Learning of Global Objective for Network Flow in Multi-Object Tracking

Shuai Li\textsuperscript{1} Yu Kong\textsuperscript{1} Hamid Rezatofighi\textsuperscript{2}

\textsuperscript{1}Rochester Institute of Technology \textsuperscript{2}Monash University
\{s16009, Yu.Kong\}@rit.edu hamid.rezatofighi@monash.edu

Abstract

This paper concerns the problem of multi-object tracking based on the min-cost flow (MCF) formulation, which is conventionally studied as an instance of linear program. Given its computationally tractable inference, the success of MCF tracking largely relies on the learned cost function of underlying linear program. Most previous studies focus on learning the cost function by only taking into account two frames during training, therefore the learned cost function is sub-optimal for MCF where a multi-frame data association must be considered during inference. In order to address this problem, in this paper we propose a novel differentiable framework that ties training and inference together during learning by solving a bi-level optimization problem, where the lower-level solves a linear program and the upper-level contains a loss function that incorporates global tracking result. By back-propagating the loss through differentiable layers via gradient descent, the globally parameterized cost function is explicitly learned and regularized. With this approach, we are able to learn a better objective for global MCF tracking. As a result, we achieve competitive performances compared to the current state-of-the-art methods on the popular multi-object tracking benchmarks such as MOT16, MOT17 and MOT20.

1. Introduction

While being a classical problem, multi-object tracking (MOT) \cite{37,52} has been one of the most active research areas in computer vision, as being a fundamental basic-level perception task for many real-world problems, e.g., in visual surveillance, and autonomous driving \cite{20}. Thanks to the great progress in object detection \cite{18,42} techniques, “tracking-by-detection” paradigm has dominated the tracking community recently. Given an input video, a set of detection hypotheses is first generated for each frame and the goal of tracking is to associate these detection responses across time, locally or globally, to form all the trajectories. Among various previous works, minimum-cost network-flow \cite{4,41,60} based methods have gained increasingly attention due to its fast inference property. In this work, we specifically focus on the network-flow based tracking.

The min-cost network flow formulation for multi-object tracking problem is, indeed, an instance of constrained integer linear program (ILP) with uni-modular constraint matrices \cite{60}. Therefore, the solution to such an ILP problem can be optimally obtained by solving its relaxed version, i.e., a constrained linear program (LP), which has an identical optimal integer solution to its ILP counterpart \cite{2}. Given its computationally tractable inference, the success of a network-flow based multi-object tracking approach largely depends on designing a proper cost function. Many previous works have focused on learning a robust objective function, e.g., the matching cost, between detections in two frames utilizing a neural network trained using a, e.g., binary cross-entropy \cite{53}, triplet \cite{11} or contrastive \cite{34} loss. The major drawback of these approaches is that they only consider a limited temporal context during training, e.g., two or three frames. As such, the learned objective function can be sub-optimal as it ignores long-term temporal contexts and associations. Several recent works adopt graph neural networks \cite{8,14,39} in order to learn a better feature representation for a spatio-temporal graph formulating multi-object tracking problem. However, their training objectives are still constrained locally, e.g., a binary cross-entropy as a local edge loss is employed during the training stage and thus, knowledge about the global tracking result is not yet properly incorporated.

Recently, there have been few attempts to learn a proper objective function for an LP problem representing a global data association in MOT \cite{19,50}. Seminal work of \cite{50} adds a log-barrier term into the objective function and adopts a change of basis technique to deal with equality constraints in the linear program, as a result, heuristics are involved in choosing the optimal temperature parameters during the interior method’s optimization process, and the tracking results are inferior compared to many top-ranked methods in tracking benchmarks \cite{20,37}. The work of \cite{19} performs 3D tracking on video and LiDAR data, followed by a re-projection step to perform 2D tracking, which is computationally demanding. In contrast to their works, we...
propose a general framework which adopts Bi-level optimization technique embraced with implicit function theorem [22] to perform end-to-end learning of a global cost function for LP based tracking. At the lower-level of our optimization, our framework solves a linear program and the upper-level contains a general loss function that regularizes the tracking solution. By approximating the original linear program as a continuous quadratic program during the forward pass, it is possible to differentiate through the optimal KKT conditions of the relaxed convex quadratic problem. In this way, the cost for data association can be trained end-to-end by back-propagating the gradient of the loss through differentiable layers. In addition, we integrate a stronger observation model [5] compared to [50], together with the learned optimal cost function for data association. Our framework achieves competitive results compared to current state-of-the-art approaches on MOT16, MOT17 and MOT20 benchmarks. In summary, our main contributions are as follows:

• We adopt the classical min-cost network flow formulation to address multi-object tracking problem and propose a novel bi-level optimization technique which is able to learn a global cost for tracking directly from multi-frame data association results.

• In order to address the non-differentiable problem of the constrained linear program, we propose to approximate the original integer linear program as a continuous quadratic program and to back-propagate quadratic program solution’s gradients w.r.t. the model’s parameters.

• The proposed tracking method achieves results comparable to current state-of-the-art trackers on the popular MOT16, MOT17 and MOT20 benchmark, demonstrating its effectiveness.

2. Related Works

Multi-Object Tracking Multi-object tracking remains an active field in computer vision for many years. The way of solving multi-object tracking can be roughly divided into two mainstream approaches namely online and offline methods. Online methods, make the decision by the observations up to the current frame. Popular approach of [55] employs Hungarian Algorithm [33] to associate observations to tracked objects first then use Kalman filter to update the object’s states in an recursive manner. JPDAF [44] extends Global Nearest Neighbor matching principle by allowing all observations to track associations within certain gating areas, making the solutions more robust at the cost of heavy computation. MCMC based data association methods [9, 29] provide a probabilistic formulation of data association and therefore incorporate arbitrary priors. There also exists tracking methods which make use of deep neural networks. Seminal work of Milan et al. [38] uses LSTM to address state estimation and data association jointly. Later work such as Fang’s et al. [17] tracked each object through RNN in real-time by coupling the internal and external memory cells. Although online methods can be used in time-critical situations, they make non-reversible decisions, due to the greedy fashion in the data association step.

Offline methods [26, 51] for MOT usually constructs a graph whose nodes are the detection hypothesis and edges are the potential links between detection hypothesis, by optimizing a well designed objective function with physically plausible constraints, the final tracking solution can be found. Among them, network-flow [4, 41, 60] based approaches have become popular due to its fast inference and global-optimal solution property, while more robust solutions can be achieved by employing higher-order terms [12, 26, 46, 54] at the cost of heavy computation.

Graph Neural Networks for MOT Multi-object tracking is essentially a graph optimization problem, and there are several works that attempt to solve tracking by adopting Graph Neural Networks [21, 32]. Early work of [27] combines CNN and LSTM to learn appearance and motion features together followed by GCN [32] for feature refinement. Li et al. [35] designs an appearance and motion graph network separately for feature learning using a modified message passing network, for online tracking. The recent work from Dai et al. [14] clusters and ranks tracklets through a graph convolution network and shows promising results. Braso and Leal-Taixé [8] leverages message passing networks for network flow based tracking but their work optimizes a binary cross-entropy loss during training and does not allow learning from data association directly, also a heuristic rounding step is applied to ensure disjoint path constraints. By contrast, our method directly perform back-propagation from data association and do not require heuristics at inference stage.

End-to-End Learning in MOT There exists several works that attempt to learn affinity measure for tracking in an end-to-end fashion. The framework in [57] utilizes a GRU to approximately differentiate Hungarian Algorithm and achieves descent performance. Burke and Ramamoorthy [10] adopts Sinkhorn Network within Kalman Filtering framework to learn association costs using an EM algorithm, but can only track fixed objects. Similar to our work, Papakis et al. [39] proposes a differentiable matching layer using Sinkhorn Network, while our work is a generalization of their work in multi-frame case. Peng et al. [40] perform joint detection and data association using a deep CNN. He et al. [23] presents an end-to-end learnable graph matching method but its quadratic formulation largely slows down the inference speed due to its exponential complexity. All of these works perform learning in an online manner. In contrast, our work performs end-to-end learning for global data association from multiple frames and is more robust than online methods during inference.
Figure 1. Illustration of the proposed tracking method. Given a sequence of frames and a set of detection hypotheses as input. Detection’s appearance features extracted by a pretrained person Re-Identification network are combined with geometric cues to build a directed flow graph where detections represent nodes and edges connect detections across frames. A MLP is used to regress the linking probability between detections. During training the lower-level linear program generates a prediction $x^*$, which is passed through a differentiable layer to produce a loss $L$ from upper-level, the loss is back-propagated through previous layers in order to learn an optimal parameterized cost $c(f;w)$. At inference time the model outputs data association by solving a linear program and achieves tracking. $A, b, G, h$ denote the Linear Program’s constraints.

3. Approach

In this section, we briefly revisit the minimum cost network flow formulation for solving multi-object tracking problem, then present our proposed end-to-end learning strategy which performs back-propagation through optimal linear program solutions that learns a suitable cost function for tracking task.

3.1. Minimum-Cost Network Flow Problem

Given a set of detection hypothesis $D = \{d_i\}$, where $d_i = (t_i, x_i, y_i, w_i, h_i, s_i)$ denotes a detection at frame $t_i$ located at position $x_i, y_i$ with bounding box size $w_i, h_i$ and confidence score $s_i$ respectively, the goal of tracking is to seek a set of $K$ trajectories $T = \{T_k\}$ according to bayes rule, which maximizes the posterior probability of data association given input detections: $P(T|D) = \frac{P(D|T)P(T)}{P(D)}$, where $P(D)$ is a normalizing constant that does not influence the solution. Assuming the tracks are independent with each other, and the detections are conditionally independent given the tracks, we aim to optimize:

$$
T^* = \arg \max_T p(T) \cdot \prod_i p(d_i|T)
= \arg \max_T \prod_k p(T_k) \cdot \prod_i p(d_i|T_k)
$$

where $p(d_i|T)$ is the likelihood of observing detection $d_i$ within a track, a Bernoulli distribution is modeled with the output of pedestrian detection. $p(T_k)$ denotes the probability of choosing a sequence of detections for track $T_k$. A first-order Markovian assumption is placed for a specific track $T_k$, the probability can be factorized as:

$$
p(T_k) = p_{en}(d_1) \prod_{i=1}^{l-1} p_{tran}(d_{i+1}|d_i) p_{ex}(d_i)
$$

Specifically, $p_{en}(d_1), p_{ex}(d_i)$ denotes the trajectory having length $l$, the flow enters at detection $d_1$ and exits at detection $d_l$, $p_{tran}(d_{i+1}|d_i)$ models the temporal transition prior that detection $d_{i+1}$ follows $d_i$ within a certain trajectory. We formulate the above tracking problem as a minimum cost network-flow problem by taking the negative logarithm of Eq 1, and incorporating the disjoint path constraint as well as flow conservation constraint that flow coming to a node is equal to the flow coming out of a node [60]. As such, the above problem can be converted to a constrained integer linear program:

$$
x^* = \arg \min_x c(f;w)^T x \quad \text{s.t.} \quad A x = b, G x \leq h
$$
3.2. End-to-End Learning of Cost Functions for Min-Cost Flow

Figure 1 illustrates our proposed training pipeline. At the lower level, we solve a linear program, where the cost \( c = [c_{\text{det}}, c_{\text{cen}}, c^{\tau}] \). Specifically, for a detection \( d_i \), \( c^{\tau}_i \) is the detection cost, \( c^\tau_n \) are designed such that a track starts or ends at this detection. \( c^{\tau} \) is a vector consisting of transition costs between two detections. The loss \( L \) at upper-level characterizes the difference between the solution produced by the LP during forward pass and the corresponding ground truth. In order to learn the parameterized cost function \( c(f; w) \) in Eq. 3, we need to calculate the gradient of loss w.r.t. \( w \): \( \frac{\partial L}{\partial w} = \frac{\partial c}{\partial x} \frac{\partial x}{\partial w} \). Therefore, our formulation requires differentiating through the optimal linear program’s solution during training. \( \frac{\partial c}{\partial x} \) are easy to calculate, while computing \( \frac{\partial x}{\partial w} \) is difficult, which requires differentiating through an arg\( \min \) operator. Furthermore, the solutions of the linear program are inherently discrete and need to satisfy certain constraints which further complicates the problem.

Back-propagation through Linear Program Inspired by the work of Amos [1], we propose to back-propagate through linear program’s solution at the optimal Karush–Kuhn–Tucker (KKT) condition, and explicitly adopt the implicit function theorem [3,22] at this condition. Specifically, for our optimization problem in Eq. 3, its Lagrangian is given by:

\[
L(x, \lambda, \nu) = f(x) + \lambda^T (Gx - h) + \nu^T (Ax - b) \quad (4)
\]

where \( f(x) = c^T x \) is the linear objective, \( \lambda \geq 0 \) and \( \nu \) are the dual variables which correspond to inequality and equality constraints, respectively. Taking into account stationarity condition, complementary slackness and primal feasibility of the convex problem’s KKT condition and apply implicit function theorem on top of these equations, we can get the following matrix equation [3]:

\[
\begin{bmatrix}
\nabla^2_x f(x) & G^T & \nu^T \\
\text{diag}(\lambda)G & \text{diag}(Gx - h) & 0 \\
A & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\frac{dx}{de} \\
\frac{dx}{dw} \\
\frac{dc}{de} \\
\frac{dc}{dw}
\end{bmatrix}
= \begin{bmatrix}
\nabla^2_x L(x, \lambda, \nu) \\
0 \\
0 \\
0
\end{bmatrix}
\quad (5)
\]

Here, \( \text{diag}(\cdot) \) operation converts a vector into a diagonal matrix. By solving Eq. 5, the desired Jacobian \( \frac{dc}{de} \) can be acquired. However, directly doing so is not feasible, since \( \nabla^2_x f(x) \) would become 0 due to the linear objective. As a result, the left-hand side of Eq. 5 would become singular, and \( \frac{dx}{de} \) would be trivial.

In order to tackle this problem, we propose to add a Tikhonov damping term \( \gamma \) into the original linear objective so that \( f(x) = c^T x + \gamma \|x\|_2^2 \), and further relax the \( x \) such that \( x \in [0, 1] \) to enable gradient-based optimization. Therefore, the original linear program in Eq. 3 essentially becomes a continuous quadratic program (QP):

\[
\hat{x} = \arg \min_{x \in X} \frac{1}{2} x^T Q x + c^T x \quad (6)
\]

s.t. \( Ax = b, Gx \leq h \)

In particular, \( Q \in \mathbb{R}^{n \times n} \), and \( Q \succ 0 \), so the quadratic objective is strictly convex. Since \( \nabla^2 f(x) = Q \), the new linear system can be written as Eq. 7. By solving this KKT equation during the forward pass, we can get the desired Jacobian \( \frac{dc}{de} \) and back-propagate the QP in order to perform gradient-based end-to-end training.

\[
\begin{bmatrix}
Q & G^T & A^T \\
\text{diag}(\lambda)G & \text{diag}(Gx - h) & 0 \\
A & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\frac{dx}{de} \\
\frac{dx}{dw} \\
\frac{dc}{de} \\
\frac{dc}{dw}
\end{bmatrix}
= \begin{bmatrix}
\nabla^2_x L(x, \lambda, \nu) \\
0 \\
0 \\
0
\end{bmatrix}
\quad (7)
\]

As for the lower-level’s loss, we employ \( L_2 \) loss, which directly measures the difference between the predicted data association \( \hat{x} \) and the ground truth assignment \( x^\text{gt} \) as \( \|\hat{x} - x^\text{gt}\|_2^2 \). Note that the other loss functions between two binary vector, e.g., hamming loss, can also be used here. Our experiment shows that our framework is not very sensitive to the tuned value of \( \gamma \) as long as \( \gamma \) is small. Therefore, we set \( \gamma = 0.1 \). The full training algorithm is detailed in Algorithm 1.

**Algorithm 1** Gradient descent for end-to-end learning of cost function for network flow.

**Input**: Training set \( D_{\text{train}} = \{ (f_i, x^\text{gt}_i) \}_{i=1}^N \) with \( N \) flow graphs, each paired with feature representation \( f_i \) and ground truth data association \( x^\text{gt}_i \).

**Output**: MLP with learned optimal model parameters \( w^* \).

1. Initialize learning rate \( \alpha \), MLP with \( w_0 \)
2. repeat (For each iteration \( t \))
3. Randomly sample \( M \) graphs from \( D_{\text{train}} \).
4. for \( i = 1 \) to \( M \) do
5. Forward \( f_i \) to MLP to obtain cost \( c_i \), and solve Eq. 6 to get \( \hat{x}_i \), calculate loss: \( L(\hat{x}_i, x^\text{gt}_i) \)
6. Calculate gradient of \( L \): \( \frac{dc}{dw_i} = \frac{dc}{dx_i} \frac{dx}{dw_i} \)
7. end for
8. Set \( \frac{dc}{dw} = \frac{1}{M} \sum_{i=1}^M \frac{dc}{dw_i} \), perform gradient descent step: \( w_{t+1} = w_t + \alpha \frac{dc}{dw} \)
9. until Convergence
10. return Learned MLP parameterized by \( w^* \)

3.3. Network Flow Cost Function

Considering trained MLPs with parameters \( w^* \), it is possible to design the unary and binary potentials defined in the flow graph for inference.
Detection Cost. Given a detection $d_i$, the unary cost $c_i^{det}$ is defined as $-s_i$, where $s_i$ is the detection confidence output by a class-specific classifier. This term favors high-confidence person detections to be selected in the tracking results.

Entry/Exit Cost. These costs are learned scalars such that during linear program’s inference, a longer-track is more likely to be selected. We cross-validated on the training set during linear program’s inference, a longer-track is more likely to be selected. Note that a high entry/exit cost trivially yields all-zero solution of the LP as it increases total cost/energy.

Transition Cost. Given a pair of detections $d_i^t$ and $d_i^{t+1}$ and their edge feature representation $f_{ij}$, their matching probability is: $p_{trans}(d_j^t|d_i^t) = MLP(f_{ij}; w^*)$. The corresponding item in $c^{trans}$ is set to $-\log p_{trans}(d_j^t|d_i^t)$, which means detection pairs with high matching probabilities should be connected.

4. Experiments

4.1. Datasets

We conduct the experiments on MOT16, MOT17 [37] and MOT20 [15] pedestrian tracking dataset. MOT16 and MOT17 contain the same videos, except that MOT16 applies DPM [18] detections as input for tracking, while MOT17 evaluates tracking performance under three different detection inputs, namely DPM [18], FRCNN [42] and SDP [58]. Further, MOT20 has been designed to challenge the tracking algorithm’s ability to track crowded scenes [15].

Since MOT16 and MOT17 has almost similar ground truth annotation, we train our model on the MOT16 training set. We use MOT16-09, MOT16-13 sequence to form our validation set and the remaining 5 sequences as the training set. For fair comparison with other methods, the tracking performance is reported in MOT16, MOT17 and MOT20 test set using the provided public detections.

4.2. Implementation Details

Detections. As network-flow based methods are very sensitive to false positives. We first pre-process the raw input detections using [5] provided detections as suggested by [25], in order to obtain a set of high quality detections.

Features. Suppose we have a pair of detections $d_i$ and $d_j$ with $(t_i, x_i, y_i, w_i, h_i, s_i)$ and $(t_j, x_j, y_j, w_j, h_j, s_j)$ respectively, we follow the work of [8] to encode spatial-temporal constraint as the geometric feature: 

$$2(x_i-x_j) / h_i + h_j, 2(y_i-y_j) / h_i + h_j, \log h_i, \log w_j.$$  

This constraint encodes important relative position information between two detections, the intuition is that under a small time gap, the pedestrians cannot move far away, and the size of the pedestrian should not change much due to markovian property.

In addition to the above mentioned spatial-temporal features, we also incorporate appearance features of person detections. For this, we adopt the pre-trained deep ReID [61] architecture, which is the state-of-the-art model in the person re-identification literature, and use the pre-trained model to extract accurate appearance feature for each detection. The normalized cosine distance between two detection’s appearance features output by the ReID network $\phi$ is concatenated with the above mentioned feature together with the generalized intersection over union (GIoU) [43] metric, which is more robust to geometric deformation, as the final edge feature for regression:

$$f_{ij} = \frac{2(x_i-x_j)}{h_i + h_j}, \frac{2(y_i-y_j)}{h_i + h_j}, \log h_i, \log w_j, \phi(d_i^t) \phi(d_j^t), \text{GIoU}(d_i^t, d_j^t).$$  

(8)

Training. In order to generate training samples, we subdivide each training sequence equally into (overlapping) $T$ frames, where we set $T = 15$. Note that we only consider connections of nodes in two adjacent frames to avoid heavy memory consumption and speed up calculation. We utilize the available ground-truth annotations which defines the ideal data association between objects across frames. In particular, for each item in $c^{trans}$ we define it as $-\log p_{trans}(d_j^t|d_i^t)$, where $p_{trans}(d_j^t|d_i^t) \in [0, 1]$ is the matching probability for ground-truth bounding box $d_i$ and $d_j$. $-1$ is used for entries in $c^{det}$ as each annotated box is a true positive. For each ground-truth box $d_i$, its corresponding entry/exit cost is set as: $c_i^{entr} = c_i^{exit} = 1$, so that a track can start and terminate at any detection.

We experimented with different network architectures for scoring the cost function, e.g., linear classifier, MLP, and found that a two-layer multi-layer perceptron works better in practice. Therefore, we adopt a two layer MLP with ReLU non-linearity which outputs a probability distribution between 0 and 1 as affinity measure, using Adam optimizer with initial learning rate of $10^{-3}$ with a weight decay of $10^{-4}$ for around 10 epochs. We select the model that performs best on the validation set for tracking at test time.

Inference. Tracking is performed over (overlapping) batches with length of 50-150 frames depending on the detection density in a specific video. Within each batch, a maximum of $\Delta = 5$ frames gap between detections is allowed to join detections, to handle false negatives and short term occlusions. We utilize Gurobi’s solver to solve each ILP within each batch to obtain tracks. Tracks are finally stitched across adjacent batches to form the final tracks.

Long-Term Occlusion Handling. Due to long-term occlusions in reality, tracks are usually fragmented and identity switches occur. In response, we propose to handle these issues using a second round of network-flow tracker, except that the nodes now are tracklets (short tracks) instead of nodes of the first round, with appearance and motion constraints. Specifically, given tracklet $T_i$, we use [61] to
extract appearance features, e.g. \( f_{i}^{app} = \frac{1}{T} \sum_{t=1}^{T} f_{d}^{app} \) denotes its final representation. Therefore for track \( T_i \) and \( T_j \), \( 1 - f_{i}^{appT} f_{j}^{app} \) is their final matching cost. Regarding motion constraint, we estimate the average velocity for each pair of temporally non-overlapping tracklet, such that \( t_{dist} \) is used to reject physically implausible connections. As such, tracklets are joined and long-term occlusions within a track could be recovered through bi-linear interpolation.

**Post-Processing.** It is possible that our tracker lost several objects during tracking, due to the drastic appearance change/illumination, etc. In order to address this problem, we add a single object tracker (SOT) to keep track of the lost objects. Specifically, for a track that disappears before the video reaches to the last frame, we make use of the last position of the tracker to initialize the single object tracker [36] and perform tracking. In order to kill the tracker in case the tracked object confuses with other objects, we compare the tracked object’s appearance with the initialized object, whenever their similarity of color histogram fall below a threshold \( \tau \), the tracker is terminated to avoid tracking false positives or drift to background. From Figure 2, by adding a SOT, our method successfully tracks the lost object, and the SOT tracker ends when a complete occlusion occurs.

### 4.3. Evaluation

In order to validate the effectiveness of the proposed end-to-end learning method, we compare our learned MLP against a baseline method, which is trained with binary cross-entropy (BCE) objective using annotated data association, which does not allow back-propagation from association. \( L_1 \) and \( L_2 \) are the proposed end-to-end learning approaches which adopt \( L_1 \) and \( L_2 \) as the loss, respectively. All methods takes the same edge features as input for a fair comparison.

**Evaluation of Tracking.** Multiple Object Tracking Accuracy (MOTA) [6] and IDF1 [45] are the two widely adopted metrics for tracking performance evaluation. The two metrics are defined as: \( \text{MOTA} = 1 - \sum_{t} \text{FP}_{t} + \text{FN}_{t} + \text{IDS}_{t} \sum_{t} \text{GT}_{t} \) and \( \text{IDF1} = \frac{2\text{IDTP}}{2\text{IDTP} + \text{IDFP} + \text{IDFN}} \). MOTA measures the tracking error in terms of number of false positive (FP), false negative (FN) and identity switches (IDS), it focus more on the detection’s perspective, while IDF1 [45] reflects the tracker’s ability to maintain identities over time and thus concentrates more on tracking side. In addition, Multiple Object Tracking Precision (MOTP), the percentage of Mostly Tracked (ML) objects and the percentage of Mostly Lost (ML) objects as well as the number of total fragmentations within a trajectory (Frag) constitutes the main evaluation metrics for tracking.

**Effectiveness of the Learned Affinity Measures.** Area Under Curve (AUC) is suitable to measure a binary classifier’s performance, therefore a stronger MLP should obtain a higher AUC score. Mean Squared Error (MSE) is adopted to compare the QP’s output \( x^* \), including the detection, entry/exit cost, against ground-truth annotation \( x^{gt} \).

MSE Edge solely takes into account the predicted data association compared with ground truth association. From Table 1, it is clear that by back-propagating through multi-frame data association, our proposed approach, regardless of the losses used, outperforms baseline method trained using BCE objective in all metrics used. Overall, \( L_1 \) and \( L_2 \) loss achieve similar performance.

<table>
<thead>
<tr>
<th>Loss</th>
<th>AUC↑</th>
<th>BCE↓</th>
<th>MSE↓</th>
<th>MSE Edge↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCE</td>
<td>0.996</td>
<td>0.064</td>
<td>0.026</td>
<td>0.017</td>
</tr>
<tr>
<td>( L_1 )</td>
<td>0.997</td>
<td>0.047</td>
<td>0.013</td>
<td>0.008</td>
</tr>
<tr>
<td>( L_2 )</td>
<td>0.997</td>
<td>0.005</td>
<td>0.010</td>
<td>0.006</td>
</tr>
</tbody>
</table>

Table 1. Evaluation of the affinity metrics and data association results using the proposed training strategy versus baseline methods on the MOT16 validation set.

<table>
<thead>
<tr>
<th>Method</th>
<th>MOTA↑</th>
<th>IDF1↑</th>
<th>IDS↓</th>
<th>MT↑</th>
<th>ML↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o 2nd-round MCF</td>
<td>38.9</td>
<td>43.5</td>
<td>134</td>
<td>22</td>
<td>57</td>
</tr>
<tr>
<td>w 2nd-round MCF</td>
<td><strong>42.9</strong></td>
<td>55.1</td>
<td><strong>73</strong></td>
<td>28</td>
<td>51</td>
</tr>
</tbody>
</table>

Table 2. Ablation study on MOT16 validation set, first-round MCF achieves decent performance, adding second-round MCF further boosts the performance due to long-term occlusion handling.

<table>
<thead>
<tr>
<th>Method</th>
<th>MOTA↑</th>
<th>IDF1↑</th>
<th>MT↑</th>
<th>ML↓</th>
<th>FP↓</th>
<th>FN↓</th>
<th>IDS↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>49.33</td>
<td>58.91</td>
<td>100</td>
<td>163</td>
<td>822</td>
<td>24814</td>
<td>157</td>
</tr>
<tr>
<td>( L_1 )</td>
<td>51.45</td>
<td>60.15</td>
<td>123</td>
<td>151</td>
<td><strong>810</strong></td>
<td>23715</td>
<td>186</td>
</tr>
<tr>
<td>( L_2 )</td>
<td><strong>51.54</strong></td>
<td>60.15</td>
<td><strong>124</strong></td>
<td><strong>147</strong></td>
<td>831</td>
<td><strong>23641</strong></td>
<td>193</td>
</tr>
</tbody>
</table>

Table 3. Evaluation of tracking performance under different approaches on MOT17 validation set. End-to-end learned cost performs better than baseline trained using BCE loss.

<table>
<thead>
<tr>
<th>Method</th>
<th>MOTA↑</th>
<th>REC↑</th>
<th>PREC↑</th>
<th>MT↑</th>
<th>IDS↓</th>
<th>FRAG↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNF <a href="Linear">50</a></td>
<td>28.25</td>
<td>38.01</td>
<td>80.09</td>
<td>9.67</td>
<td>342</td>
<td>1620</td>
</tr>
<tr>
<td>DNF <a href="MLP">50</a></td>
<td>31.10</td>
<td>37.53</td>
<td>85.88</td>
<td>8.51</td>
<td>289</td>
<td>1562</td>
</tr>
<tr>
<td>Proposed Method</td>
<td><strong>44.27</strong></td>
<td><strong>45.03</strong></td>
<td><strong>98.84</strong></td>
<td><strong>14.9</strong></td>
<td><strong>260</strong></td>
<td><strong>365</strong></td>
</tr>
</tbody>
</table>

Table 4. Evaluation of tracking performance under different training strategy on MOT16 training set, ↑ means high number is better, ↓ is opposite. Best performance under each metric is shown in bold-faced font.
Effectiveness of Second Stage Network Flow Data Association. In order to outline the contribution of the first-stage data association on the final tracking performance, we test the proposed method on MOT16 validation set. According to the results in Table 2, the first stage NF data association provides decent results (as a good tracklet initialization) for the second round of network flow tracker, where long-term occlusions must be handled. We can also see the performance improvements in second stage network flow tracking compared to solely having the first stage tracking. In terms of MOTA, the improvements are not as significant as IDF1/IDS, which properly reflects the contribution of the second stage to the long-term occlusions. Therefore the core method (i.e. the first stage) plays a crucial rule to the final performance.

Effectiveness of The Proposed Approach over Baseline in Tracking. In Table 3, we compare our method against a NF baseline trained using BCE objective, in terms of tracking metrics. Note that for all three methods, we use the same strategy of second-round MCF for long-term occlusion handling, thus the difference of final tracking performance merely stems from the first-stage’s association results. Though our method is slightly inferior in IDS metric, our proposed approach surpass the baseline in MOTA, and ML metrics, which suggests back-propagation from data association is indeed necessary to reach a stronger tracking performance.

Comparison against [50]. We compare our proposed learning method against the approach proposed by Schulter et al. [50] in Table 4. We followed the same train/val split as their work and cross-validated on the MOT16 training set for fair comparison. For all tracking metrics, we outperform their end-to-end learning method. The higher recall and precision can be attributed to the fact that we perform end-to-end training of data association, we achieve better MOTA and IDF1 compared to those in [50].

Table 5, 6 and 7 compares our method, namely LinNet, against existing methods. On MOT16 benchmark, our method has second-best performance in MOTA, IDF1, MT and FN. Note that the work of [8] use Message Passing Networks that learns a better feature representation for temporal connections, while our method does not utilize Graph Networks, and our result is still comparable with their approaches.

Comparison with Other State-of-the-arts. We compare our tracker’s performance against other tracking algorithms on MOT16, MOT17 and MOT20 benchmark. In our final results, we selected the models trained using $L_2$ loss for benchmark evaluation. We do not apply SOT in our final implementation as the improvements are marginal.

Table 5. Tracking results on MOT16 test set with DPM [18] detections as input. The best and second best performances are shown in bold and underlined numbers respectively.

Table 6. Tracking results on MOT17 test set using public detections [18, 42, 58] as input. The best and second best performances are shown in bold and underlined numbers respectively.

Table 7. Tracking results on MOT20 test sequences with public detections. Bold and underlined numbers indicate the best and the second best performances.

In Table 3, we compare our method against the approach proposed by Schulter et al. [50] in Table 4. We followed the same train/val split as their work and cross-validated on the MOT16 training set for fair comparison. For all tracking metrics, we outperform their end-to-end learning method. The higher recall and precision can be attributed to the fact that we perform end-to-end training of data association, we achieve better MOTA and IDF1 compared to those in [50].

Table 5, 6 and 7 compares our method, namely LinNet, against existing methods. On MOT16 benchmark, our method has second-best performance in MOTA, IDF1, MT and FN. Note that the work of [8] use Message Passing Networks that learns a better feature representation for temporal connections, while our method does not utilize Graph Networks, and our result is still comparable with their approaches.

On MOT17 benchmark, the work of DeepMOT [57] trains a neural network using the MOT Metric in an end-to-end manner. However their method is mainly optimized for improving two-frame data association. In contrast with their method, our method works in an offline manner which incorporates longer temporal contextual information both during training and inference. The results show that our method achieves better MOTA and IDF1 compared to those in [57]. A notable strong baseline is [39], which leverages a Graph Convolution Network followed by a Sinkhorn Network to perform end-to-end training of data association, we achieve similar performance in terms of MOTA compared to their approach but have better IDF1 and IDS score, due to the
advantage of accurately learned costs as well as the multi-frame data association formulation. It is expected by further incorporating Graph Convolution Network (GCN) into feature learning, the performance of our method would be better. Figure 3 shows some qualitative results, our method is able to track objects through long-term occlusions and recover missing detections.

Compared with the current SOTA method LifT [25], our method achieves slightly worse MOTA and IDS metric. It should be noted that, the work of LifT considers a lifted connections between detections that span over 50 frames, making the formulation NP-hard and computationally heavy. By contrast, our method has polynomial time complexity. Although our method is slightly worse in terms of performance, we achieve faster inference speed: our tracker consumes 1-5 mins while their ILP solver requires 26.6 mins per sequence on average. We believe by working with a more powerful optimization technique such as multi-cut, multiple-hypothesis tracking during inference, our performance could be further leveraged.

Finally, we test our method on MOT20 [15], which is designed for tracking the crowds. It is worth to mention that various pruning heuristics are applied in state-of-the-art method [26], in order to make their NP-hard problem tractable, while our method does not require complex pre-processing steps to sparsify the graph. Overall, our method achieves second-best on MOTA and IDF1 metrics, which are slightly inferior than [25] but have better IDs metric than their results. Thanks to the multi-frame data association used during training/inference, our performance largely exceeds [5, 39, 48] in MOTA and MT metrics.

5. Conclusion

In summary, we have presented a general framework and a novel training method for learning the cost functions of the min-cost flow multi-object tracking problem. By solving a differentiable continuous quadratic program (QP), our approach is able to incorporate multi-frame data association results as well as tracking specific constraints in order to obtain a better global objective for tracking. Although we perform tracking using network flow linear program inference, other formulations that respect tracking constraints can be employed for end-to-end learning as well. One major limitation of our method is that, only data association is learned end-to-end, whilst the part of object detection is separated from training. Since the success of this tracker largely relies on the quality of input detections, in the future we plan to explore training object detection jointly with our network flow framework. We also aim to consider if the end-to-end learning of a higher order optimization objective can further improve the tracking performance.
References


[27] Xiaolong Jiang, Peizhao Li, Yanjing Li, and Xiantong Zhen. Graph neural based end-to-end data association frame-


