converge point is not predictable, and the computation of the constrained LP problem with large value is costly. Instead, we bound the value of $\boldsymbol{c}^{det,link}$ to range $[-\gamma, \gamma]$ by adding a $tanh$ function to the output of the MLP, and we can arbitrarily initialize $\boldsymbol{c}^{init,term}$ to some constant $\beta \in [-\frac{\gamma}{2}, \frac{\gamma}{2}]$. Fig 3b shows the absolute value of $\boldsymbol{c}^{det}$ and $\boldsymbol{c}^{link}$ finally converges at the line of 2β, indicating that c^{det} and c^{link} can be automatically adjusted to the boundary of tracklet TP and FP: -2β, only if the value of -2β is reachable.

Fig. 3a shows the learning curve by using the approximate gradient. In practice, we find the training speed with our approximation approach is over 20× larger than the original, without any sacrifice of performance. It is because the accurate gradient in Eq. 7 contains high dimensional matrix multiplication and inversion, while the approximate one is a lot cheaper.

Other parameter choices. To deal with long video sequences, we use a W-frame sliding window to segment video sequences into subgraphs. We can use a larger window and steps because tracklets generates much fewer candidate pairs comparing to detection responses. Tracklets with head or tail locating in the temporal window are included as nodes in the subgraph. We set the stepping size to $\frac{1}{4} \sim \frac{1}{2} W$ to preserve the overlapping necessary for merging subgraph associations into the longer tracks. We use Hungarian matching over intersections between tracklet sets to merge these subgraphs. An example of subgraph merging result shows in Figure 1. Here, rectangle blocks are tracklets with their widths showing different tracklet lengths, and three subgraphs generate three hypothetic long tracks. As depicted in the figure, *Subgraph 1* produces three intermediate long tracks while *Subgraph 2* produces four. These intermediate tracks generate a matching matrix in $\mathcal{R}^{3 \times 4}$, with elements being the number of overlapping blocks between tracks. Then the Hungarian algorithm is used to find the matching with largest overlaps, and merge-sort is used to combine tracklets in order.



(a) (b)

Figure 3: (a) The learning curves of the end-to-end association by calculating the accurate gradient or approximate gradient. (b) The transition curve of the unary and pairwise costs in absolute form.

The setting of weights $\boldsymbol{w}$ in Eq. 3 for tracklet fusion is more straightforward than those at the detection response level. Existing work [29] empirically sets w lower for a punishment of ambiguous links like FP-FP(two false positives) and TP-TP+Far(true positives with the same identity but distant). However, in the case of tracklets, the weighting directly reflects on the final metric MOTA. For instance, a FP edge between tracklets leads to $|F_i^{tail} - F_j^{head}|$ FP boxes in the final trajectory; in contrast, a FN pair losses $|F_i^{tail} - F_j^{head}|$ expected boxes in the final trajectory, and same deduction applies to unary terms. Thus, we set the weights of x_i^{init}, x_i^{term} to 1, the weights of x_i^{det} to the length of tracklets, and the weights of $x_{i,j}^{link}$ to the time gap between tracklet pairs. Our experiments in Section 4 show that the weighting strategy improves performance.

3.3. Post Processing

There are certain detection gaps between tracklets caused by occlusions, and we fill them in post-processing. Bilinear interpolation is a general approach, but it is prone to introduce errors. To validate the legitimacy of the interpolated boxes (a.k.a virtual patches), we feed the corresponding virtual patches into Proposal Alignment Network to get the humanity and regressed boxes, and compute appearance distance corresponding to the last box in the filled trajectory. We discard the interpolated boxes with low humanity or appearing very different from others in trajectory.

4. Evaluation

4.1. Implementation Details

Datasets. We evaluate our approach on both the 2D MOT2016 and MOT2017 Benchmarks. MOT2016 offers a variety of 14 video sequences (7 for training and 7 for testing) [22] which are captured by static or moving cameras indoors and outdoors. MOT2016 provides with the detection responses of DPM detector for training and testing. MOT2017 contains the same sequences but provides with more precise ground-truth annotations. The organizers release the detection results of Faster R-CNN and SDP in addition to that of DPM so that researchers are supposed to submit tracking results under the three detectors.

Platform. All of our experiments are conducted on a 1.2GHZ Intel Xeon server with 8 NVIDIA TITAN X GPUs. The deep learning framework we use is MXNet [5].

Appearance model. We use an ImageNet pre-trained ResNet-50 [13] as the CNN backbone for both the Proposal Aligner and the Triplet Network. We set the input size to 128×64, and use ReLU for activation and ADAM [18] optimizer to train the networks. The base learning rate is set to $1e-4$, $1e-6$, respectively. The hyper-parameters of sampling are set as: $l = 8, N = 50, K = 100, \alpha = 0.8, \tau = 60$.

Bi-level optimization training. We use a three-layer MLP with Leaky ReLU activation for the pairwise network. The reason of using Leaky ReLU rather than ReLU is that the regression target is near a single value: both $\boldsymbol{c}^{det}$ and $\boldsymbol{c}^{link}$ are expected to converge at around of -2β so that the neural work tends to move simultaneously at same pace for all samples, and using ReLU may wipe out sample difference at very beginning. We choose *cvxpy* [11] as the convex problem solver, and also ADAM [18] as the MLP optimizer. We set the value of β to 0.7 and γ to 5 in all experiments.

Metrics. For evaluation, we report the following metrics: 1) the most commonly used metric, $MOTA = 1 - \frac{FP+FN+IDS}{GT}$, which provides a combination of the false positive (FP), false negative (FN) and ID switch (IDS) frequency among all trajectories against the ground truth (GT); 2) mostly tracked (MT) and 3) mostly lost (ML) that provide an indication of the trajectory fragmentation.

4.2. Comparison to the State-of-the-art

Table 1 shows our MOT2016 benchmark results in comparison with the current competitive approaches. We observe that 1) our MOTA of 49.0 outperforms all published results as of March of 2018. We believe this high MOTA is a direct result of low FN of $82,506$, attributing to both the alignment and association approach.

Table 2 shows our result on MOT2017 benchmark. We haven't performed any fine-tuning on this dataset based on MOT2016, and 3 detectors share the same set of configuration. Our result ranks the 3rd place with MOTA of 51.5 by April 9th, 2018, with the 1st and 2nd place being non-published. The performance on Faster R-CNN is the highest on board. By fine-tuning the parameters, it's possible to achieve higher performance. We choose not to do so but only show the fact that our method applies to any detectors, and better detector benefits tracking result.

Tracker	MOTA↑	MOTP↑	IDS↓	MT↑	ML↓
GMMCP [9]	38.1	75.8	937	8.6	50.9
QCNN [30]	44.1	76.4	745	14.6	44.9
MHT_DAM [17]	45.8	76.3	590	16.2	43.2
NOMT [7]	46.4	76.6	**359**	18.3	41.4
AMIR [27][a]	47.2	75.8	774	14.0	41.6
FWT [14]	47.8	75.5	852	19.1	38.2
LMP [33]	48.8	**79.0**	481	18.2	40.1
TAT	**49.0**	78.0	899	**19.1**	**35.7**

[a]Online tracker

Table 1: Results on the MOT2016 test data.

Tracker	Detector	MOTA↑	IDS↓	MT↑	ML↓
[17]	DPM	44.6	593	15.3	45.2
	FRCNN	46.9	742	18.6	34.7
	SDP	60.6	979	28.7	30.8
	Total	50.7	2,314	20.8	36.9
[14]	DPM	46.4	833	18.2	40.8
	FRCNN	48.2	780	18.5	35.9
	SDP	59.4	1035	27.6	29.0
	Total	51.3	2,648	21.4	35.2
eHAF 17[a]	DPM	45.9	552	18.3	44.9
	FRCNN	47.8	605	20.0	36.7
	⋆SDP	61.8	677	31.9	32.1
	Total	51.8	**1,834**	**23.4**	37.9
CNN _search[b]	⋆DPM	49.8	1,209	19.5	36.2
	FRCNN	46.5	828	17.6	34.4
	SDP	60.2	1022	27.1	29.4
	Total	**52.2**	3,059	21.4	**33.3**
TAT	DPM	47.0	698	15.2	42.3
	⋆FRCNN	48.3	866	17.5	36.6
	SDP	59.2	1,029	29.2	27.6
	Total	51.5	2,593	20.6	35.5

[a]Anonymous submission
[b]Non-published work from Hikvision Research Institute

Table 2: Result on the MOT2017 test set. The ⋆ denotes the best performance achieved yet of each detector.

4.3. Ablation Study

To give a transparent demonstration of the impact of each component we have introduced, we perform comparative experiments at each stage on MOT2016. For all experiments in this section, we use the last two training sequences (MOT16-11, MOT16-13) for validation and the

Case	MOTA↑			RTP↑	RFP↓
Seq	11+13	11	13	11+13	11+13
DEFAULT	32.5	49.9	18.6	8,500	5,445
AL	34.5	51.2	21.2	8,590	5,355
AL + S-NMS	**37.3**	53.8	**24.1**	8,512	3,181
AL + S-NMS + S-SF	37.0	**54.9**	22.6	8,307	1,244

Table 3: Performance of the proposal aligner and the proposal selector on the validation set. RFP and RTP denote the residual FP and TP boxes after each step.

rest for training.

Proposal aligner & proposal selector. We evaluate the effects of the proposal aligner and proposal selector, which have shown significant improvement on the final result. We denote the Aligner as *AL*, non-maximum suppression and the score filtering of the selector as *S-NMS* and *S-SF*. We compute the IOU ratio against the ground truth. We define an IOU greater than 0.5 as a true positive (TP), and otherwise as a false positive (FP). Under this setup, we get $8,500$ TPs, $5,445$ FPs from the two validation sequences. We separately evaluate these elements, and use the best association setting of TAT to do the two-level association. Table 3 summarizes the results.

We notice that by performing the proposal alignment, we can rectify 90 FPs to TPs in the validation set. It indicates that tighter boxes can not only benefit feature extraction but also explicitly enhance the recall of proposals. Besides, by performing S-NMS and S-SF, the number of FP decreases from 5355 to 1244, with a sacrifice of only 283 TPs. The MOTA after alignment and selection is 13.9% larger than that of raw input. To clarify, we use 0.7 as NMS threshold, and remove the boxes with humanity lower than 0.1 and the detection score lower than 0 in this experiment. Moreover, the experiment shows MOTA decreases slightly in MOT16-13. It is because the sequence contains many people in shadow, which is a rare scenario in the training set. Thus they are assigned low humanity, and S-SF filters them out. However, the case happens rarely in general circumstances, and we employ all these strategies in other experiments.

To validate whether all detectors benefit from the aligner and selector, we report the statistics on the MOT2017 dataset in Table 4. From training set, we see remarkable FP decrease on DPM and respectively a high improvement on MOTA. While for Faster R-CNN and SDP which have coordinate regression module in their algorithms, the benefit is less but still visible. It's partially because for these detectors the alignment is good enough.

Using tracklets vs. detection responses directly. To demonstrate the improvements that tracklets bring, we train an end-to-end association model directly on features extracted from the detection responses. The feature extraction is the same as in Section 3.2.1. We set W to 5 frames and step size to 1 to accommodate the large candidate pairs and ensure overlapping. The two experiments achieve same MOTA at 35.9, while TAT runs much faster than the other. Moreover, TAT achieves best performance 37.9 at $W = 30$ as shown in Fig 4. In the same configuration, we cannot even obtain a model on detection response level due to the large number of search space.

Detector	RTP/OTP	RFP/OFP	MOTA(R/O)
DPM	50,160/52,760	2,890/27,030	34.9/47.0
FRCNN	64,235/64,252	3,316/3,387	46.0/48.3
SDP	78,349/78,623	3,733/4,164	56.2/59.2

Table 4: Effect of proposal aligner/selector under different detectors on the MOT2017 dataset. We report the TP/FP on MOT2017 training set as a reference. OTP/OFP denotes original TP/FP; RTP/RFP denotes the remaining TP/FP after performing alignment and selection. The MOTA shown here is evaluated on the test set.

End-to-end Learned vs. Hand-crafted Affinity. The significant improvement of our association method is to incorporate learned-features and cost parameters. To evaluate the improvements, we compare the following two association methods with TAT.

[NETFLOW] As a comparative method, we use an independent affinity model together with a standalone inference method to replace the bi-level optimization association paradigm. The proposal alignment, selection and tracklet generation are performed as the same. We train a 2-class MLP classifier as the affinity model, with label 1 representing tracklets of same target and label 0 representing tracklets of different targets. We denote the output of the MLP as s^p, and manually design the edge cost as $c_i^{link} = -log(s_i^p)$, the node cost as a product of 1 minus humanity, i.e. $c_i^{det} = log(\prod_k (1 - h_{i,k})), k = 1, \cdots, |F_i|$. Then we apply the Algorithm 1 [37] as tracklet association approach.

[E2EP] It is a transition method between TAT and *[NETFLOW]*. It also uses an end-to-end model, except that we use the same unary feature c_i^{det} as *[NETFLOW]*. However, to avoid applying grid search for $\boldsymbol{c}^{init,term}$, we use a linear model to learn the direction and bias. Thus we modify $c^{det} = a * log(\prod_k (1 - h_{i,k})) + b, a, b \in \mathcal{R}$, and set $c^{init,term}$ to 0.7.

[TAT] We use both learnable unary and pairwise terms for TAT, with features defined in Section 3.2.2. We construct the unary and pairwise MLP with [8, 4, 1] and [256, 256, 1] network architecture, respectively.

For each method, we conduct experiments using window sizes ranging from 10 to 100 frames. Fig 4 shows the results on the validation set. Our key observations are:

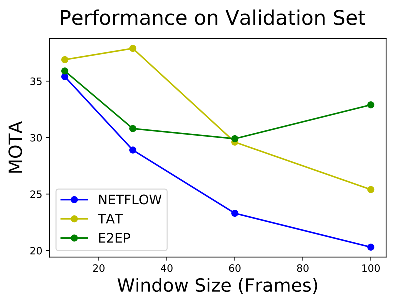


Figure 4: End-to-end Learned vs. Hand-crafted Affinity. The curve in yellow, green, and blue presenting the MOTA achieved on the validation set at different window length by *TAT*, *[E2EP]*, and *[NETFLOW]*, respectively.

Case	MOTA↑	IDS↓	FN↓/FP↓	MT↑/ML↓
Uniform	35.9	**69**	12770/**378**	11.9/54.6
TL	36.9	77	12550/393	11.9/54.6
TG	35.9	91	12716/413	11.9/54.0
TL + TG	**37.0**	75	**12528**/388	11.9/**51.7**

Table 5: The performance on the validation set of variant weighting strategy. The window size is set to 10.

1) Both TAT and *[E2EP]* outperform *[NETFLOW]*. This improvement shows the effectiveness of the end-to-end training.

2) When the window size is small, TAT significantly outperforms the other two methods, thanks to the automatically tuned cost parameters.

3) When the window size is over 30 frames, both TAT and *[NETFLOW]* show a significant drop in performance, except for *[E2EP]*. The drop is because performing long-term association has little benefit when 95% expected links are less than 30 frames. In contrast, it brings in a risk of obscuring the pairwise features because of Δt. Thus, the false connections will result in more false positives interpolated. However, *E2EP* is robust to the large window size. It is because the hand-crafted feature $\boldsymbol{c}^{det}$ has a clear intrinsic relation to affinity, which makes the association results more stable when $\boldsymbol{c}^{link}$ becomes inferior.

Weighting. We claim that tracklet weight $\boldsymbol{w}$ in Equation 3 should be related to the tracklet length (*TL*) and time gap of connections (*TG*), whose errors directly reflect on the MOTA. From Table 5 we notice the major improvement comes from TL weighting, which matches our expectation that it costs more to make mistakes on longer tracklets.

5. Conclusion and Future Work

Unlike many other tasks in computer vision that adopts end-to-end training, MOT still requires much hand-tuning and optimizations in various stages. TAT brings MOT an extra step closer to fully end-to-end. With TAT, we combine the classic tracklet-based association into the new bi-level optimization framework. It is easy to integrate additional features since they can be jointly considered in the end-to-end learning framework, and the training process converges much more stable and faster by using the approximated gradient.

Combining tracklets and end-to-end training opens many opportunities for future improvements. To begin with, we can encode human interactions in the tracklet feature. Furthermore, we can adopt LSTM-based tracklet features, so that the tracker can perform end-to-end learning from raw data. Lastly, we can investigate how to improve linking consistency around short tracklets in a learning-based association framework.

References

[1] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine (AESM)*, 19(1):5–18, 2004.

[2] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *International Conference on Computer Vision (ICCV)*, pages 1515–1522. IEEE, 2009.

[3] W. Brendel, M. Amer, and S. Todorovic. Multiobject tracking as maximum weight independent set. In *Computer Vision and Pattern Recognition (CVPR)*, pages 4091–4099. IEEE, 2011.

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

[5] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[6] W. Chen, X. Chen, J. Zhang, and K. Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2. IEEE, 2017.

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

[8] R. T. Collins. Multitarget data association with higher-order motion models. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1744–1751. IEEE, 2012.

[9] A. Dehghan, S. Modiri Assari, and M. Shah. GMMCP tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking. In *Computer Vision and Pattern Recognition (CVPR)*, pages 4091–4099. IEEE, 2015.

[10] A. Dehghan, Y. Tian, P. H. Torr, and M. Shah. Target identity-aware network flow for online multiple target tracking. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1146–1154. IEEE, 2015.

[11] S. Diamond and S. Boyd. CVXPY: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research (JMLR)*, 17(83):1–5, 2016.

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

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

[14] R. Henschel, L. Leal-Taixé, D. Cremers, and B. Rosenhahn. Improvements to Frank-Wolfe optimization for multi-detector multi-object tracking. *arXiv preprint arXiv:1705.08314*, 2017.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] C. Huang, B. Wu, and R. Nevatia. Robust object tracking by hierarchical association of detection responses. In *European Conference on Computer Vision (ECCV)*, pages 788–801. Springer, 2008.

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

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

[20] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler. Learning by tracking: Siamese CNN for robust target association. In *Computer Vision and Pattern Recognition Workshops*, pages 33–40. IEEE, 2016.

[21] W. Luo, J. Xing, X. Zhang, X. Zhao, and T.-K. Kim. Multiple object tracking: A literature review. *arXiv preprint arXiv:1409.7618*, 2014.

[22] 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.

[23] D. Mitzel and B. Leibe. Real-time multi-person tracking with detector assisted structure propagation. In *International Conference on Computer Vision (ICCV) Workshop*, pages 974–981. IEEE, 2011.

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

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99. NIPS Foundation, 2015.

[26] M. Rodriguez, J. Sivic, I. Laptev, and J.-Y. Audibert. Data-driven crowd analysis in videos. In *International Conference on Computer Vision (ICCV)*, pages 1235–1242. IEEE, 2011.

[27] A. Sadeghian, A. Alahi, and S. Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *arXiv preprint arXiv:1701.01909*, 2017.

[28] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Computer Vision and Pattern Recognition (CVPR)*, pages 815–823. IEEE, 2015.

[29] S. Schulter, P. Vernaza, W. Choi, and M. Chandraker. Deep network flow for multi-object tracking. *arXiv preprint arXiv:1706.08482*, 2017.

[30] J. Son, M. Baek, M. Cho, and B. Han. Multi-object tracking with quadruplet convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 5620–5629. IEEE, 2017.

[31] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Subgraph decomposition for multi-target tracking. In *Computer Vision and Pattern Recognition (CVPR)*, pages 5033–5041. IEEE, 2015.

[32] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Multi-person tracking by multicut and deep matching. In *European Conference on Computer Vision (ECCV)*, pages 100–111. Springer, 2016.

[33] S. Tang, M. Andriluka, B. Andres, and B. Schiele. Multiple people tracking by lifted multicut and person re-identification. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3539–3548. IEEE, 2017.

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

[35] J. Xing, H. Ai, and S. Lao. Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1200–1207. IEEE, 2009.

[36] A. R. Zamir, A. Dehghan, and M. Shah. GMCP-Tracker: Global multi-object tracking using generalized minimum clique graphs. In *European Conference on Computer Vision (ECCV)*, pages 343–356. Springer, 2012.

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