Learning Non-Uniform Hypergraph for Multi-Object Tracking

Longyin Wen,[1]* Dawei Du,[2]* Shengkun Li,[2] Xiao Bian,[3] Siwei Lyu[2]
[1]JD Finance, Mountain View, CA, USA
[2]University at Albany, State University of New York, NY, USA
[3]GE Global Research, NY, USA
longyin.wen@jd.com, ddu@albany.edu, sli29@albany.edu, bian.xiao@ge.com, slyu@albany.edu

Abstract

The majority of Multi-Object Tracking (MOT) algorithms based on the tracking-by-detection scheme do not use higher order dependencies among objects or tracklets, which makes them less effective in handling complex scenarios. In this work, we present a new near-online MOT algorithm based on non-uniform hypergraph, which can model different degrees of dependencies among tracklets in a unified objective. The nodes in the hypergraph correspond to the tracklets and the hyperedges with different degrees encode various kinds of dependencies among them. Specifically, instead of setting the weights of hyperedges with different degrees empirically, they are learned automatically using the structural support vector machine algorithm (SSVM). Several experiments are carried out on various challenging datasets (*i.e.*, PETS09, ParkingLot sequence, SubwayFace, and MOT16 benchmark), to demonstrate that our method achieves favorable performance against the state-of-the-art MOT methods.

Introduction

Multi-object tracking (MOT) is an important problem in computer vision with many applications, such as surveillance, behavior analysis, and sport video analysis. Although the performance of MOT has been significantly improved in recent years (Choi 2015; Kim et al. 2015; Wen et al. 2016; Tang et al. 2017), it is still a challenging problem due to factors such as missed detections, false detections, and identification switches.

An automatic MOT system usually employs a pre-trained object detector to locate candidate object regions in each frame, then match the detections across frames to form target trajectories. Most existing methods only consider the pairwise dependencies of detections (*e.g.*, (Rezatofighi et al. 2015; Dehghan, Assari, and Shah 2015; Milan, Schindler, and Roth 2016; Fagot-Bouquet et al. 2016)), and do not take full advantage of the high-order dependencies among multiple targets across frames. This strategy is less effective when nearby objects with similar appearance or motion patterns occlude each other in the video. Several recent methods (Kim et al. 2015; Collins 2012; Shi et al. 2014; Kim et al. 2015; Wen et al. 2014; 2016) attempt to use the high-order information to improve the tracking performance, such as dense structure search on hypergraph (Wen et al. 2014; 2016), tensor power iterations (Shi et al. 2014), high-order motion constraints (Collins 2012; Butt and Collins 2013), and multiple hypothesis tracking (Kim et al. 2015). However, the aforementioned methods merely exploit fixed degrees of dependencies among objects, which limits the flexibility of the hypergraph model[1] in complex environments, and calls for adaptive dependency patterns. As shown in Figure 1, 3-uniform hypergraph is unable to describe the dependencies between two tracklets of target 1 and 4 correctly. On the contrary, non-uniform hypergraph better adapts to different degrees of dependencies among tracklets, and achieves more reliable performance.

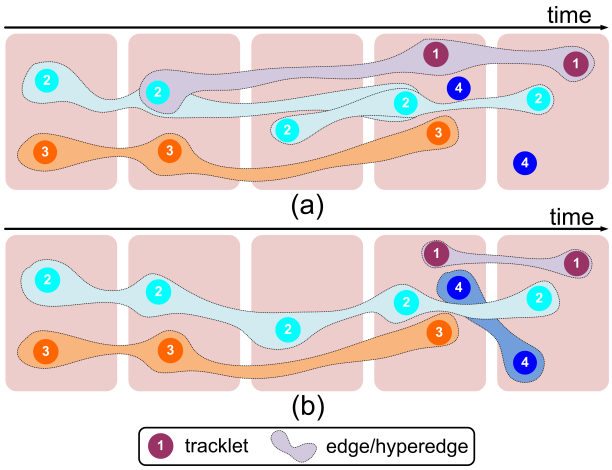

Figure 1: (a) Two previous methods using 3-uniform hypergraph H^{2}T (Wen et al. 2014) and FH^{2}T (Wen et al. 2016), often fails to describe the dependencies among tracklets, when occlusion or missed detection happen. (b) The proposed method uses the non-uniform hypergraph to encode different degrees of dependencies among tracklets effectively.

In this paper, we describe a new *non-uniform* hypergraph

*Equal contribution.

[1]A hypergraph is a generalization of a conventional graph where an edge can join more than two nodes.

learning based tracker (NT), which has much stronger descriptive power to accommodate different tracking scenarios than the conventional graph (Dehghan, Assari, and Shah 2015) or uniform hypergraph (Wen et al. 2014; 2016). The nodes in the hypergraph correspond to the tracklets[2], and the hyperedges with different degrees encode similarities among tracklets to assemble various kinds of appearance and motion patterns. The tracking problem is formulated as searching dense structures on the non-uniform hypergraph. Different from previous methods (Wen et al. 2014; 2016), we do not fix the degree of the hypergraph model, but mix hyperedges of different degrees and learn their relative weights automatically from the data using the structural support vector machine (SSVM) method (Joachims, Finley, and Yu 2009). We propose an efficient approximation algorithm to exploit the dense structures to form long object trajectories to complete the tracking task. In addition, to achieve both accuracy and efficiency, we use a near-online strategy for MOT, *i.e.*, we perform the dense structure searching on the non-uniform hypergraph to generate short tracklets in a temporal window, and then associate those short tracklets to the tracked targets to get the final trajectories of targets at the current time stamp. This process is carried out repeatedly to complete the tracking task in a video.

The main contributions are summarized as follows. (1) We propose a non-uniform hypergraph learning based near-online MOT method, which assembles the hyperedges with different degrees to encode various types of dependencies among objects. (2) The weights of hyperedges with different degrees in the non-uniform hypergraph are learned from data using the SSVM algorithm. (3) We propose an efficient approximation algorithm to complete the dense structure searching problem on the non-uniform hypergraph.

Related Work

MOT methods can be roughly classified into three categories, 1) online strategy, 2) off-line processing strategy, and 3) near-online strategy. If there occurs an error in tracking, it is hard for online strategy (*e.g.*, (Yang et al. 2014; Xiang, Alahi, and Savarese 2015; Yoon et al. 2016)) to recover from due to imprecise appearance or motion measurements. Thus, many algorithms focus on off-line strategy (*e.g.*, (Berclaz et al. 2011; Tang et al. 2017; Milan, Schindler, and Roth 2016)). To make the association step efficient, (Berclaz et al. 2011) formulate the association as a constrained flow optimization problem, solved by the k-shortest paths algorithm. (Tang et al. 2017) present a graph-based formulation that links and clusters person hypotheses over time by solving an instance of a minimum cost lifted multicut problem. In addition, Milan *et al.* (Milan, Schindler, and Roth 2016) pose MOT as minimization of a unified discrete-continuous energy function using the L-BFGS and QPBO algorithms. However, as only association between pairs of detections in local temporal domain are considered, the aforementioned methods do not perform well when multiple similar objects appear in proximity with clutter backgrounds.

To alleviate this problem, (Dehghan, Assari, and Shah 2015) use a graph to integrate all the relations among objects in a batch of frames and formulate the MOT problem as a Generalized Maximum Multi Clique problem on the graph. (Wen et al. 2014) exploit the motion information to help tracking and formulate MOT as the dense structure searching on a uniform hypergraph, in which the nodes correspond to tracklets and the edges encode the high-order dependencies among tracklets. To further improve the efficiency, an approximate RANSAC-style approach is proposed in (Wen et al. 2016) to complete the dense structure searching.

Besides, (Choi 2015) designs a near-online strategy, which inherits the advantages of both online and offline approaches. The tracking problem is formulated as a data-association between targets and detections in a temporal window, that is performed repeatedly at every frame. In this way, the algorithm is able to fix any association error made in the past when more detections are provided. (Wang and Fowlkes 2015) present an end-to-end framework to learn parameters of min-cost flow for MOT problem using a tracking-specific loss function in the SSVM framework. Nevertheless our approach uses the non-uniform hypergraph to describe the high-order dependencies among tracklets, and uses SSVM framework to learn the weights of the hyperedges with different degrees.

Non-uniform Hypergraph

Definition. A hypergraph is a generalization of a conventional graph, where an edge can join more than two nodes. We use $\mathcal{G}(V, \mathcal{E}, \mathcal{A})$ to denote a (weighted) hypergraph, where $V = \{v_1, \cdots, v_n\}$ is the node set, v_i is the i-th node and n is the total number of nodes, $\mathcal{E}$ is the set of hyperedges, and $\mathcal{A}$ is the affinity set corresponding to the edges/hyperedges. Specifically, we define $\mathcal{E} = E_1 \cup \cdots \cup E_D$, where $E_1 = \{(v_1), \cdots, (v_n)\}$ is the set of self-loops, $E_2 \subseteq V \times V$ is the set of conventional graph edges, $E_d \subseteq V^d$ is the set of hyperedges with degree d, $d = 3, \cdots, D$, and D is the maximal degree of hyperedges. If all hyperedges in $\mathcal{G}$ have the same cardinality d, $\mathcal{G}$ is a d-uniform hypergraph (*i.e.*, $E_{d'} = \emptyset$ for $d' \neq d$); otherwise, $\mathcal{G}$ is a non-uniform hypergraph. For node v, we denote its neighborhood as $\mathcal{N}(v)$, which is the set of nodes connected to v.

Similar to (Wen et al. 2016), we define a dense structure on $\mathcal{G}$ as a sub-hypergraph that has the maximum affinities combining all hyperedges, edges and self-loops of nodes. We introduce an indicator variable $\mathbf{y} = (y_1, \cdots, y_n)^\top$, such that $\sum_{i=1}^{n} y_i = 1$, and $y_i = \{0, 1/\alpha\}$, where α is the number of nodes in the dense structure. The affinity summation of the hyperedges, edges and self-loops of nodes of the dense structure can be calculated as

$$\Theta(\mathbf{y}) = \sum_{d=1}^{D} \lambda_d \sum_{\mathbf{v}_{1:d} \in V} \mathcal{A}(\mathbf{v}_{1:d}) \overbrace{y_1 \cdots y_d}^{d} \qquad (1)$$

where $\mathbf{v}_{1:d} = \{v_1, \cdots, v_d\}$, y_i is the indicator variable corresponding to node v_i ($i = 1, \cdots, d$), *i.e.*, $y_i = 1/\alpha$ if node v_i belongs to the dense structure; otherwise, $y_i = 0$. Thus, $y_1 \cdots y_d$ indicates the confidence of the hyperedge ($d > 2$),

[2]The terminology "tracklet" indicates a fragment of target trajectory. Notably, the input detection responses in each frame can be treated as tracklets of length one.

edge ($d = 2$), or self-loop ($d = 1$) $\mathbf{v}_{1:d}$ included in the dense structure. Weights $\lambda_1, \cdots, \lambda_D$ are used to balance the significance of different degrees of hyperedges[3]. The affinity summation from degree 1 to D in (1) describes the overall affinity score combining all the hyperedges, edges, and self-loops of the nodes in the dense structure. Thus, we need to maximize the overall affinity score to exploit the dense structures to complete multi-object tracking.

MOT formulation. We use the non-uniform hypergraph to encode the relations among different tracklets. For each video clip, MOT is initialized by the tracklets[4]. Let $\mathbf{T} = \{T_1, \cdots, T_n\}$ be the tracklet set in the video sequence, where T_i is the i-th tracklet. $T_i = \{B_1^i, \cdots, B_{m_i}^i\}$ consists of m_i frame detections, and $B_j^i = (x_j^i, y_j^i, w_j^i, h_j^i, t_j^i)$, where (x_j^i, y_j^i) and (w_j^i, h_j^i) are center location and dimension of the detection, and t_j^i is the corresponding frame index.

We formulate the MOT problem as searching dense structures on a non-uniform hypergraph $\mathcal{G}(V, \mathcal{E}, \mathcal{A})$[5]. We set every node in $\mathcal{G}$ as the starting point, and search the corresponding dense structure from their neighborhoods. Specifically, for a starting point v_s, we initialize the indicator variable $y_i^\circ = \frac{1}{|\mathcal{N}(v_s)|}$, $i = 1, \cdots, |\mathcal{N}(v_s)|$, where $|\mathcal{N}(v_s)|$ is the number of nodes in v_s's neighborhood. For node v_s, the dense structure searching problem is formulated as

$$\begin{aligned} &\operatorname{argmax}_{\mathbf{y}} \textstyle\sum_{d=1}^{D} \lambda_d \sum_{\mathbf{v}_{1:d} \in \mathcal{N}(v_s)} \mathcal{A}(\mathbf{v}_{1:d}) \overbrace{y_1 \cdots y_d}^{d} \\ &\text{s.t. } \textstyle\sum_{i=1}^{|\mathcal{N}(v_s)|} y_i = 1,\ y_s = \frac{1}{\alpha},\ \forall i,\ y_i \in \{0, \frac{1}{\alpha}\}, \end{aligned} \tag{2}$$

where $\mathcal{N}(v_s)$ is the neighborhood of node v_s. Notably, the constraint $y_s = 1/\alpha$ indicates that the node v_s is included in the searched dense structure, and $y_i = 1/\alpha$ indicates that the i-th node in $\mathcal{N}(v_s)$ is included in the searched dense structure, otherwise, $y_i = 0$.

The problem in (2) is a combinational optimization problem, since we cannot know the number of nodes in the dense structure α priorly. To reduce the complexity of this NP-hard problem, we relax the constraint $y_i \in \{0, \frac{1}{\alpha}\}$ to $y_i \in [0, \frac{1}{\alpha}]$. In addition, we set a minimal size of the sub-hypergraph to be a constant number $\hat{\alpha}$ to avoid the degeneracy, *i.e.*, $\hat{\alpha} \leq \alpha$. Thus, the constraint is converted to $y_i \in [0, \frac{1}{\hat{\alpha}}]$. We would like to highlight that the objective function for dense structure exploiting in (Wen et al. 2016) is a specific case of (2), *i.e.*, if we set $\lambda_{d^\star} \neq \mathbf{0}$ for a specific $d^\star \geq 3$, and make $\lambda_d = 0$, $\forall d \neq d^\star$, the non-uniform hypergraph $\mathcal{G}$ will degenerate into a $d^\star$-uniform hypergraph, and the objective in (2) becomes similarly to that in (Wen et al. 2016). The optimization algorithm in (Wen et al. 2016) for uniform hypergraph model cannot be directly applied to solve the problem in (2).

After exploiting the dense structures, the radical post-processing strategy presented in (Wen et al. 2014) is adopted to remove the conflicts among the searched dense structures. Then, we stitch the tracklets in each post-processed dense structures to form the long trajectories.

Enforcing edge/hyperedge constraints. In the practical MOT scenarios, the objects have two physical constraints: 1) one object cannot occupy two different places at a time; 2) the velocity of a object is below certain maximum possible velocity. As such, in constructing the hypergraph, two nodes connected by one edge/hyperedge should not overlap in time, and the distance between the last and first detections of the tracklet should not larger than the maximal distance that can reach with the maximal possible velocity. These two constraints can reduce the number of edges and hyperedges and computational complexity.

Calculating self-loop affinity. We associate a node with a score to reflect its reliability being a true tracklet of an object, *i.e.*, $\mathcal{A}(v_i) = \rho(v_i)$, where $\rho(v_i)$ ($0 \leq \rho(v_i) \leq 1$) is the confident score of the tracklet v_i calculated by averaging the scores of all detections in the tracklet.

Calculating edge affinity. The edges in the hypergraph encode the similarities between two nodes (tracklets), which consists of three terms: HSV histogram similarity $\mathcal{P}_{\text{col}}(v_i, v_j)$, CNN feature similarity $\mathcal{P}_{\text{cnn}}(v_i, v_j)$, and local motion similarity $\mathcal{P}_{\text{mot}}(v_i, v_j)$, *i.e.*, $\mathcal{A}(v_i, v_j) = \big[\mathcal{P}_{\text{col}}(v_i, v_j), \mathcal{P}_{\text{cnn}}(v_i, v_j), \mathcal{P}_{\text{mot}}(v_i, v_j)\big]$.

Specifically, the HSV histogram similarity $\mathcal{P}_{\text{col}}(v_i, v_j)$ is calculated as $\mathcal{P}_{\text{col}}(v_i, v_j) = \chi\big(h^-(v_i), h^+(v_j)\big)$, where $\chi(\cdot, \cdot)$ is the cosine similarity between the HSV histograms of the detections in the last frame of v_i (*i.e.*, $h^-(v_i)$) and the first frame of v_j (*i.e.*, $h^+(v_j)$).

Moreover, the CNN feature similarity $\mathcal{P}_{\text{cnn}}(v_i, v_j)$ is calculated as $\mathcal{P}_{\text{cnn}}(v_i, v_j) = \frac{1+\chi\left(\mu^-(v_i), \mu^+(v_j)\right)}{2}$, where $\mu^-(v_i)$ and $\mu^+(v_j)$ are the CNN features of the detections in the last frame of v_i and the first frame of v_j.

Finally, the similarity between two bounding boxes based on the generalized KLT tracker (Zhou, Tang, and Wang 2013) is calculated as $\mathcal{P}_{\text{mot}}(v_i, v_j) = 1 - \frac{2}{1+\exp\left(\frac{2 \cdot \zeta(v_i, v_j)}{\gamma(B_{m_i}^i) + \gamma(B_1^j)}\right)}$, where $\gamma(B_{m_i}^i)$ and $\gamma(B_1^j)$ are the areas of the detections in the last frame of v_i and the first frame of v_j, and $\zeta(v_i, v_j)$ is the number of point trajectories generated by KLT tracker across the bounding boxes of both the first frame of v_i and first frame of v_j.

Calculating hyperedge affinity. We count the number of local point trajectories passing through the regions of $\mathbf{v}_{1:d}$ to calculate the affinities of hyperedges, which encodes the mo-

[3]Notably, in this paper, we use the terminology "affinity" to indicate the value associated to each edge/hyperedge, which reflects the similarities of the nodes in the corresponding edge/hyperedge. Meanwhile, the terminology "weight" is adopted to indicate the numbers used to balance the significance of different degrees of hyperedges, edges and self-loops in dense structure searching. The weights of d-th hyperedges may consist of $\kappa > 1$ terms (*e.g.*, the weights of the second degree hyperedges may consist of the appearance similarity and motion consistency between two tracklets). In such cases, the weight λ_d is a vector with the size $1 \times \kappa$, and the affinity $\mathcal{A}(\mathbf{v}_{1:d})$ is also a vector with the size $\kappa \times 1$.

[4]Our definition of tracklet generalizes cases for single detection, *i.e.*, $m_i = 1$, or continuous sequence of detections, *i.e.*, the frame index set $\{t_1^i, \cdots, t_{m_i}^i\}$ corresponding the detections on the tracklet, where t_j^i is an integer, and $t_j^i < t_{j+1}^i$, $j = 1, \cdots, m_i - 1$.

[5]Specifically, we only consider the edges/hyperedges with no duplicate nodes, *i.e.*, each edge/hyperedge contains different nodes.

tion consistency of tracklets $\mathbf{v}_{1:d}$. Thus, for the i-th hyperedge with degree d, the affinity is calculated as $\mathcal{A}(\mathbf{v}_{1:d}) = 1 - \frac{2}{1+\exp\left(\frac{d\cdot\zeta(\mathbf{v}_{1:d})}{\sum_{u=1}^{d}\sum_{j=1}^{l_u}\gamma(B_j^u)}\right)}$, where $\zeta(\mathbf{v}_{1:d})$ measures the number of local point trajectories crossing all regions of $\mathbf{v}_{1:d}$, l_u is the length of tracklet v_u, B_j^u is the j-th detection on v_u, and $\gamma(B_j^u)$ is the area of the detection B_j^u.

Near-online tracking. It is difficult to handle all detections in a long video sequences at a time, since it requires large memory and computation sources to construct non-uniform hypergraphs and perform dense structure search on all detections. In order to achieve both accuracy and efficiency, inspired by (Choi 2015), we use a near-online strategy for MOT. Specifically, after getting τ video frames at time t, we construct a *non-uniform* hypyergraph to describe the hybrid orders of dependencies among detections and search the dense structures on the hypergraph to generate short tracklets in the temporal window $[t-\tau, t]$. Then, we construct a conventional graph[6] to describe the associations between the tracked targets and the short tracklets within $[t-\tau, t]$. After that, we perform the dense structure searching on the conventional graph to associate the short tracklets and the tracked targets to get the final trajectories at the current time stamp. This process is carried out repeatedly every τ frames to complete the tracking task in the whole video.

Inference

For efficiency, we use the simple pairwise update algorithm (Liu et al. 2012) to solve the dense structure searching problem on hypergraph $\mathcal{G}$ corresponding to node v_s in (2). We first form the Lagrangian of the problem as

$$\begin{aligned}\mathcal{L}(\mathbf{y}, a, \mathbf{b}, \mathbf{c}) &= \Theta(\mathbf{y}) - a\cdot\big(\textstyle\sum_{i=1}^{|\mathcal{N}(v_s)|} y_i - 1\big) \\ &+ \textstyle\sum_{i,i\neq v_s} b_i\cdot y_i + \sum_{i,i\neq v_s} c_i\cdot(\frac{1}{\hat{\alpha}} - y_i),\end{aligned} \tag{3}$$

where a, $\mathbf{b} = (b_1, \cdots, b_{|\mathcal{N}(v_s)|})$, and $\mathbf{c} = (c_1, \cdots, c_{|\mathcal{N}(v_s)|})$ are Lagrangian multipliers with $a \geq 0$, $b_i \geq 0$, and $c_i \geq 0$, $i = 1, \cdots, |\mathcal{N}(v_s)|$. Any local maximizer $\mathbf{y}^*$ of the objective function must satisfy the Karush-Kuhn-Tucker (KKT) conditions (Kuhn and Tucker 1951), *i.e.*,

$$\begin{cases} \frac{\partial\Theta(\mathbf{y}^*)}{\partial y_i} - a + b_i - c_i = 0, i \neq v_s; \\ \sum_{i,i\neq v_s} y_i^*\cdot b_i = 0; \\ \sum_{i,i\neq v_s} c_i\cdot(\frac{1}{\hat{\alpha}} - y_i^*) = 0; \\ a \geq 0, b_i \geq 0, c_i \geq 0, i = 1, \cdots, |\mathcal{N}(v_s)|; \\ \sum_{i=1}^{n} y_i = 1, y_i \in [0, \frac{1}{\hat{\alpha}}]. \end{cases} \tag{4}$$

We define $\phi_i(\mathbf{y}) = \frac{\partial\Theta(\mathbf{y})}{\partial y_i}$ as *reward* at node v_i, which is calculated as

$$\begin{aligned}\phi_i(\mathbf{y}) \quad &= \lambda_1\mathcal{A}(i) \\ &+ \textstyle\sum_{d=2}^{D}\lambda_d\sum_{\mathbf{v}_{1:d-1}\in\mathcal{N}(v_s)}\mathcal{A}(\mathbf{v}_{1:d-1}, i)\prod_{j=1}^{d-1} y_{v_j}.\end{aligned}$$

Since $\forall i$, $y_i^* \geq 0$, $b_i \geq 0$, $\sum_{i,i\neq v_s} y_i^*\cdot b_i = 0$, we have that if $y_i^* > 0$, then $b_i = 0$. Meanwhile, since $\forall i$, $c_i \geq 0$, and $y_i^* \leq \frac{1}{\hat{\alpha}}$, we have that if $0 < y_i^* < \frac{1}{\hat{\alpha}}$, then $c_i = 0$. In this way, for node $i \neq v_s$, the KKT conditions can be further rewritten as

$$\phi_i(\mathbf{y}) = \begin{cases} \leq a, & y_i^* = 0, i \neq v_s; \\ = a, & 0 < y_i^* < \frac{1}{\hat{\alpha}}, i \neq v_s; \\ \geq a, & y_i^* = \frac{1}{\hat{\alpha}}, i \neq v_s. \end{cases} \tag{5}$$

Based on $\mathbf{y}$ and α, we can partition the solution space into three disjoint subsets, $\Omega_1(\mathbf{y}) = \{i | y_i = 0\}$, $\Omega_2(\mathbf{y}) = \{i | y_i \in (0, \frac{1}{\hat{\alpha}})\}$, and $\Omega_3(\mathbf{y}) = \{i | y_i = \frac{1}{\hat{\alpha}}\}$. Thus, similar to Theorem 1 in (Liu et al. 2012), we find that there exists an appropriate a, such that (1) the rewards at all node in $\Omega_1(\mathbf{y})$ are no larger than a; (2) the rewards at all nodes in $\Omega_2(\mathbf{y})$ are equal to a; and (3) the rewards at all nodes in $\Omega_3(\mathbf{y})$ are larger than a.

A simple pairwise updating method is used to optimize (2). That is, we can increase one component y_p and decrease another one y_q appropriately, to increase the objective $\Theta(\mathbf{y})$. To be specific, we first introduce another variable y_l' that is defined as: $y_l' = y_l$, for $l \neq p$ and $l \neq q$; $y_l' = y_l + \eta$, for $l = p$; and $y_l' = y_l - \eta$, for $l = q$, where $\mathbf{y}' = (y_1', \cdots, y_{|\mathcal{N}(v_s)|}')$ is the updated indicator variable in optimization process. Then, the change of objective after updating is

$$\begin{aligned}\Delta\Theta(\mathbf{y}) &= \Theta(\mathbf{y}') - \Theta(\mathbf{y}) \\ &= \varphi_{p,q}(\mathbf{y})\cdot\eta^2 + \big(\phi_p(\mathbf{y}) - \phi_q(\mathbf{y})\big)\cdot\eta,\end{aligned} \tag{6}$$

where $\varphi_{p,q}(\mathbf{y}) = -\lambda_2\cdot\mathcal{A}(p,q) - \sum_{d=3}^{D}\lambda_d\sum_{\mathbf{v}_{1:d-2}\neq p,q}\mathcal{A}(\mathbf{v}_{1:d-2}, p, q)\prod_{j=1}^{d-2} y_{v_j}$.

To maximize the objective difference $\Delta\Theta(\mathbf{y})$, we select the updating step η as follows[7]:

$$\eta = \begin{cases} \min(y_q, \frac{1}{\hat{\alpha}} - y_p), \text{ if } \varphi_{p,q}(\mathbf{y}) \geq 0; \\ \min\big(y_q, \frac{1}{\hat{\alpha}} - y_p, \frac{\phi_q(\mathbf{y}) - \phi_p(\mathbf{y})}{2\cdot\varphi_{p,q}(\mathbf{y})}\big), \text{ if } \varphi_{p,q}(\mathbf{y}) < 0; \\ \min(y_q, \frac{1}{\hat{\alpha}} - y_p), \text{ if } \phi_p(\mathbf{y}) = \phi_q(\mathbf{y}), \varphi_{p,q}(\mathbf{y}) > 0. \end{cases} \tag{7}$$

We use a heuristic strategy to compute a local maximizer $\mathbf{y}^*$ of (2), *i.e.*, gradually select pairs of nodes (v_p, v_q) to maximize the increase of $\Theta(\mathbf{y})$ by updating the indicator variable $\mathbf{y}$ based on the updating step η calculated by (7). Specifically, from (6) and (7), we find that (1) if $\phi_p(\mathbf{y}) > \phi_q(\mathbf{y})$, there exists a such that the objective $\Theta(\mathbf{y})$ can be increased by updating $\mathbf{y}$ based on (6); (2) when $\phi_p(\mathbf{y}) = \phi_q(\mathbf{y})$ and $\varphi_{p,q}(\mathbf{y}) > 0$, the objective $\Theta(\mathbf{y})$ can be increased by increasing either y_p or y_q, and decreasing the other one; (3) when $\phi_p(\mathbf{y}) = \phi_q(\mathbf{y})$ and $\varphi_{p,q}(\mathbf{y}) = 0$, the objective $\Theta(\mathbf{y})$ will not be affected by changing $\mathbf{y}$.

Thus, in each iteration, we can select node v_p with the largest reward from set $\Omega_1 \cup \Omega_2$, *i.e.*, $v_p \in \Omega_1 \cup \Omega_2$, and node v_q with the smallest reward from set $\Omega_2 \cup \Omega_3$, *i.e.*, $v_q \in \Omega_2 \cup \Omega_3$, satisfying $\phi_p(\mathbf{y}) > \phi_q(\mathbf{y})$, to increase $\Theta(\mathbf{y})$ by increasing y_p and decreasing y_q with an appropriate η in (7). This process is iterated until the reward of v_p equals to v_q. If $\Theta(\mathbf{y})$ can not be increased according to (6), then $\mathbf{y}$ is already a local maximizer. The overall procedure is summarized in Algorithm 1.

[6]The conventional graph is a special case of the non-uniform hypergraph, which only includes the conventional edges in the graph, *i.e.*, $D = 2$.

[7]In general, we can assume $\phi_p(\mathbf{y}) > \phi_q(\mathbf{y})$. When $\phi_p(\mathbf{y}) < \phi_q(\mathbf{y})$, we can exchange indexes p and q to maximize $\Delta\Theta(\mathbf{y})$. Please see the supplementary material for more details.

Algorithm 1 Compute the local maximizer $\mathbf{y}^*$

Input: The affinity set $\mathcal{A}$ corresponding to the hyperedges in $\mathcal{G}$, the starting point $\mathbf{y}^\circ = (y_1^\circ, \cdots, y_{|\mathcal{N}(v_s)|}^\circ)$ and the minimal size of sub-hypergraph $\hat{\alpha}$.

Initialize the indicator variable $\mathbf{y} = \mathbf{y}^\circ$.
while $\mathbf{y}$ is the local maximizer **do**
  Select $v_p \in \Omega_1 \cup \Omega_2$ with the largest reward $\phi_p(\mathbf{y})$;
  Select $v_q \in \Omega_2 \cup \Omega_3$ with the smallest reward $\phi_q(\mathbf{y})$;
  if $\phi_p(\mathbf{y}) > \phi_q(\mathbf{y})$ **then**
    Compute η according to (7), update $\mathbf{y}$ and the corresponding rewards.
  else if $\phi_p(\mathbf{y}) = \phi_q(\mathbf{y})$ **then**
    Find another pair of nodes (v_i, v_j) satisfying $\varphi_{i,j}(\mathbf{y}) > 0$ and $\phi_i(\mathbf{y}) = \phi_j(\mathbf{y})$, where $v_i \in \Omega_1 \cup \Omega_2$ and $v_j \in \Omega_2 \cup \Omega_3$.
    if such a pair exists **then**
      Compute the corresponding η according to (7).
      Update $\mathbf{y}$ and the corresponding rewards.
    else
      $\mathbf{y}$ is already a local maximizer, *i.e.*, $\mathbf{y}^* = \mathbf{y}$.
    end if
  end if
end while

Output: The local maximizer indicator variable $\mathbf{y}^*$.

Learning

Instead of selecting the weights $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_D)$ in (1) empirically, we use a structured SVM (Joachims, Finley, and Yu 2009) to learn $\boldsymbol{\lambda}$ automatically from the training data. Specifically, given a set of ground-truth bounding boxes of objects in the j-th training video ($1 \leq j \leq U$, where U is the total number of training videos), we aim to recover the trajectories of objects, which is equivalent to cluster the input bounding boxes into several groups. That is to obtain the indicator variables of the clusters $\mathbf{Y}_j = (\mathbf{y}_{1,j}, \cdots, \mathbf{y}_{k_j,j})$, where $\mathbf{y}_{i,j}$ ($1 \leq i \leq k_j$) is the indicator variable of the i-th target, and k_j is the total number of targets in the video. The bounding boxes in each group belong to the same target.

The function defined in (1) can be rewritten as a linear function of $\boldsymbol{\lambda}$, *i.e.*, $\Theta(\mathbf{Y}_j) = \boldsymbol{\lambda}^\top \cdot \mathcal{S}(\mathbf{Y}_j)$, where

$$\begin{aligned}\mathcal{S}(\mathbf{Y}_j) \quad &= \big[\textstyle\sum_{\varsigma=1}^{k_j} \sum_{v_i \in V} \mathcal{A}(v_i) y_{\varsigma,i}, \cdots, \\ &\quad \textstyle\sum_{\varsigma=1}^{k_j} \sum_{\mathbf{v}_{1:D} \in V} \mathcal{A}(\mathbf{v}_{1:D}) \prod_{i=1}^{D} y_{\varsigma,i}\big].\end{aligned}$$

We aim to find the optimal weights $\boldsymbol{\lambda}$ by maximizing the objective function $\Theta(\mathbf{Y}_j)$ with the same input object detections. Then, the objective using a SSVM with margin rescaling is formulated as

$$\begin{aligned} &\min_{\boldsymbol{\lambda}} \tfrac{1}{2}\|\boldsymbol{\lambda}\|_2 + C \cdot \textstyle\sum_{j=1}^{U} \xi_j, \\ \text{s.t.} \quad &\boldsymbol{\lambda}^\top \Big(\mathcal{S}(\mathbf{Y}_j^*) - \mathcal{S}(\mathbf{Y}_j)\Big) + \xi_j \geq \Delta(\mathbf{Y}_j, \mathbf{Y}_j^*), \\ &\xi_j \geq 0, \;\; j = 1, \cdots, U. \end{aligned} \tag{8}$$

Intuitively, this formulation requires that the score $\boldsymbol{\lambda}^\top \cdot \mathcal{S}(\mathbf{Y}_j^*)$ of any ground-truth annotated video must be larger than the score $\boldsymbol{\lambda}^\top \cdot \mathcal{S}(\mathbf{Y}_j)$ of any other results $\mathbf{Y}_j$ by the loss $\Delta(\mathbf{Y}_j, \mathbf{Y}_j^*)$ minus the slack variable ξ_j. The constant C adjusts the importance of minimizing the slack variables. The loss function $\Delta(\mathbf{Y}_j, \mathbf{Y}_j^*)$ measures how incorrect $\mathbf{Y}_j$ is according to the weighted Hamming loss in (Wang and Fowlkes 2015). Meanwhile, the SSVM formulation in (8) has exponential number of constraints for each training sequence. We use a cutting plain algorithm (Joachims, Finley, and Yu 2009) to solve this problem, which has time complexity linear in the number of training examples.

Experiments

We conduct experiments on several popular MOT evaluation datasets, *i.e.*, the multi-pedestrian tracking (Wen et al. 2016) (including the PETS09 and ParkingLot sequences), MOT2016 (Milan et al. 2016), and multi-face tracking (Wen et al. 2016) datasets, to evaluate the performance of the proposed MOT method (denoted as NT subsequently)[8]. We use the MOT2016-train set to train the set-to-set recognition model (Liu, Yan, and Ouyang 2017) to calculate the CNN feature similarity, and the multi-pedestrian tracking dataset to analyze the influence of the degree of hypergraph to tracking performance. In addition, we conduct the ablation study to demonstrate the effectiveness of non-uniform hypergraph and SSVM learning.

Evaluation Metrics. Following previous MOT methods, we use the widely adopted multi-object tracking accuracy (MOTA) metric (Bernardin and Stiefelhagen 2008) to compare the performance of the trackers. MOTA is a cumulative measure combing false negatives (FN), false positives (FP), and identity switches (IDS). We report mostly tracked (MT), mostly lost (ML), FP, FN, IDS, and the fragmentation of the tracked objects (FM) to measure a tracker comprehensively. In addition, for the multi-pedestrian and multi-face tracking datasets (Wen et al. 2016), we also report the multi-object tracking precision (MOTP) score, which computes the total error of tracked positions comparing with the manually annotated ground-truth, with normalization to the hit/miss threshold value. Following the evaluation protocol in MOT2016, we use the ID F1 score (IDF1) (Ristani et al. 2016) instead of MOTP, which is the ratio of correctly identified detections over the average number of ground-truth and computed detections.

Parameters. We conduct an experiment to select the maximal degree of the hypergraph D. We set $D = 2, \cdots, 5$ while keeping other parameters fixed, and denote the resulting models as NT_d(2), $\cdots$, NT_d(5). For each maximal degree, we use the sequences in the training set of MOT2016 to learn the weights of different degrees of hyperedges $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_D)$ using SSVM, and use the sequences in multi-pedestrian tracking dataset for testing. The uniform average performance of the trackers in multi-pedestrian tracking dataset is presented in Table 1. Specifically, we divide each sequence in the MOT2016 train-set into non-overlapping sequences of 14 frames. And then, we take the detections that have more than 50% overlap with the ground-truth as true detections to collect training samples for the weights $\boldsymbol{\lambda}$ learning.

As shown in Table 1, NT achieves the best performance with the maximal degree $D = 4$, indicated by higher MOTA

[8]The source code of the proposed method is available at https://github.com/longyin880815.

Table 1: Comparisons of variants of the proposed NT tracker on multi-pedestrian tracking dataset.

Variants	D	$\boldsymbol{\lambda}$	MOTA	MOTP	IDS	FM
NT_d(2)	2	learned	67.5	62.4	103.7	92.2
NT_d(3)	3	learned	68.8	64.5	83.8	76.2
NT_d(4)	4	learned	**68.9**	**65.0**	68.3	68.8
NT_d(5)	5	learned	68.5	64.7	**61.5**	**63.7**
NT_r(4)	4	learned, $\lambda_i = 0, i = 3$	68.4	63.5	72.7	74.2
NT_r(5)	5	learned, $\lambda_i = 0, i = 3, 4$	67.6	63.5	64.3	66.0
NT_e(2)	2	$\lambda_i = 1, i = 1, 2$	67.1	62.6	103.7	87.0
NT_e(3)	3	$\lambda_i = 1, i = 1, \cdots, 3$	67.5	63.7	103.3	87.5
NT_e(4)	4	$\lambda_i = 1, i = 1, \cdots, 4$	67.4	63.7	104.0	86.7
NT_e(5)	5	$\lambda_i = 1, i = 1, \cdots, 5$	67.1	64.6	93.2	81.7

and lower IDS and FM scores. We notice that the performance of NT decreases when $D > 4$, this may be because the hypergraph with excessive high degree fails to describe the motion patterns of objects well, particularly for the objects moving fast with drastic variations of directions. Thus, we set $D = 4$ in our experiments, and the learned weights of different degree of hyperedge are $\lambda_1 = 0.58535$, $\lambda_2 = [0.15576, 3.0332, 0.34388]$, $\lambda_3 = 1.2879$, and $\lambda_4 = 0.22324$. The batch size τ in near-online tracking is set to 7. The minimal size of the sub-hypergraph is set as $\hat{\alpha} = 2$. We fix all parameters to these values in the experiments.

Ablation Study. To demonstrate the contribution of non-uniform hypergraph, we construct two variants of the proposed NT tracker by removing the hyperedges with certain degrees, *i.e.*, NT_r(3) and NT_r(4), and evaluate them on the multi-pedestrian tracking dataset (Wen et al. 2016), shown in Table 1. The results in Table 1 shows that removing the hyperedges with degrees 3 and 4 will negatively affect the performance (*i.e.*, reduce 0.5% and 0.9% MOTA scores), which shows that exploiting different degrees of dependencies among objects is important for MOT performance.

Besides, to demonstrate the contribution of SSVM, in Table 1, we present the performance of non-uniform hypergraph based trackers with equal weights of different degrees of hyperedges in multi-pedestrian tracking, denoted as NT_e(2), $\cdots$, NT_e(5). The NT_d(i) methods perform consistently better than the NT_e(i) methods with the same maximal degrees, *e.g.*, NT_d(2) *vs.* NT_e(2), and NT_d(5) *vs.* NT_e(5), where $i = 2, \cdots, 5$. The results show that using SSVM to learn the weights of hyperedges of different degrees can improve the performance.

Multi-Pedestrian Tracking. We perform experiments for the multi-pedestrian tracking on five sequences from the PETS09 dataset (Ellis and Ferryman 2010): S2L1 (795 frames), S2L2 (436 frames), S2L3 (240 frames), S1L1-1 (221 frames), and S1L1-2 (241 frames), and ParkingLot sequence from (Zamir, Dehghan, and Shah 2012) (996 frames). These sequences are captured in the crowded surveillance scenes with frequent occlusions, abrupt motion, illumination changes, etc. Following (Wen et al. 2016; Andriyenko, Schindler, and Roth 2012), we report the uniform average scores on different metrics over sequences of the proposed NT algorithm, as well as five state-of-the-art trackers, *i.e.*, KSP (Berclaz et al. 2011), DPMF (Pirsiavash, Ramanan, and Fowlkes 2011), CEM (Andriyenko and Schindler 2011), DCT (Andriyenko, Schindler, and Roth 2012) and FH^{2}T (Wen et al. 2016), in Table 2. The tracking results of previous methods are taken from (Wen et al. 2016). For fair and comprehensive comparisons, we use the same frame detections, ground-truth annotations as well as the evaluation protocol provided by the authors of (Wen et al. 2016). We train the set-to-set recognition method (Liu, Yan, and Ouyang 2017) based on the pre-trained GoogLeNet (Szegedy et al. 2015) on the training set of MOT2016 to extract the CNN features of the detections.

Table 2: Comparison of the proposed tracker with the previous trackers in multi-pedestrian tracking sequences.

Method	MOTA	MOTP	MT[%]	ML[%]	FP	FN	IDS	FM
KSP	45.5	67.1	33.4	35.6	107.8	2223.2	**42.2**	**49.8**
DPMF	51.6	**70.0**	21.5	27.0	**68.8**	1897.0	61.8	80.7
CEM	55.7	66.6	30.1	21.7	127.3	1652.8	63.7	56.7
DCT	58.1	67.6	43.1	21.3	119.5	1610.2	64.2	53.2
FH^{2}T	66.2	64.9	54.3	14.7	194.5	1150.8	45.2	73.7
NT	**68.9**	65.0	**58.2**	**9.6**	252.7	**974.3**	68.3	68.8

As shown in Table 2, we find that our NT tracker performs better than the state-of-the-art methods on several important metrics (*e.g.*, MOTA, MT, and ML). Specifically, NT improves 2.7% and 3.9% average MOTA and MT scores, and reduces 5.1% average ML score, against the second best tracker FH^{2}T (Wen et al. 2016). This may be attributed to that our method uses non-uniform hypergraph instead of uniform hypergraph in (Wen et al. 2016), especially for tracking in crowded scenes with different motions and frequent occlusions of objects. By the way, we notice that the FH^{2}T method (Wen et al. 2016) performs better than the methods (*e.g.*, DPMF (Pirsiavash, Ramanan, and Fowlkes 2011) and DCT (Andriyenko, Schindler, and Roth 2012)), both only considering the similarities between pairs of tracklets (*i.e.*, FH^{2}T (Wen et al. 2016) produces 14.6% and 8.1% higher average MOTA score than DPMF (Pirsiavash, Ramanan, and Fowlkes 2011) and DCT (Andriyenko, Schindler, and Roth 2012)), which indicates that exploiting the high-order similarities among multiple tracklets is crucial for MOT.

MOT2016 Benchmark. The MOT2016 benchmark (Milan et al. 2016) is a collection of 14 video sequences (7/7 for training and testing, respectively), with a relatively high variations in object movements, camera motion, viewing angle and crowd density. The benchmark primarily focuses on pedestrian tracking. The ground-truths for testing set are strictly invisible to all methods, *i.e.*, all results on testing set were submitted to the respective testing servers for evaluation. We use the training set to learn the parameters of the proposed algorithm, and submit our results on testing set for evaluation, shown in Table 3. For a fair comparison with the state-of-the-art MOT methods, we use the reference object detections provided by the benchmark (Milan et al. 2016). We train the set to set recognition method (Liu, Yan, and Ouyang 2017) based on the pre-trained GoogLeNet (Szegedy et al. 2015) on the training set of MOT2016 to ex-

Table 3: Comparison of the proposed tracker with the state-of-the-art trackers in the test set of the MOT2016 benchmark (accessed on 08/18/2018).

Method	MOTA	IDF1	MT[%]	ML[%]	FP	FN	IDS	FM	Hz
online:									
EAMTT	38.8	42.4	7.9	49.1	8,114	102,452	965	1,657	**11.8**
DCCRF	44.8	39.7	14.1	42.3	5,613	94,133	968	1,378	0.1
STAM	46.0	50.0	14.6	43.6	6,895	91,117	473	1,422	0.2
AMIR	47.2	46.3	14.0	41.6	2,681	92,856	774	1,675	1.0
offline:									
Quad	44.1	38.3	14.6	44.9	6,388	94,775	745	1,096	1.8
INT	45.4	37.7	18.1	38.7	13,407	85,547	600	930	4.3
MHT	45.8	46.1	16.2	43.2	6,412	91,758	590	781	0.8
NLPa	47.6	47.3	17.0	40.4	5,844	89,093	629	768	8.3
FWT	47.8	44.3	19.1	38.2	8,886	85,487	852	1,534	0.6
LMP	**48.8**	51.3	18.2	40.1	6,654	86,245	481	595	0.5
near-online:									
NOMT	46.4	**53.3**	18.3	41.4	9,753	87,565	**359**	**504**	2.6
Ours	47.5	43.6	**19.4**	**36.9**	13,002	**81,762**	1,035	1,408	0.8

tract the CNN features of the detections.

In Table 3, NT is compared with the state-of-the-art methods including EAMTT (Sanchez-Matilla, Poiesi, and Cavallaro 2016), Quad (Son et al. 2017), MHT (Kim et al. 2015), STAM (Chu et al. 2017), NOMT (Choi 2015), AMIR (Sadeghian, Alahi, and Savarese 2017), NLPa (Levinkov et al. 2017), FWT (Henschel et al. 2017), LMP (Tang et al. 2017), INT (Lan et al. 2018), and DCCRF (Zhou et al. 2018). Our NT method performs on par with the state-of-the-art trackers (*e.g.*, FWT and LMP) in terms of tracking accuracy. Specifically, LMP uses additional person re-identification datasets to train a deep StackNet with body part fusion to associate pedestrians across frames, achieving the top tracking accuracy (*i.e.*, 48.8% MOTA), while FWT incorporates multiple detectors to improve the tracking performance. In contrast to the aforementioned methods using complex appearance model, our NT algorithm focuses on exploiting different degrees of dependencies among tracklets to assemble various kinds of appearance and motion patterns. The appearance modeling strategies proposed in those methods are complementary to our NT tracker. Meanwhile, we notice that NT achieves better performance than the high-order information based MHT in terms of tracking accuracy (47.5% *vs.* 45.8%), which implies that exploiting adaptive dependencies among objects is important for MOT.

Multi-Face Tracking. In addition to pedestrian tracking, we also evaluate NT on the SubwayFaces dataset used in (Wen et al. 2016). The dataset consists of four sequences, namely S001, S002, S003, and S004 with $1,199$, $1,000$, $1,600$, and $1,001$ frames, captured from surveillance videos in subway with manually annotations. We compare our approach with five state-of-the-art MOT algorithms, *i.e.*, CEM (Andriyenko and Schindler 2011), KSP (Berclaz et al. 2011), DCT (Andriyenko, Schindler, and Roth 2012), DPMF (Pirsiavash, Ramanan, and Fowlkes 2011) and FH^{2}T (Wen et al. 2016), with uniform average scores on different metrics over sequences presented in Table 4. We use the same input detections, ground-truth annotations and the evaluation protocol as (Wen et al. 2016), and the results of the state-of-the-art trackers in Table 4 are taken from (Wen et al. 2016). We use pre-trained AlexNet (Krizhevsky, Sutskever, and Hinton 2012) to extract the CNN features of the detected faces.

Table 4: Comparison of the proposed tracker with other state-of-the-art trackers in the SubwayFace dataset.

Method	MOTA	MOTP	MT[%]	ML[%]	FP	FN	IDS	FM
CEM	18.9	71.4	18.8	37.4	1185.3	4095.3	69.8	100.3
KSP	32.8	**74.0**	15.1	32.2	648.5	3589.3	70.0	82.3
DCT	37.6	73.7	25.5	12.6	1235.0	2691.0	66.3	59.3
DPMF	42.6	73.7	24.6	14.3	679.0	2858.3	62.8	74.0
FH^{2}T	45.8	73.4	27.4	11.5	742.3	2634.0	43.0	57.3
NT	**53.1**	70.4	**34.2**	**8.5**	**648.5**	**2292.8**	**37.5**	**36.3**

As presented in Table 4, we find that our approach achieves the best performance on almost all evaluation metrics except MOTP. Specifically, the NT method produces 7.3% and 6.8% larger average MOTA and MT scores, and 3.0% lower average ML score, comparing to the second best FH^{2}T tracker. The evaluated sequences are recorded in the unconstrained scenes with fast motion, illumination variations, motion blurs and frequent occlusions. Since different degrees of dependencies among objects are considered, our method is able to exploit different types of motion patterns to improve the tracking performance, indicated by the consistent highest scores of almost all metrics (*i.e.*, MOTA, MT, ML, FP, FN, IDS, and FM). Meanwhile, comparison with the state-of-the-art methods, our approach tracks the objects more robustly even when occlusions occur, indicated by the IDS, FM and FN scores. However, the linear interpolation is used in our method to estimate the occluded parts of the trajectories, which is not accurate enough to achieve good MOTP score, especially for crowded scenes containing non-linear motion patterns.

Running Time. We implement the NT algorithm in C++ without any code optimization. To demonstrate the running time of NT, we run it five times using a single thread on a *laptop* with a 2.8 GHz Intel processor and 16 GB memory. Given the detections with the corresponding CNN features, the average speeds on the multi-pedestrian tracking dataset, MOT2016 dataset, and multi-face tracking dataset are 7.9, 0.8, and 9.0 frame per second (FPS), respectively.

Conclusions

In this work, we propose a non-uniform hypergraph learning based near-online MOT method, which assembles different degrees of dependencies among tracklets in a unified objective. In contrast to previous graph or hypergraph based methods, our formulation exploit different high-degree cues among multiple tracklets in a computationally efficient way. Extensive experiments on several datasets, including the multi-pedestrian and multi-face tracking datasets, and MOT2016 benchmark, show that our method achieves comparable performance regarding to the state-of-the-arts. For future work, we plan to investigate and compare different optimization strategies to solve the dense structure searching problem on non-uniform hypergraphs.

Acknowledgments

Dawei Du and Siwei Lyu are supported by US NSF IIS-1816227 and the National Natural Science Foundation of China under Grant 61771341.

References

Andriyenko, A., and Schindler, K. 2011. Multi-target tracking by continuous energy minimization. In *CVPR*, 1265–1272.

Andriyenko, A.; Schindler, K.; and Roth, S. 2012. Discrete-continuous optimization for multi-target tracking. In *CVPR*.

Berclaz, J.; Fleuret, F.; Türetken, E.; and Fua, P. 2011. Multiple object tracking using k-shortest paths optimization. *TPAMI* 33(9):1806–1819.

Bernardin, K., and Stiefelhagen, R. 2008. Evaluating multiple object tracking performance: the clear mot metrics. *Journal on Image and Video Processing* 2008:1.

Butt, A. A., and Collins, R. T. 2013. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *CVPR*, 1846–1853.

Choi, W. 2015. Near-online multi-target tracking with aggregated local flow descriptor. In *ICCV*, 3029–3037.

Chu, Q.; Ouyang, W.; Li, H.; Wang, X.; Liu, B.; and Yu, N. 2017. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In *ICCV*, 4846–4855.

Collins, R. T. 2012. Multitarget data association with higher-order motion models. In *CVPR*, 1744–1751.

Dehghan, A.; Assari, S. M.; and Shah, M. 2015. GMMCP tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking. In *CVPR*, 4091–4099.

Ellis, A., and Ferryman, J. M. 2010. PETS2010 and PETS2009 evaluation of results using individual ground truthed single views. In *AVSS*, 135–142.

Fagot-Bouquet, L.; Audigier, R.; Dhome, Y.; and Lerasle, F. 2016. Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In *ECCV*, 774–790.

Henschel, R.; Leal-Taixé, L.; Cremers, D.; and Rosenhahn, B. 2017. A novel multi-detector fusion framework for multi-object tracking. *CoRR* abs/1705.08314.

Joachims, T.; Finley, T.; and Yu, C. J. 2009. Cutting-plane training of structural svms. *Machine Learning* 77(1):27–59.

Kim, C.; Li, F.; Ciptadi, A.; and Rehg, J. M. 2015. Multiple hypothesis tracking revisited. In *ICCV*, 4696–4704.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1106–1114.

Kuhn, H. W., and Tucker, A. W. 1951. Nonlinear programming. *2nd Berkeley Symposium. Berkeley: University of California Press* 481–492.

Lan, L.; Wang, X.; Zhang, S.; Tao, D.; Gao, W.; and Huang, T. S. 2018. Interacting tracklets for multi-object tracking. *TIP* 27(9):4585–4597.

Levinkov, E.; Uhrig, J.; Tang, S.; Omran, M.; Insafutdinov, E.; Kirillov, A.; Rother, C.; Brox, T.; Schiele, B.; and Andres, B. 2017. Joint graph decomposition & node labeling: Problem, algorithms, applications. In *CVPR*, 6012–6020.

Liu, H.; Yang, X.; Latecki, L. J.; and Yan, S. 2012. Dense neighborhoods on affinity graph. *IJCV* 98(1):65–82.

Liu, Y.; Yan, J.; and Ouyang, W. 2017. Quality aware network for set to set recognition. In *CVPR*, 4694–4703.

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

Milan, A.; Schindler, K.; and Roth, S. 2016. Multi-target tracking by discrete-continuous energy minimization. *TPAMI* 38(10):2054–2068.

Pirsiavash, H.; Ramanan, D.; and Fowlkes, C. C. 2011. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 1201–1208.

Rezatofighi, S. H.; Milan, A.; Zhang, Z.; Shi, Q.; Dick, A. R.; and Reid, I. D. 2015. Joint probabilistic data association revisited. In *ICCV*, 3047–3055.

Ristani, E.; Solera, F.; Zou, R. S.; Cucchiara, R.; and Tomasi, C. 2016. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCVW*, 17–35.

Sadeghian, A.; Alahi, A.; and Savarese, S. 2017. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *ICCV*, 300–311.

Sanchez-Matilla, R.; Poiesi, F.; and Cavallaro, A. 2016. Online multi-target tracking with strong and weak detections. In *ECCVW*, 84–99.

Shi, X.; Ling, H.; Hu, W.; Yuan, C.; and Xing, J. 2014. Multi-target tracking with motion context in tensor power iteration. In *CVPR*, 3518–3525.

Son, J.; Baek, M.; Cho, M.; and Han, B. 2017. Multi-object tracking with quadruplet convolutional neural networks. In *CVPR*, 3786–3795.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S. E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*, 1–9.

Tang, S.; Andriluka, M.; Andres, B.; and Schiele, B. 2017. Multiple people tracking by lifted multicut and person re-identification. In *CVPR*, 3539–3548.

Wang, S., and Fowlkes, C. C. 2015. Learning optimal parameters for multi-target tracking. In *BMVC*, 4.1–4.13.

Wen, L.; Li, W.; Lei, Z.; Yi, D.; and Li, S. Z. 2014. Multiple target tracking based on undirected hierarchical relation hypergraph. In *CVPR*, 3457–3464.

Wen, L.; Lei, Z.; Lyu, S.; Li, S. Z.; and Yang, M. 2016. Exploiting hierarchical dense structures on hypergraphs for multi-object tracking. *TPAMI* 38(10):1983–1996.

Xiang, Y.; Alahi, A.; and Savarese, S. 2015. Learning to track: Online multi-object tracking by decision making. In *ICCV*, 4705–4713.

Yang, M.; Liu, Y.; Wen, L.; You, Z.; and Li, S. Z. 2014. A probabilistic framework for multitarget tracking with mutual occlusions. In *CVPR*, 1298–1305.

Yoon, J. H.; Lee, C.-R.; Yang, M.-H.; and Yoon, K.-J. 2016. Online multi-object tracking via structural constraint event aggregation. In *CVPR*, 1392–1400.

Zamir, A. R.; Dehghan, A.; and Shah, M. 2012. GMCP-tracker: Global multi-object tracking using generalized minimum clique graphs. In *ECCV*, 343–356.

Zhou, H.; Ouyang, W.; Cheng, J.; Wang, X.; and Li, H. 2018. Deep continuous conditional random fields with asymmetric inter-object constraints for online multi-object tracking. *TCSVT*.

Zhou, B.; Tang, X.; and Wang, X. 2013. Measuring crowd collectiveness. In *CVPR*, 3049–3056.