Multi-Camera Multi-Target Tracking with Space-Time-View Hyper-graph

Longyin Wen, Zhen Lei, Ming-Ching Chang, Honggang Qi & Siwei Lyu

International Journal of Computer Vision

ISSN 0920-5691

Int J Comput Vis
DOI 10.1007/s11263-016-0943-0

Springer

Int J Comput Vis
DOI 10.1007/s11263-016-0943-0

Multi-Camera Multi-Target Tracking with Space-Time-View Hyper-graph

Longyin Wen[1] · Zhen Lei[2] · Ming-Ching Chang[3] · Honggang Qi[4] · Siwei Lyu[1]

Received: 15 August 2015 / Accepted: 12 August 2016

Abstract Incorporating multiple cameras is an effective solution to improve the performance and robustness of multi-target tracking to occlusion and appearance ambiguities. In this paper, we propose a new multi-camera multi-target tracking method based on a space-time-view hyper-graph that encodes higher-order constraints (i.e., beyond pairwise relations) on 3D geometry, appearance, motion continuity, and trajectory smoothness among 2D tracklets within and across different camera views. We solve tracking in each single view and reconstruction of tracked trajectories in 3D environment simultaneously by formulating the problem as an efficient search of dense sub-hypergraphs on the space-time-view hyper-graph using a sampling based approach. Experimental results on the PETS 2009 dataset and MOTChallenge 2015 3D benchmark demonstrate that our method performs favorably against the state-of-the-art methods in both single-camera and multi-camera multi-target tracking, while achieving close to real-time running efficiency. We also provide experimental analysis of the influence of various aspects of our method to the final tracking performance.

Keywords Multi-camera multi-target tracking · Single-camera multi-target tracking · Space-time-view hyper-graph · Dense sub-hypergraph search

Communicated by Hiroshi Ishikawa, Takeshi Masuda, Yasuyo Kita and Katsushi Ikeuchi.

Electronic supplementary material The online version of this article (doi:10.1007/s11263-016-0943-0) contains supplementary material, which is available to authorized users.

✉ Honggang Qi
hgqi@jdl.ac.cn

Longyin Wen
lwen@albany.edu

Zhen Lei
zlei@nlpr.ia.ac.cn

Ming-Ching Chang
mchang2@albany.edu

Siwei Lyu
slyu@albany.edu

1 Computer Science Department, University at Albany, State University of New York, Albany, USA

2 National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China

3 Computer Engineering Department, University at Albany, State University of New York, Albany, USA

4 School of Computer and Control Engineering, University of the Chinese Academy of Sciences, Beijing, China

1 Introduction

As an important problem in computer vision, multi-target tracking finds wide applications in video surveillance, traffic monitoring and crowd analysis. With the maturity of detection algorithms (Dollár et al. 2012), the current state-of-the-art performance in multi-target tracking is attained with the *tracking-by-detection* methodology, in which reliable detection of short sequence of moving objects or *tracklets* are linked based on their affinities in appearance and motion to form long tracks. Albeit these successes, the majority of existing multi-target tracking algorithms use a single camera view. As such, their performance succumbs to false/miss detections due to target occlusions and ambiguous appearances. Using detections gleaned from different but overlapping camera views, these problems can be effectively solved and the accuracy of multi-target tracking can be significantly improved.

Published online: 06 September 2016

Many previous methods on multi-camera multi-target tracking (Fleuret et al. 2008; Wu et al. 2009; Berclaz et al. 2011; Wu et al. 2011; Liu et al. 2012; Attanasi et al. 2015) entail two consecutive steps: (i) repeated single view *tracking* that finds tracklets in each individual camera view, and (ii) cross-view *reconstruction* of 2D tracklets using 3D geometric constraints[1]. These simple approaches do not take advantage of the fact that single view tracking and cross-view reconstruction provide mutually bootstrapping information: 3D geometric constraints can rule out false detection and improve tracklet linking in each view, while reliable linking of tracklets in individual views can compensate the effect of noise and outliers that often plague the reconstruction step.

An alternative strategy of multi-camera tracking is to jointly solve the tracking and reconstruction problems in a single optimization framework. However, as will be detailed in Sect. 2, two existing methods (Leal-Taixé et al. 2012; Hofmann et al. 2013) using this strategy rely on pairwise association of 3D tracklets (Fig. 2a, b), and do not take full advantage of the strong higher-order correlations among the tracklets across time and space. For instance, as shown in Fig. 1, if there are three 3D tracklets $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ of the same trajectory, but due to ambiguity in appearance of each camera view, tracklet $\mathcal{T}_2$ is not the strongest association with either $\mathcal{T}_1$ or $\mathcal{T}_3$. On contrary, both tracklets $\mathcal{T}_1$ and $\mathcal{T}_3$ are similar to $\mathcal{T}_4$ in appearance. In this case, considering the motion consistency of *three* tracklets in joint is crucial to associate them into correct trajectory ($\mathcal{T}_6$), but pairwise association will lead to wrong linkings (associate tracklets $\mathcal{T}_1$, $\mathcal{T}_3$, and $\mathcal{T}_4$ to generate wrong trajectory $\mathcal{T}_5$). Specifically, the hypothetical trajectory $\mathcal{T}_6$ is more smooth than $\mathcal{T}_5$ with facile motion direction changes, which indicates the more consistency of tracklets $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ than $\mathcal{T}_1$, $\mathcal{T}_3$, and $\mathcal{T}_4$ in motion. Here, the distinction between higher order dependencies and pairwise dependencies among non-consecutive frames should be made clear, i.e., the higher order dependency corresponding to the trajectory smoothness of tracklets $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ can not be simply represented as the pairwise constraints between $\mathcal{T}_1$ and $\mathcal{T}_2$, $\mathcal{T}_2$ and $\mathcal{T}_3$, and $\mathcal{T}_1$ and $\mathcal{T}_3$. Such higher order correlations are particularly useful to handle multi-camera tracking scenarios with severe occlusions.

In this work, we describe a new multi-camera multi-target tracking method based on a weighted hyper-graph that represents higher-order affinities of 2D tracklets, which characterize their consistencies in 3D geometry, appearance, motion continuity and trajectory smoothness. We term this hyper-graph as Space-Time-View hyper-graph (STV hyper-graph). The nodes of STV hyper-graph correspond to potential *3D couplings* of 2D tracklets, which are reconstructed 3D tracklets from 2D tracklets across different views that are potentially associated with the trajectory of the same tracked target (see Fig. 2c for an illustrative example). Geometric consistency of these 2D tracklets in forming a coupling is encoded with the weight of each node. Hyper-edges of STV hyper-graph with their associated weights reflect affinities among the couplings. In order to correctly associate tracklets across multiple views and eventually reconstruct the 3D trajectory, we further perform a search of dense sub-hypergraphs on STV hyper-graph, which correspond to sub-hypergraphs with higher weights over including nodes and hyper-edges, and then accumulatively link couplings in such sub-hypergraphs.

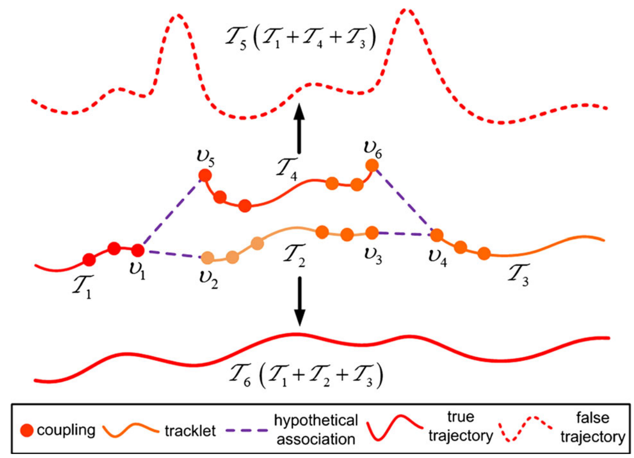

Fig. 1 Example of the advantages of using higher-order dependencies among multiple tracklets instead of the pairwise dependencies. Notably, the more similar of the *colors* of nodes indicates the more similar of the couplings in appearance over all camera views

Contributions The contributions of our work are summarized as follows.

- We introduce STV hyper-graph constructed from input videos of multiple views as a flexible and compact representation for the inference of higher-order correlations among tracklets across space, time, and camera views, which is a generalization of the hyper-graph based single view multi-target tracking method presented in Wen et al. (2014).
- We formulate the multi-camera multi-target tracking problem as searching for dense sub-hypergraphs on STV hyper-graph, which is solved efficiently by the proposed sampling based approximation method. In comparison with the optimization strategy for single-camera tracking in Wen et al. (2014), our method can scale up to the much larger number of correlations in the hyper-graph, which are prerequisite in multi-camera tracking scenario.
- In comparison with previous works on multi-camera multi-target tracking (Leal-Taixé et al. 2012; Hofmann et al. 2013), our method enables the incorporation of

[1] Many methods do not form tracklets but perform association directly on detections in each frame. In this work, we unify these methods by treating individual frame detections as tracklets of length one.

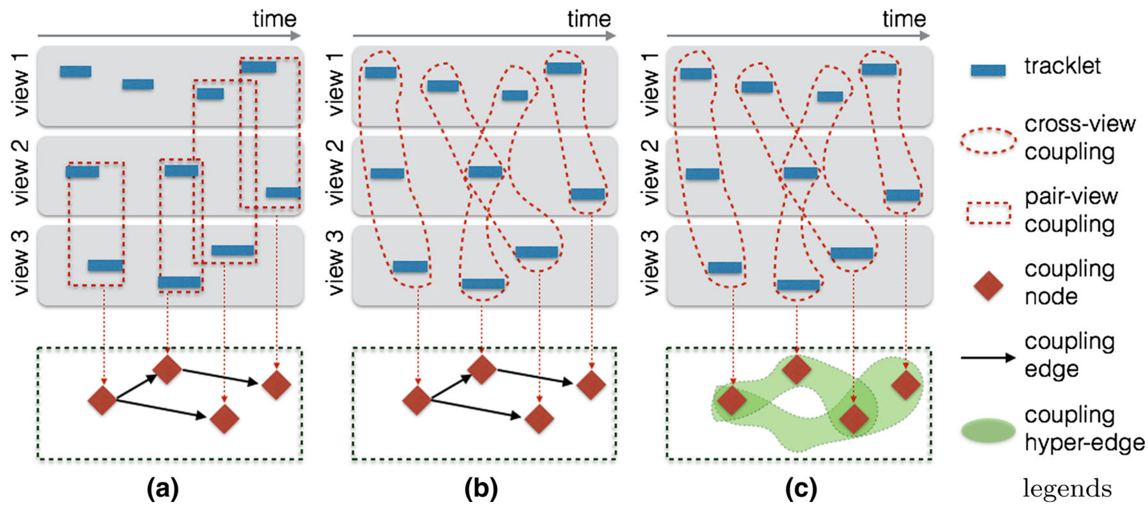

Fig. 2 Models of two previous methods and this work on multi-camera multi-target tracking. For clarity, we illustrate only a partial set of edges and hyper-edges. This figure is better viewed in *color*

higher-order dependencies among the tracklets across both time and space, and is more robust to occlusions and appearance ambiguities.

– Extensive experiments are performed on the PETS 2009 dataset and MOTChallenge 2015 3D benchmark to compare with the state-of-the-art methods, and show improved effectiveness and running efficiency of our method in both single-camera and multi-camera multi-object tracking tasks.

The rest of the paper is organized as follows. In Sect. 2 we review relevant works. In Sect. 3 we describe our method in detail. Experimental results are presented in Sect. 4 and Sect. 5 concludes the paper with discussion of future works.

2 Related Works

2.1 Single-Camera Multi-Target Tracking

A traditional approach to multi-target tracking is to predict the states (i.e., location and size) of tracked targets using Bayesian filtering methods, e.g., Kalman or particle filters (Isard and Blake 1998; Marchesotti et al. 2002; Khan et al. 2005; Smith et al. 2005; Leven and Lanterman 2009; Yang et al. 2014). These methods can track targets state effectively in short durations and run in real-time, but are not effective in handling occlusion and appearance changes that often occur in complex tracking scenarios.

Many recent effective single-camera multi-target tracking methods are based on the tracking-by-detection approach, and formulate tracking as a data association problem. The Joint Probabilistic Data Association Filter (JPDAF) (Hong and Cui 2000) and Multiple Hypotheses Tracking (MHT) (Reid 1979) have been proposed to handle the data association problem efficiently. The JPDAF algorithm focuses on estimating the best assignments between the tracked targets and the detections in a probabilistic framework. Different from frame-by-frame association in JPDAF, MHT computes the likelihoods of all candidate assignments over several time steps. However, the number of candidate assignments grows exponentially with the number of frames, which makes MHT not efficient when handling the long-term association. Yu and Medioni (2009) present a data driven Markov Chain Monte Carlo method to accomplish the data association task in multiple frames. The sampling-based inference algorithm may have long "burn-in" time, i.e., the time to run the Markov chain before we can collect samples from the equilibrium, and difficult to evaluate due to the lack of practical check for convergence.

To handle the long-term association problem, several algorithms have been proposed, which differs in the specific optimization methods used, including network flow (Zhang et al. 2008; Izadinia et al. 2012; Pirsiavash et al. 2011; Chari et al. 2015), K-Shortest Path (KSP) (Berclaz et al. 2011), maximum weight independent set (Brendel et al. 2011), linear programming (Jiang et al. 2007), multi-frame matching (Shu et al. 2012), hierarchical Hungarian algorithm (Huang et al. 2013; Yang and Nevatia 2012a, b), tensor power iteration (Shi et al. 2014), hyper-graph based optimization (Wen et al. 2014). Most of these works merely focus on using the pairwise similarities of 2D detections/tracklets to complete the tracking task, except the method (Wen et al. 2014). In particular, the method of Wen et al. (2014) is the most related work, because it also uses a weighted hyper-graph to represent higher order affinities between 2D tracklets, and directly searches the dense subgraphs on the hyper-graph to solve the tracking problem. However, this method is not suitable for the multi-camera multi-target tracking problem. In particular, STV hyper-graph models relations among 3D tracklets reconstructed from 2D tracklets of multiple views, and nodes in STV hyper-graph have weights reflecting unreliabilities of 3D reconstruction. Such node weights are crucial in our method, which are not addressed in Wen et al. (2014). Furthermore, the number of nodes in STV hyper-graph is several orders of magnitude larger than that used in single view tracking (Wen et al. 2014), which renders the simple optimization algorithm of (Wen et al. 2014) impractical due to high computation and memory requirements.

2.2 Multi-Camera Multi-Target Tracking

Using multiple camera views can potentially improve the performance of multi-target tracking, but it also brings up some challenging issues. In particular, the tracking algorithm must link 2D tracklets and at the same time reconstruct their 3D trajectories. Early multi-camera multi-target tracking methods (Fleuret et al. 2008; Wu et al. 2009, 2011; Berclaz et al. 2011; Liu et al. 2012) usually solve the tracking and reconstruction problems in separate stages, which do not take advantage of the mutually bootstrapping relation between these two tasks.

Recently, two existing works attempt to solve these two problems, i.e., tracking and reconstruction, within a single optimization framework. In Leal-Taixé et al. (2012) (model illustrated in Fig. 2a), multiple graphs are constructed for detections in each view to capture their affinities, and the associations of these detections are encoded with another type of graph that are constructed for each pair of camera views. The overall tracking problem is solved using the Dantzig-Wolfe decomposition and branching algorithm. This method is further improved in Hofmann et al. (2013) (model illustrated in Fig. 2b), where the couplings between 2D detections across two or more camera views are formed, and longer tracks are obtained from a directed graph capturing pairwise dependencies of reconstructed 3D couplings cross the frames. However, in both works, only *pairwise* correlations between candidate associations of 2D detections across camera views are modeled. Accordingly, dependencies among a set of more than two candidate associations of 2D detections across camera views cannot be effectively modeled. If a target fails to appear in any camera view due to occlusion or miss detection, using only pairwise correlations will lead to fragmentations and identity switches, which can significantly deteriorate the overall performance and robustness of the tracking method. Our proposed method is different from these two methods. (1) Unlike Leal-Taixé et al. (2012), our algorithm uses only one global hyper-graph for both reconstruction and tracking, which considers the high-order dependencies among multiple couplings rather than the pairwise dependencies in multiple graphs. (2) Different from Hofmann et al. (2013), the proposed method explores higher-order dependencies among couplings instead of the pairwise dependencies by constructing a hyper-graph. The tracking problem is naturally formulated as the dense subgraph exploiting problem on the hyper-graph, which is solved by the proposed sampling based approximate optimization method.

3 Methodology

Our multi-camera multi-target tracking method is based on the STV hyper-graph, representing the cross-view and temporal associations of detected 2D tracklets in individual camera views. The process starts with the generation of couplings from tracklets in each single camera view (Sect. 3.1) and computation of affinity measures (Sect. 3.2). In Sect. 3.3, we introduce the STV hyper-graph, which is the major data structure to incorporate higher-order dependencies among tracklets. From the STV hyper-graph, trajectories of moving targets are extracted from the dense sub hypergraphs. The details of extracting such dense sub-hypergraphs and forming longer trajectories are provided in Sect. 3.4. The notations used in this paper are listed in Table 1.

3.1 Generating Couplings

We postulate that there are V static camera views, where videos from each view are synchronized with the same frame rate. Furthermore, from each video, tentative short sequences of detected targets (tracklets) are assumed to have been obtained from frame detections (e.g., using Felzenszwalb et al. 2008) or using single-view tracklet linking methods (e.g., Shi et al. 2014; Wen et al. 2014). Throughout this paper, we use v to index camera views, i to index the tracklets, and j to index the detections of a tracklet.

We denote the collection of detected 2D tracklets from the v-th camera view as $\mathbf{T}_v = \{T_1^v, \cdots, T_{n_v}^v\}$. A single 2D tracklet, $T_i^v = \{D_1^{v,i}, \cdots, D_{m_{c,i}}^{v,i}\}$, corresponds to a series of frame detections, $m_{c,i}$ is the number of detections in the tracklet, and $D_j^{v,i} = (t_j^{v,i}, q_j^{v,i})$, where $t_j^{v,i}$ is the frame number of the detection, and $q_j^{v,i} = (x_j^{v,i}, y_j^{v,i}, w_j^{v,i}, h_j^{v,i})$ specifies the bounding box of the detection with center pixel location $(x_j^{v,i}, y_j^{v,i})$ and dimension $(w_j^{v,i}, h_j^{v,i})$. We also use $\mathbf{t}_i^v = \{t_1^{v,i}, \cdots, t_{m_{c,i}}^{v,i}\}$ to denote the set of all frame indices of the corresponding 2D tracklet T_i^v. Our definition of 2D tracklets generalizes cases of single detection (i.e., $|\mathbf{t}_i^v| = 1$), or continuous sequence of detections (i.e., $\mathbf{t}_i^v = \{a, a+1, \cdots, b-1, b\}$ where $a < b$ are two integers). We also consider calibrated cameras with known parameters, where targets are moving on a common ground-plane, such that any 2D pixel location (x, y) in the video frame can be

Table 1 Notations

Symbol	Meaning
V	Number of used camera views.
$\mathbf{T}_v = \{T_1^v, \cdots, T_{n_v}^v\}$	The collection of detected 2D tracklets from the v-th camera view
$T_i^v = \{D_1^{v,i}, \cdots, D_{m_{c,i}}^{v,i}\}$	The i-th tracklet of camera view v, $m_{c,i}$ is the number of detections in the tracklet
$D_j^{v,i} = (t_j^{v,i}, q_j^{v,i})$	The j-th detection in the i-th tracklet of camera view v, and $t_j^{v,i}$ is the frame index of the detection
$q_j^{v,i} = (x_j^{v,i}, y_j^{v,i}, w_j^{v,i}, h_j^{v,i})$	The bounding box of the j-th detection in the i-th tracklet of camera view v, $(x_j^{v,i}, y_j^{v,i})$ is the center pixel location and $(w_j^{v,i}, h_j^{v,i})$ is the dimension.
$\mathcal{T}$	A coupling in the 3D world
k	The degree of the STV hyper-graph
ν_i	The i-th node (tracklet) in the STV hyper-graph.
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	The STV hyper-graph, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the hyper-edge set, i.e., $\mathcal{E} \subset \overbrace{\mathcal{V} \times \cdots \times \mathcal{V}}^{k}$
e	The k-tuple nodes involved in a hyper-edge, *i.e.*, $e = (\nu_1, \cdots, \nu_k)$
$\boldsymbol{\pi}_i$	The i-th connection samples involving $k-1$ nodes
β^*	The minimal size of the searched dense subgraph
$\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$	The approximate STV graph, where $\mathcal{V}^*$ is the same weighted node set as the corresponding STV hyper-graph $\mathcal{G}$, and $\mathcal{E}^* = \mathcal{V}^* \times \mathcal{V}^*$ is edge set describing the supports between the nodes

back projected to the 3D world coordinates (X, Y, Z) using a mapping function, $\phi^v(x, y) = (X, Y, Z)$.

A 3D *coupling* collects 2D tracklets from different camera views that potentially correspond to the same trajectory of a target in the 3D world. Formally, we define a coupling $\mathcal{T}$ as a non-empty subset of $\bigcup_{v=1}^{V} \mathbf{T}_v$, i.e., $\emptyset \neq \mathcal{T} \subset \bigcup_{v=1}^{V} \mathbf{T}_v$ such that *no more than one* tracklet from any camera view will be included, *i.e.*,

$$\left(T_i^v \in \mathcal{T}\right) \wedge \left(T_{i'}^{v'} \in \mathcal{T}\right) \Rightarrow v \neq v'. \tag{1}$$

As such, the maximum total number of unique couplings is given by $\prod_{v=1}^{V}(n_v + 1) - 1$. This number is obtained as the following: from each camera view, at most one tracklet can be included into the coupling, leaving the total choices for one view as $n_v + 1$. The total is given by $\prod_{v=1}^{V}(n_v + 1) - 1$, where the minus one corresponds to the case that no tracklets are chosen from any views.

We add another constraint for a coupling constructed by two or more 2D tracklets, i.e., for each 2D tracklet included in the coupling, it must have overlapping frame indices with *at least one* other 2D tracklets that are also in the coupling, as:

$$T_i^v \in \mathcal{T} \Rightarrow \exists T_{i'}^{v'} \in \mathcal{T} \wedge \mathbf{t}_i^v \cap \mathbf{t}_{i'}^{v'} \neq \emptyset. \tag{2}$$

Note that Eq. (2) is weaker than requiring *all* 2D tracklets to have overlapping time, which is justified by the nature of multi-camera tracking scenario that the target may not be observed in all camera views due to occlusion.

We define the frame indices of a coupling as the union of frame indices of its composing 2D tracklets, as $\mathbf{t}_{\mathcal{T}} = \bigcup_{T_i^v \in \mathcal{T}} \mathbf{t}_i^v$. For each frame of the coupling $t \in \mathbf{t}_{\mathcal{T}}$, we also maintain a data structure that backtracks its composing 2D frame detections at t, as

$$\mathcal{D}_t^{\mathcal{T}} = \{(v, i, j) : T_i^v \in \mathcal{T} \wedge t \in \mathbf{t}_{\mathcal{T}} \wedge t = t_j^{v,i}\}, \tag{3}$$

where Eqs. (1), (2), and (3) together ensures three conditions: (i) 2D tracklet T_i^v must be included in coupling $\mathcal{T}$, (ii) frame index t is in the frame indices of $\mathcal{T}$, and (iii) T_i^v has a detection at frame t. Using $\mathcal{D}_t^{\mathcal{T}}$, we compute the predicted 3D position of the coupling at t as the average 3D positions obtained from the corresponding 2D tracklets,

$$\mathcal{P}_t^{\mathcal{T}} = \frac{1}{|\mathcal{D}_t^{\mathcal{T}}|} \sum_{(v,i,j) \in \mathcal{D}_t^{\mathcal{T}}} \phi^v\left(x_j^{v,i}, y_j^{v,i}\right), \tag{4}$$

and the scattering (uncertainty) of the predicted 3D position,

$$\epsilon_t^{\mathcal{T}} = \frac{1}{|\mathcal{D}_t^{\mathcal{T}}|} \sum_{(v,i,j) \in \mathcal{D}_t^{\mathcal{T}}} \left\| \mathcal{P}_t^{\mathcal{T}} - \phi^v\left(x_j^{v,i}, y_j^{v,i}\right) \right\|^2, \tag{5}$$

where Eq. (5) is used as a measure of coherence of the set of 2D tracklets in forming the 3D trajectory.

3.2 Affinity Among Couplings

The obtained 3D couplings may correspond to segments of a longer 3D trajectory belonging to a moving target being tracked in multiple camera views. To evaluate the likelihood of a set of couplings in forming a longer trajectory, similar to Wen et al. (2014), we introduce three affinity measures for appearances, motion continuity, and trajectory smoothness. As couplings overlapping in time cannot be associated with one target, we set all affinity measures to zero in that case. Subsequently, we consider only couplings with no overlapping frame indices in the appearance Eq. (6), motion continuity Eq. (7), and trajectory smoothness Eq. (8) affinities calculation.

Appearance affinity The appearance affinity between a pair of non-overlapping couplings $\mathcal{T}$ and $\mathcal{T}'$, with $\mathcal{T}$ preceding $\mathcal{T}'$, is computed from three image features of detections from the last frames in $\mathcal{T}$ and the first frames in $\mathcal{T}'$[2]. The features we used are histograms of color, gradient and local binary patterns as in Ojala et al. (2000).

Color affinity $\psi_c(\mathcal{T}, \mathcal{T}') = 0$, if there is no common camera view between frame detections of the last frame of $\mathcal{T}$ and those of the first frame of $\mathcal{T}'$. Otherwise, for each common camera view of the two sets of frames, we extract color histograms of the corresponding two detections and evaluate their Bhattacharyya distance. $\psi_c(\mathcal{T}, \mathcal{T}')$ is computed as the average of such Bhattacharyya distances over all common views. The similarity based on histograms of gradient $\psi_s(\mathcal{T}, \mathcal{T}')$ and local binary patterns $\psi_b(\mathcal{T}, \mathcal{T}')$ are computed similarly. We denote the correspondence between a hypergraph node and a coupling as $\nu \sim \mathcal{T}$. The appearance affinity of node set $\boldsymbol{\nu} = (\nu_1, \cdots, \nu_k)$ with $\nu_i \sim \mathcal{T}_i$ in ascending order of time is computed as

$$\Psi_{\mathrm{app}}(\boldsymbol{\nu}) = \sum_{i,j} e^{\lambda_1 \psi_c(\mathcal{T}_i, \mathcal{T}_j) + \lambda_2 \psi_s(\mathcal{T}_i, \mathcal{T}_j) + \lambda_3 \psi_b(\mathcal{T}_i, \mathcal{T}_j)}, \qquad (6)$$

where λ_1, λ_2 and λ_3 are parameters controlling the sensitivity of the affinity score with regards to each type of appearance features, and $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

Motion continuation affinity The motion continuation affinity between a pair of non-overlapping couplings $\mathcal{T}$ and $\mathcal{T}'$, with $\mathcal{T}$ preceding $\mathcal{T}'$, is based on the forward-backward predictions between the last frame detections of $\mathcal{T}$ and the first frame detections of $\mathcal{T}'$.

We first estimate the "ending" velocity of $\mathcal{T}$ by dividing the difference of 3D positions [computed with Eq. (4)] of its last two frame detections with their corresponding time lapse. The predicted position for the start of $\mathcal{T}'$ is obtained by projecting the 3D position of the last frame detection of $\mathcal{T}$ with the estimated ending velocity, multiplied by the time lapse between the last frame detection of $\mathcal{T}$ and that of the first frame detection of $\mathcal{T}'$, Fig. 3a. We then compute the ℓ_2 distance between the actual 3D position of the first frame of $\mathcal{T}'$ and its *forward* prediction from $\mathcal{T}$, as $d_{fp}(\mathcal{T}, \mathcal{T}')$.

Similarly, the *backward* prediction of the 3D position of the last frame detection of $\mathcal{T}$ is obtained with the 3D position of the first frame detection of $\mathcal{T}'$ and the estimated beginning velocity from its first two frame detections, Fig. 3a. We compute the ℓ_2 distance between the 3D position of the last frame of $\mathcal{T}$ and its backward prediction from $\mathcal{T}'$, as $d_{bp}(\mathcal{T}, \mathcal{T}')$. Then the motion continuation affinity of node set $\boldsymbol{\nu} = (\nu_1, \cdots, \nu_k)$ with $\nu_i \sim \mathcal{T}_i$ in ascending order of time is computed as

$$\Psi_{\mathrm{mot}}(\boldsymbol{\nu}) = \sum_{i=1}^{k-1} e^{-\lambda_4 \left(d_{fp}(\mathcal{T}_i, \mathcal{T}_{i+1}) + d_{bp}(\mathcal{T}_i, \mathcal{T}_{i+1})\right)}, \qquad (7)$$

where λ_4 is the parameter controlling the sensitivity of the affinity score to the prediction errors.

Trajectory smoothness affinity A common assumption for visual tracking task is that tracked targets should move continuously and smoothly for most of the time. The trajectory smoothness affinity evaluates the spatial-temporal coherence of a long trajectory formed from a set of non-overlapping couplings $\mathcal{T}_1, \cdots, \mathcal{T}_k$. Specifically, we first compute the 3D positions of these couplings with Eq. (4). We then fit a piecewise second order smooth parametric trajectory with cubic spline interpolation to a subset of these 3D positions, which are sampled with equal time interval, Fig. 3b. The ℓ_2 distance, $d_{\mathrm{int}}(\mathcal{T}_1, \cdots, \mathcal{T}_k)$, of the remaining 3D positions with their predictions based on the interpolated smooth curve is computed, which evaluates the smoothness of the fitted trajectory (where small values indicate coherent fit). The trajectory smoothness affinity score of node set $\boldsymbol{\nu} = (\nu_1, \cdots, \nu_k)$ with $\nu_i \sim \mathcal{T}_i$ in ascending order of time is computed from $d_{\mathrm{int}}(\mathcal{T}_1, \cdots, \mathcal{T}_k)$ as

$$\Psi_{\mathrm{smo}}(\boldsymbol{\nu}) = e^{-\lambda_5 d_{\mathrm{int}}(\mathcal{T}_1, \cdots, \mathcal{T}_k)}, \qquad (8)$$

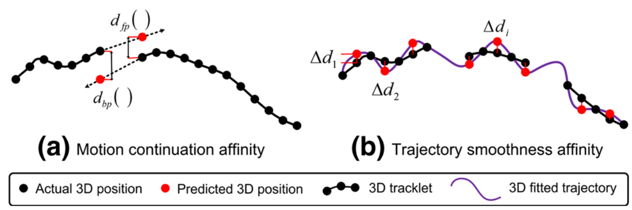

Fig. 3 **a** Motion continuation affinity calculation of a pair of couplings. **b** Trajectory smoothness affinity calculation of a set of couplings. See text for more details

[2] The last frame index of $\mathcal{T}$ and the first frame index of $\mathcal{T}'$ may correspond to multiple detections from different camera views.

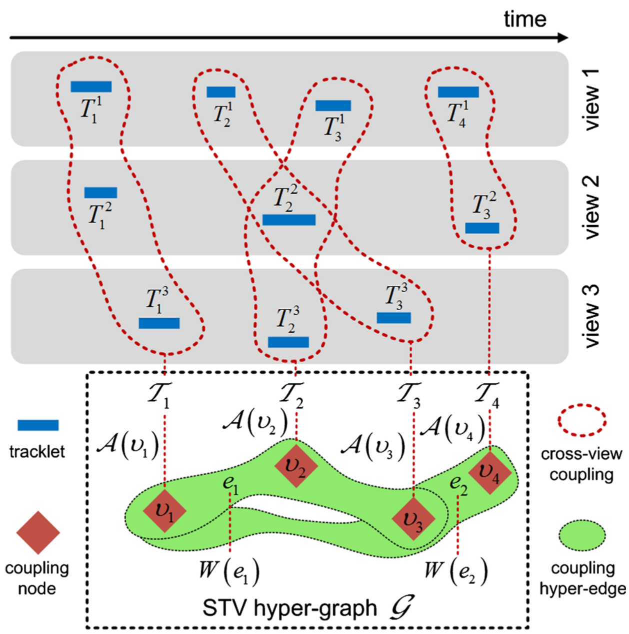

Fig. 4 An example of the constructed STV hyper-graph encoding both the reconstruction and linking of detected 2D tracklets in different camera views

where parameter λ_5 controls the sensitivity of the affinity score to the deviation of smooth trajectories.

3.3 The Space-Time-View Hyper-graph

The STV hyper-graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ encodes both the reconstruction and linking of detected 2D tracklets in different camera views, and is the central data structure of our multi-camera multi-target tracking method. See Fig. 4 for an illustrated example.

A **node** v in the STV hyper-graph corresponds to a coupling $\mathcal{T}$ (as described in Sect. 3.1) and is associated with a weight reflecting its reliability:

$$\mathcal{A}(v) = e^{-\lambda_6 \max_{t \in \mathbf{t}_{\mathcal{T}}} \{\epsilon_t^{\mathcal{T}} - \lambda_7 |\mathcal{D}_t^{\mathcal{T}}|\}}, \tag{9}$$

where $\epsilon_t^{\mathcal{T}}$ [from Eq. (5)] is the scattering of coupling $\mathcal{T}$ at frame t, and $|\mathcal{D}_t^{\mathcal{T}}|$ [see Eq. (3)] is the number of 2D tracklets associated with the coupling at frame t. λ_6 controls the sensitivity of the weight to the reliability score, and λ_7 represents the trade-off between lower scattering and larger number of associated 2D views. Higher node weights suggest increasing likelihood of the composing 2D tracklets corresponding to a single 3D tracklet. To avoid the case where couplings corresponding to a single 2D view dominate the weight (where the scattering is always zero), we also penalize couplings with smaller number of associated 2D views through parameter λ_7.

A *hyper-edge* in the STV hyper-graph connects multiple nodes with a weight, whose corresponding couplings potentially form a longer trajectory as shown in Fig. 2c. We only consider hyper-edge of degree k, i.e., each hyper-edge in STV hyper-graph is associated with k nodes[3].

We enforce two constraints that are important to reduce the number of hyper-edges. First, for all nodes connected by one hyper-edge, their couplings should not overlap in time. Second, we evaluate the distance between the last and first detections of either pair of couplings in a hyper-edge. If the distance is significantly larger than the maximum possible velocity (e.g., < 5 m/s for a pedestrian), then the two nodes should not be grouped together by a hyper-edge. The weight of a hyper-edge in STV hyper-graph, $e = (v_1, \cdots, v_k)$ is computed using the appearance, motion continuity and trajectory smoothness affinities defined in Sect. 3.2,

$$W(e) = \lambda_8 \Psi_{\text{app}}(e) + \lambda_9 \Psi_{\text{mot}}(e) + \lambda_{10} \Psi_{\text{smo}}(e), \tag{10}$$

where parameters λ_8, λ_9, and λ_{10} balance the three types of affinity scores, and $\lambda_8 + \lambda_9 + \lambda_{10} = 1$.

3.4 Dense Sub-hypergraph Search

We formulate the problem of recovering longer trajectories of the targets as searching for "dense" sub-hypergraphs on the STV hyper-graph. Here a dense sub-hypergraph corresponds to a group of reliable nodes (couplings) that are inter-connected with a set of hyper-edges with high weights. We extend a local search algorithm for dense subgraphs in graphs (Liu et al. 2012) to search the dense sub-hypergraphs on the STV hyper-graph. The basic idea is to find a dense neighborhood for each node in STV hyper-graph and then remove the conflicts between such neighborhoods to obtain the longer target trajectories. To accommodate the large number of nodes in STV hyper-graph, we further employ a sampling based algorithm to accelerate such search.

3.4.1 Problem Formulation

For a node v, we denote its *neighborhood* as $\mathcal{N}(v)$, which is the set of nodes containing direct neighbors of v (i.e., connected with one hyper-edge to v) on the STV hyper-graph. We then aim to find a subset of $\mathcal{N}(v)$ with β nodes such that they jointly form an β-subhypergraph[4] that has the maximum weights combining both the hyper-edges and nodes. To this end, we introduce an indicator variable $z_{v'}$ for each $v' \in \mathcal{N}(v)$, which is $1/\beta$ if v' is in a dense sub-hypergraph

[3] Note that this is different from the degree of the nodes, which specifies how many hyper-edges can associate with one node.

[4] The β-subhypergraph indicates the sub-hypergraph of STV hyper-graph, which includes β nodes.

and 0 otherwise. The search of dense sub-hypergraph can then be formulated as the following discrete optimization problem

$$\begin{aligned} \mathbf{z}_\nu^* &= \operatorname*{argmax}_{\nu' \in \mathcal{N}(\nu)} \sum_{e \in \mathcal{U}_\nu} W(e) \prod_{\nu' \in e} z_{\nu'} + \sum_{\nu' \in \mathcal{N}(\nu)} \mathcal{A}(\nu') z_{\nu'}^k \\ \text{s.t.} \quad & \sum_{\nu' \in \mathcal{N}(\nu)} z_{\nu'} = 1, \\ & \forall \nu' \in \mathcal{N}(\nu), \quad z_{\nu'} \in \{0, 1/\beta\}, \end{aligned} \tag{11}$$

in which $\mathbf{z}_\nu^*$ is the optimal indicator variable vector, consisted of nonzero $z_{\nu'}$, corresponding to the searched dense sub-hypergraph, $\mathcal{U}_\nu$ is the hyper-edge set corresponding to the node set $\nu \cup \mathcal{N}(\nu)$, $\mathcal{A}(\nu)$ and $W(e)$ are the weights of nodes in Eq. (9) and hyper-edges in Eq. (10), respectively. The first term in the objective function encourages the inclusion of hyper-edges in ν's sub-hypergraph with larger weights, while the second term penalizes the inclusion of nodes correspond to less reliable couplings indicated by a lower $\mathcal{A}(\nu)$. The first constraint in Eq. (11) requires that the sub-hypergraph should include β nodes, and the second constraint enforces that the label can only take two values.

The optimization problem in Eq. (11) is different from the one formulated for single-camera multi-target tracking as in Wen et al. (2014). The hyper-graph in Wen et al. (2014) does not have node weights reflecting uncertainty in forming 3D tracklets. Furthermore, the optimization algorithm in Wen et al. (2014) will run inefficiently if it is directly applied to solve Eq. (11), due to the large size of our optimization problem. For instance, for three camera views with each containing ten tracklets, a 3-degree STV hyper-graph has $10^3 = 1000$ nodes and $\binom{10^3}{3} \approx 1.67 \times 10^8$ hyper-edges.

3.4.2 Constructing STV Graph

Motivated by Liu and Yan (2012), we propose an approximate approach to search dense sub-hypergraph for efficient optimization of Eq. (11), the basic idea is to perform a more efficient search on a graph approximation to the hyper-graph. This method thus strikes a balance between the expressiveness of the hyper-graph representation and computational efficiency of the graph approximation.

To be specific, we construct a STV graph to approximate the STV hyper-graph. Then, we search for dense *subgraphs* on STV graph, from which dense sub-hypergraphs of STV hyper-graph can be recovered. Unlike in previous works (Leal-Taixé et al. 2012; Hofmann et al. 2013), we construct STV graph from STV hyper-graph to explicitly capture higher-order temporal correlations while maintaining efficacy. We essentially combine the advantages of using a hyper-graph to capture higher-order correlations and the computational efficiency of a graph. We analyze the effectiveness of our approach in Sect. 4.

Algorithm 1 Constructing the STV Graph.

Input: The node set $\mathcal{V} = \{\nu_1, \cdots, \nu_n\}$ in the STV hyper-graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Set the CS set to be empty set.
for $i = 1$ to n **do**
  $j = 0$.
  while $j < \xi$ **do**
    Randomly sample $k-2$ nodes to obtain node set Λ_i from $\mathcal{N}(\nu_i) - \{\nu_i\}$.
    $\boldsymbol{\pi}_i = \{\nu_i\} \cup \Lambda_i$.
    Add $\boldsymbol{\pi}_i$ to the CS set.
    $j = j + 1$.
  end while
end for
for $i = 1$ to $n \cdot \xi$ **do**
  for $j = 1$ to n **do**
    if The j-th node ν_j belongs to $\boldsymbol{\pi}_i$ **then**
      The confidence score of the j-th node ν_j to the i-th CS $\boldsymbol{\pi}_i$, $\mathcal{S}_j(\boldsymbol{\pi}_i) = \mu$.
    else
      The confidence score of the j-th node ν_j to the i-th CS $\boldsymbol{\pi}_i$, $\mathcal{S}_j(\boldsymbol{\pi}_i)$ is calculated as Eq. (12).
    end if
  end for
end for
Set the node set of the STV graph $\mathcal{V}^* = \mathcal{V}$.
Set all elements in the weight matrix W^* corresponding to all candidate edges in STV graph $\mathcal{G}^*$ to zeros.
for $i = 1$ to $n \cdot \xi$ **do**
  Obtain the reliable node set $\Omega_i = \{\nu_j | \mathcal{S}_j(\boldsymbol{\pi}_i) \geq \mu, j = 1, \cdots, n\}$.
  Calculate $\rho_i = \binom{|\Omega_i|-3}{k-3}$.
  for Each node pair $\{\nu, \nu'\}$, $\nu, \nu', \nu_j \in \Omega_i$, $\nu \neq \nu'$, $\nu \neq \nu_j$, and $\nu' \neq \nu_j$ **do**
    $W^*(\nu, \nu') = W^*(\nu, \nu') + \rho_i \cdot \mathcal{S}_j(\boldsymbol{\pi}_i)$.
    $W^*(\nu', \nu) = W^*(\nu', \nu) + \rho_i \cdot \mathcal{S}_j(\boldsymbol{\pi}_i)$.
  end for
end for
Set the edge set in STV graph $\mathcal{G}^*$ to be empty set, *i.e.*, $\mathcal{E}^* = \emptyset$.
for $i = 1$ to $n - 1$ **do**
  for $j = i + 1$ to n **do**
    The edge $\mathcal{E}^*(\nu, \nu')$ is added with the weight $W^*(\nu, \nu')$, iff $W^*(\nu, \nu') > 0$.
  end for
end for

Output: The STV Graph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$.

We construct STV graph by keeping all the nodes in STV hyper-graph and sampling the hyper-edges through a set of Connection Samples (CSs), $\{\boldsymbol{\pi}_1, \cdots, \boldsymbol{\pi}_i, \cdots\}$. Each CS $\boldsymbol{\pi}_i$ is a set of $k-1$ nodes from STV hyper-graph, constructed by *traversing* each node ν for ξ times, and for each time randomly select other $k-2$ nodes from $\mathcal{N}(\nu) - \{\nu\}$. Obviously, the number of sampled CS pairs is much smaller than the total number of hyper-edges. Thus, it is more efficient to search the dense subgraph using sampling strategy than traverse all the hyper-edges in Wen et al. (2014). In total, we can obtain $n \cdot \xi$ sampled CSs, where n is the number of nodes in

STV hyper-graph. For each node in the STV hyper-graph, we form a new hypothetical hyper-edge with each CS by pooling all nodes together. The scores of all nodes to a CS $\boldsymbol{\pi}_i$ are calculated, $\boldsymbol{\mathcal{S}}(\boldsymbol{\pi}_i) = \{\mathcal{S}_1(\boldsymbol{\pi}_i), \cdots, \mathcal{S}_n(\boldsymbol{\pi}_i)\}$. If the j-th node belongs to $\boldsymbol{\pi}_i$, we set $\mathcal{S}_j(\boldsymbol{\pi}_i)$ to a predefined confidence score threshold μ. Otherwise, we use

$$\mathcal{S}_j(\boldsymbol{\pi}_i) = \lambda_8 \cdot \Psi_{\text{app}}(\boldsymbol{\pi}_i \cup \{\nu_j\}) + \lambda_9 \cdot \Psi_{\text{mot}}(\boldsymbol{\pi}_i \cup \{\nu_j\}) + \lambda_{10} \cdot \Psi_{\text{smo}}(\boldsymbol{\pi}_i \cup \{\nu_j\}). \tag{12}$$

For each hyper-edge e, we define its approximate weight after sampling $\boldsymbol{\pi}_i$ as:

$$W^{[i]}(e) = \max\{W^{[i-1]}(e), \min_{j=\{1,\cdots,k\}} \mathcal{S}_{\nu_j}(\boldsymbol{\pi}_i)\}. \tag{13}$$

The approximate weight of each hyper-edge satisfies $0 \leq W^{[1]}(e) \leq \cdots W^{[i]}(e) \leq W(e)$, where i is the number of sampled CSs. As i increases, $W^{[i]}(e)$ approximates $W(e)$ gradually. We do not need to store $W^{[i]}(e)$ of each hyper-edge in the sampling procedure. Instead, the scores $\{\boldsymbol{\mathcal{S}}(\boldsymbol{\pi}_1), \cdots, \boldsymbol{\mathcal{S}}(\boldsymbol{\pi}_i), \cdots\}$ are stored. They contain hyper-edge weight of the included node and CS pairs, which represent crucial information of our method.

We construct the STV graph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$ using the scores from the nodes and hyper-edges of STV hyper-graph. Specifically, $\mathcal{V}^*$ is the same weighted node set as the corresponding STV hyper-graph $\mathcal{G}$, and $\mathcal{E}^* = \mathcal{V}^* \times \mathcal{V}^*$ is edge set describing the supports between the nodes. Intuitively, if two nodes belong to the same dense sub-hypergraph on $\mathcal{G}$, they are expected to simultaneously appear in several hyper-edges with large weights. Specifically, for node ν and ν', we set the weight of the edge connecting them in STV graph to reflect the number of hyper-edges including both ν and ν' with large weights in the STV hyper-graph, based on the scores $\{\boldsymbol{\mathcal{S}}(\boldsymbol{\pi}_1), \cdots, \boldsymbol{\mathcal{S}}(\boldsymbol{\pi}_i), \cdots\}$. To exclude the information contained in the unreliable hyper-edges, We define $\Omega_i = \{\nu_j | \mathcal{S}_j(\boldsymbol{\pi}_i) \geq \mu, j = 1, \cdots, n\}$ to be the reliable node set with the score larger than μ, where n is the number of nodes in the STV hyper-graph. Then the weight of the edge connecting them in STV graph is calculated as

$$W^*(\nu, \nu') = \sum_{i=1}^{n \cdot \xi} \sum_{\nu, \nu', \nu_j \in \Omega_i} \rho_i \cdot \mathcal{S}_j(\boldsymbol{\pi}_i), \tag{14}$$

where $\rho_i = \binom{|\Omega_i|-3}{k-3}$ is the total number of hyper-edges containing nodes ν and ν' in the original STV hyper-graph[5]. Algorithm 1 shows the main steps to construct the STV graph.

[5] The calculation of the number of hyper-edges, including nodes ν, ν' and ν_j is a combinational problem, that is to choose $k-3$ nodes from the reliable node set $\Omega_i - \{\nu, \nu', \nu_j\}$. Specifically, we set $\rho_i = 0$ for $|\Omega_i| < 3$, since there does not exist enough nodes to construct a hyper-edge in that case.

3.4.3 Dense Subgraph Search on STV Graph

After constructing the STV graph $\mathcal{G}^*$, we search the dense subgraphs on it to complete the tracking task. Similar to Eq. (11), the problem is formulated as

$$\begin{aligned} \mathbf{z}_\nu^* = \operatorname{argmax} & \sum_{\nu' \in \widetilde{\mathcal{N}}(\nu)} \sum_{e^* \in \mathcal{U}_\nu^*} W^*(e^*) z_\nu z_{\nu'} + \sum_{\nu' \in \widetilde{\mathcal{N}}(\nu)} \mathcal{A}(\nu') z_{\nu'}^2 \\ \text{s.t.} & \sum_{\nu' \in \widetilde{\mathcal{N}}(\nu)} z_{\nu'} = 1, \\ & \forall \nu' \in \widetilde{\mathcal{N}}(\nu), \quad z_{\nu'} \in \{0, 1/\beta\}, \end{aligned} \tag{15}$$

where $\mathbf{z}_\nu^*$ is the optimal indicator variable vector, consisted of nonzero $z_{\nu'}$, corresponding to the searched dense sub-hypergraph, $e^* = (\nu, \nu')$ is the edge in the STV graph, $\widetilde{\mathcal{N}}(\nu)$ is the neighborhood of node ν, $\mathcal{U}_\nu^*$ is the edge set corresponding to the node set $\nu \cup \widetilde{\mathcal{N}}(\nu)$. We denote the node set of the searched dense subgraph corresponds to node ν as $\hat{\gamma}_\nu$, and set $\hat{\gamma}_\nu = \emptyset$ at first. Then, we can obtain the searched dense subgraph corresponding to the node indicator variable $\mathbf{z}_\nu^*$, i.e., if $z_{\nu'} > 0$, we add node ν' to $\hat{\gamma}_\nu$. Meanwhile, we can also calculate the corresponding confidence score $\hat{\varpi}_\nu$ of each search dense subgraph, which is the function value of the objective in Eq. (15) corresponding to the optimal solution $\mathbf{z}_\nu^*$.

Optimization problem in Eq. (15) is an NP-hard discrete optimization problem (Liu et al. 2012). To reduce its complexity, we relax the discrete constraint $z_{\nu'} \in \{0, 1/\beta\}$ to its continuous counterpart $z_{\nu'} \in [0, 1/\beta]$, and thus convert Eq. (15) into a continuous optimization problem. Meanwhile, to avoid degeneracy, we require the minimal size of the subgraph to be a fixed number $\beta^* \leq \min_{\nu \in \mathcal{V}} |\widetilde{\mathcal{N}}(\nu)| \leq \beta$, where $\widetilde{\mathcal{N}}(\nu)$ is a set containing the direct neighbors of ν on the graph $\mathcal{G}^*$. Thus, the constraint is further converted as $z_{\nu'} \in [0, 1/\beta^*]$. An efficient method based on pairwise coordinate update given in Liu et al. (2012) is used to solve Eq. (15) for each node in $\mathcal{G}^*$ to obtain the dense subgraphs $\widehat{\Gamma} = \{\hat{\gamma}_i\}_{i=1}^n$ and the corresponding confidence scores $\hat{\boldsymbol{\varpi}} = \{\hat{\varpi}_i\}_{i=1}^n$, which will be described as follows.

3.4.4 Optimization Using Pairwise Updates

We adopt the pairwise update algorithm to optimize Eq. (15) as in Liu et al. (2012). The formulation in Eq. (15) is a constrained optimization problem, we introduce Lagrangian multipliers $a, b_1, \cdots, b_u$, and $c_1, \cdots, c_u$ for each variable z_i, $i \neq \nu$ and $i \in \widetilde{\mathcal{N}}(\nu)$, i.e., where $a \geq 0$ and $b_i \geq 0$ and $c_i \geq 0$ for all $i = 1, \cdots, u$, u is the number of nodes in the neighborhood $\widetilde{\mathcal{N}}(\nu)$. The Lagrangian of the original problem Eq. (15) is

$$\mathcal{M}(\mathbf{z}, a, b, c) = f(\mathbf{z}) - a \cdot \left(\sum_{i=1}^{u} z_i - 1\right) + \sum_{i, i \neq \nu} b_i \cdot z_i$$

$$+\sum_{i,i\neq v} c_i \cdot \left(\frac{1}{\beta} - z_i\right), \tag{16}$$

where $f(\mathbf{z}) = \sum_{e^*_{ij}\in\mathcal{U}^*} W^*_{ij}\cdot z_i z_j + \sum_{j\in\widetilde{\mathcal{N}}(i)} \mathcal{A}(j) z_j^2$, e^*_{ij} is the edge connecting node i and j in $\mathcal{G}^*$, $\mathbf{z} = (z_1, \cdots, z_u)$ is the indicator vector ($z_i = \frac{1}{\beta}$ means the node i is involved in the dense subgraph and $z_i = 0$ means the node i is excluded from the dense subgraph), and β is the number of nodes included in the searched dense subgraph. Similar to Liu et al. (2012), we introduce a *reward* score $r_i(\mathbf{z}) = \sum_l W^*_{il}\cdot z_l + \frac{1}{2}\mathcal{A}(i)$ at node i reflecting the total weights of node i to other nodes described by the indicator $\mathbf{z}$. Then, we have $\frac{\partial f}{\partial z_i}(\mathbf{z}^*) = 2\cdot r_i(\mathbf{z}^*)$, i.e., the *reward* score is proportional to the gradient of $f(\mathbf{z})$ at $\mathbf{z}$.

Any local maxima $\mathbf{z}^*$ must satisfy the Karush-Kuhn-Tucker (KKT) conditions (Kuhn and Tucker 1951),

$$\begin{cases} 2\cdot r_i(\mathbf{z}^*) - a + b_i - c_i = 0, i \neq v; \\ \sum\limits_{i,i\neq v} z_i^* \cdot b_i = 0; \\ \sum\limits_{i,i\neq v} c_i \cdot (1/\beta - z_i^*) = 0. \end{cases} \tag{17}$$

Since z_i^*, b_i, and c_i are all nonnegative, and $\sum_{i,i\neq v} z_i^*\cdot b_i = 0$, we have two rewritten constraints: (1) $\forall i \neq v$, if $z_i^* > 0$, then $b_i = 0$; (2) $\forall i \neq v$, if $z_i^* < 1/\beta$, then $c_i = 0$. Thus, for nodes $i \neq v$, the KKT conditions can be further reformulated as:

$$r_i(\mathbf{z}^*) = \begin{cases} \leq a/2, & z_i^* = 0; \\ = a/2, & 0 < z_i^* < 1/\beta; \\ \geq a/2, & z_i^* = 1/\beta. \end{cases} \tag{18}$$

Similar to Liu et al. (2012), the node set in $\mathcal{G}^*$ can be divided into three disjoint subsets, $\Xi_1(\mathbf{z}) = \{i|z_i = 0\}$, $\Xi_2(\mathbf{z}) = \{i|z_i \in (0, 1/\beta)\}$ and $\Xi_3 = \{i|z_i = 1/\beta\}$.

Using Theorem 1 in Liu et al. (2012), we increase a component z_i and decrease z_j to increase $f(\mathbf{z})$, as

$$\hat{\mathbf{z}} = \begin{cases} z_l, & l \neq i, l \neq j; \\ z_l + \alpha, & l = i; \\ z_l - \alpha, & l = j, \end{cases} \tag{19}$$

and define $r_{ij} = W^*_{ii} + W^*_{jj} - 2W^*_{ij}$. Then, we have

$$\begin{aligned} \Delta f(\mathbf{z}) = & f(\hat{\mathbf{z}}) - f(\mathbf{z}) = (W^*_{ii} + W^*_{jj} - 2W^*_{ij})\alpha^2 \\ & + 2\Big(\sum_l W^*_{il}\cdot z_l - \sum_l W^*_{jl}\cdot z_l \\ & + \frac{1}{2}\big(\mathcal{A}(i) - \mathcal{A}(j)\big)\Big)\alpha \\ = & r_{ij}\cdot\alpha^2 + 2\Big(r_i(\mathbf{z}) - r_j(\mathbf{z})\Big)\alpha, \end{aligned} \tag{20}$$

where $\hat{\mathbf{z}} = (\hat{z}_1, \cdots, \hat{z}_u)$, u is the number of nodes in the neighborhood $\widetilde{\mathcal{N}}(v)$. Note that we can assume $r_i(\mathbf{z}) \geq r_j(\mathbf{z})$, when $r_i(\mathbf{z}) < r_j(\mathbf{z})$, we can exchange i and j to maximize $\Delta f(\mathbf{z})$. α can be calculated based on Eq. (20) and the constraints over α and $\mathbf{z}$ that is $\alpha = \min(z_j, 1/\beta - z_i)$, if $r_i(\mathbf{z}) > r_j(\mathbf{z})$ and $r_{ij} \geq 0$; $\alpha = \min(z_j, 1/\beta - z_i, \frac{r_j(\mathbf{z}) - r_i(\mathbf{z})}{r_{ij}})$, if $r_i(\mathbf{z}) > r_j(\mathbf{z})$ and $r_{ij} < 0$; and $\alpha = \min(z_j, 1/\beta - z_i)$, if $r_i(\mathbf{z}) = r_j(\mathbf{z})$ and $r_{ij} > 0$. After that, we can compute the local maximizer $\mathbf{z}^*$ of Eq. (15) by iteratively using the update strategy Eq. (19) and calculating α based on the discussions above from any starting points. We adopt the kNN(o) strategy given in Liu et al. (2012) to complete the initialization. A complete analysis of this algorithm can be found in Liu et al. (2012).

Algorithm 2 Conflict removal of the searched dense subgraphs.

Input: The node sets of the searched dense subgraphs $\widehat{\Gamma} = \{\hat{\gamma}_i\}_{i=1}^n$ and the corresponding confidence scores $\hat{\boldsymbol{\varpi}} = \{\hat{\varpi}_i\}_{i=1}^n$.

Sort the dense subgraphs in $\widehat{\Gamma}$ in descending order according the confidence scores $\hat{\boldsymbol{\varpi}}$ to get $\widetilde{\Gamma} = \{\tilde{\gamma}_i\}_{i=1}^n$.
Initial the dense subgraphs without conflicts $\Gamma^* = \emptyset$.
for $i = 1$ to n **do**
  if $\tilde{\gamma}_i \cap \gamma_j^* = \emptyset, \forall j, \gamma_j^* \in \Gamma^*$ **then**
    $\Gamma^* \leftarrow \Gamma^* \bigcup \{\tilde{\gamma}_i\}$.
  else
    $\gamma_j^* \leftarrow \gamma_j^* \bigcup \tilde{\gamma}_i$.
  end if
end for

Output: The dense subgraphs without conflicts Γ^*.

3.4.5 Conflict Removal and Formation of Long Trajectory

After identifying the node sets of the dense subgraphs $\widehat{\Gamma}$ and the corresponding confidence scores $\hat{\boldsymbol{\varpi}}$ of all nodes, we use some post-processing strategies to filter out the conflicts involved in $\widehat{\Gamma}$, *e.g.*, one node may appear in multiple clusters. We use a similar post-processing strategy as in Wen et al. (2014). We first produce an ordered cluster set $\widetilde{\Gamma} = \{\tilde{\gamma}_i\}_{i=1}^n$ according to the corresponding confidence score $\hat{\varpi}_i$ in descending order from the searched dense subgraphs $\widehat{\Gamma}$. Let Γ^* be the dense subgraphs without conflicts. We initialize $\Gamma^* = \emptyset$. For the i-th searched dense subgraph in $\widetilde{\Gamma}$, *i.e.*, $\tilde{\gamma}_i \in \widetilde{\Gamma}$, if $\tilde{\gamma}_i \cap \gamma_j^* = \emptyset, \forall j, \gamma_j^* \in \Gamma^*$, we add $\tilde{\gamma}_i$ directly to Γ^*, *i.e.*, $\Gamma^* \leftarrow \Gamma^* \bigcup \{\tilde{\gamma}_i\}$. Otherwise, a greedy approach is designed by directly adding $\tilde{\gamma}_i$ to the existing clusters γ_j^*, i.e., $\gamma_j^* \leftarrow \gamma_j^* \bigcup \tilde{\gamma}_i$. Algorithm 2 shows the main steps to remove the conflicts of searched dense subgraphs.

3.4.6 Handle Long Videos

For long videos (e.g., with more than 500 frames), constructing the STV hyper-graph on all couplings and performing search require large memory and computation. To improve running efficiency, we divide long videos into non-overlapping segments of fixed length. For each segment,

Algorithm 3 Main steps of the STV hyper-graph based multi-camera multi-target tracking.

Input: Video sequences captured from multi-camera views synchronously with the same frame rate, and the corresponding 2D detection results of each frame in each camera view.
1: Divide long videos into J non-overlapping segments of fixed length.
2: **while** $J > 1$ **do**
3: **for** $j = 1$ to J **do**
4: Generate the candidate couplings based on the detected tracklets in each camera view in the j-th segment (§ 3.1) to obtain the node set $\mathcal{V}$ of STV hyper-graph $\mathcal{G}$, *i.e.*, each candidate coupling corresponds a node in $\mathcal{G}$.
5: Calculate the weights of nodes based on Eq. (9) in $\mathcal{G}$.
6: Construct STV graph according to Algorithm 1 (§ 3.4.2).
7: Search dense subgraphs on STV graph (§ 3.4.3).
8: Remove conflicts in searched dense subgraphs and generate long tracklets according to Algorithm 2 (§ 3.4.5).
9: **end for**
10: Merge the neighboring segments and update segment number J.
11: **end while**
Output: Target long trajectories in the video sequences.

the STV hyper-graph is constructed and the dense sub-hypergraph search is performed. We then treat the recovered 3D trajectories as a new coupling, and construct a new STV hyper-graph as Sect. 3.3, from which another round of dense sub-hypergraph search and trajectory linking are performed. This procedure continues until the whole video sequence is merged into a single hyper-graph, where the sub-hypergraph search yields final trajectories. In addition, to avoid exclusion of correct trajectories generated in previous layers, we append the nodes that are not included in the searched dense subgraphs while satisfying the length requirement after the conflict removal step in each layer. That is, we first consider the set of nodes that are excluded from the detected dense subgraph set $\Theta = \{v_1, \cdots, v_n\} - \bigcup_{i=1}^{\tau} \gamma_i^*$, where $\Gamma^* = \{\gamma_i^*\}_{i=1}^{\tau}$ is the set of detected dense subgraphs. For $\theta_i \in \Theta$, if $\mathcal{L}(\theta_i) \geq \ell$, we add θ_i to Γ^*, *i.e.*, $\Gamma^* \leftarrow \Gamma^* \bigcup \{\theta_i\}$, where $\mathcal{L}(\theta_i)$ indicates the trajectory length corresponding to node θ_i, and ℓ is the preset minimal length of the target trajectory. In this way, the multi-camera multi-target tracking task can be completed efficiently. Algorithm 3 shows the main steps of our approach to complete the multi-camera multi-target tracking task.

4 Experiments

4.1 Dataset

Multi-camera multi-object tracking We evaluate the performance of our approach and compare with several state-of-the-art methods on the PETS 2009 multi-camera multi-object tracking dataset Ferryman and Shahrokni (2009), which includes three video sequences obtained from multiple synchronized cameras:

– **S2.L1** low target density, 19 moving pedestrians in 795 frames;
– **S2.L2** medium target density, 43 pedestrians spreading in 436 frames;
– **S2.L3** high target density, 44 pedestrians moving together in 240 frames.

These videos represent practical challenges in multi-target tracking, including frequent target occlusions, close targets with similar appearance, and low frame rate (∼7 frame-per-second). In our experiments, we compare tracking results using multiple camera views for each of the three PETS 2009 sequences. To make fair comparison, we use frame detections obtained with the Deformable Part Model (DPM) algorithm (Felzenszwalb et al. 2008) as the input for all evaluated methods. In the performance evaluation, We use ground truth annotation provided in Milan et al. (2011).

Single-camera multi-object tracking To demonstrate the generality of the proposed approach, we also evaluate our approach on the PETS 2009 single-camera multi-object tracking dataset (Ferryman and Shahrokni 2009) and MOTChallenge 2015 single-camera 3D benchmark (Leal-Taixé et al. 2015). For the PETS 2009 dataset, following the previous single-camera multi-object tracking methods (Andriyenko and Schindler 2011; Andriyenko et al. 2012; Wen et al. 2014), we use videos captured by camera #1 of sequences S2L1, S2.L2 and S2.L3 to complete the tracking task. The MOTChallenge 3D benchmark consists of four sequences captured using a static camera, i.e., AVG-TownCentre, PETS 2009-S2.L1, PETS 2009-S2.L2 and TUD-Stadtmitte, with the calibration files used to compute a 2D homography between the image plane and the ground plane. The PETS 2009-S2.L1 and TUD-Stadtmitte sequences are used for training, while the remaining two sequences, i.e., PETS 2009-S2.L2 and AVG-TownCentre, are used for testing. The DPM algorithm (Felzenszwalb et al. 2008) is used to generate the input detections for all trackers in the PETS 2009 dataset evaluation. While for the MOTChallenge 3D benchmark, the publicly provided input detection results (Leal-Taixé et al. 2015) are adopted to complete the tracking task.

4.2 Evaluation Metrics

To quantitatively evaluate the performance of both multi-camera and single-camera multi-target tracking scenarios, we adopt two CLEAR MOT metrics for multi-target tracking (Stiefelhagen et al. 2006): (i) Multi-Object Tracking Accuracy (MOTA), a consolidated score of false/miss detection rates of ground truth and identity switches of tracked trajectories; and (ii) Multi-Object Tracking Precision (MOTP), the average distance between the tracking results and the ground truth normalized to the hit/miss threshold. The MOTA

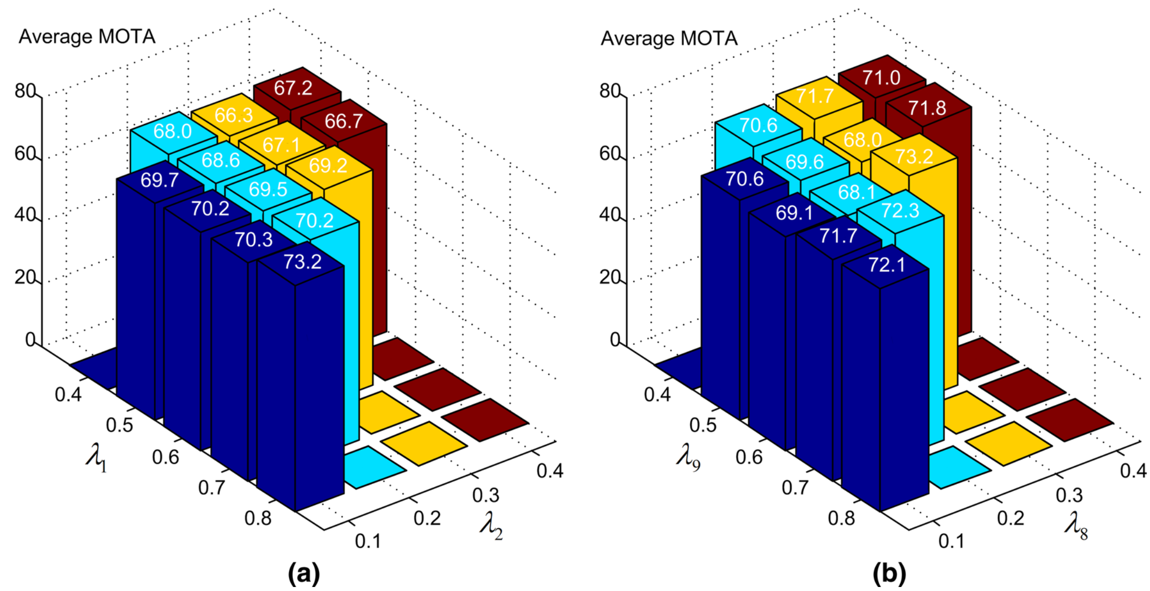

Fig. 5 **a** Influence of balance parameters between color histogram, gradient histogram, and local binary pattern features, on tracking performance. **b** Influence of balance parameters between appearance, motion, and trajectory smoothness affinities on tracking performance

score is perhaps the widely used figure to evaluate the performance of the tracker, since it combines three errors [i.e., False Negatives (FN), False Positives (FP), and Identity Switches (IDS)] into a single number (Leal-Taixé et al. 2015). To measure the performance of the tracker, similar to Andriyenko et al. (2012), we use the greedy strategy to match the locations of tracked targets and the ground truth within a hit/miss distance threshold. To describe the performance of the tracker completely, we plot the MOTA score at the hit/miss distance thresholds varied from 0 to 2 m. We use the MOTA score at the hit/miss distance threshold 1 m as the representative score to rank each tracking algorithm in the MOTA versus distance curves.

In addition, we also report the the number of the ground truth trajectories (GT), the ratio of ground truth trajectories that are tracked for more than 80 % of total length (MT), the ratio of ground truth trajectories that are tracked for less than 20 % of total length (ML), the number of times that a ground truth trajectory is detected with several separate trajectories (FM), and the number of times that a tracked trajectory changes its matched identity (IDS) at the hit/miss distance threshold 1 meter. To demonstrate the overall performance of the trackers, we follow the evaluation protocol in Leal-Taixé et al. (2015), which introduces the AvgRank score indicating the rank of each tracker averaged over all present evaluation measures with the perfect value 1. The lower value of AvgRank indicates the better performance.

In previous works, tracking performance on the PETS 2009 dataset has been evaluated in either the whole camera view (Leal-Taixé et al. 2012; Kuo and Nevatia 2011; Yang and Nevatia 2012a) or a predefined area in the intersection of all views (Milan et al. 2014; Wen et al. 2014). In this work, we use the whole camera view for both single-camera and multi-camera evaluations, as it is more relevant to practical tracking scenarios.

4.3 Parameters

We carry out several experiments on PETS 2009 dataset to study the influence of some important parameters in our algorithm. Firstly, we evaluate the influence of the trade-off parameters between color histogram, gradient histogram, and local binary pattern features in appearance affinity calculation in Eq. (6), i.e., λ_1, λ_2, and λ_3, on tracking performance. In our empirical study, we find that the color histogram is more discriminative than other two features. Thus, we keep $\lambda_1 \geq \lambda_2 > 0$ and $\lambda_1 \geq \lambda_3 > 0$. Since $\lambda_1 + \lambda_2 + \lambda_3 = 1$, we vary $\lambda_1 \in [0.4, 0.8]$ and $\lambda_2 \in [0.1, 0.4]$ with interval 0.1, and $\lambda_3 = 1 - \lambda_1 - \lambda_2$, while keep all other parameters fixed, and report the changes in the average MOTA score over three sequences, i.e., S2.L1, S2.L2, and S2.L3, in PETS 2009 dataset with two camera views in Fig. 5a. As presented in Fig. 5a, we find that our algorithm performs relative stable with these parameters, *i.e.*, the standard deviation of the average MOTA score is 1.91 %. Based on the maximal value in Fig. 5a, we set $\lambda_1 = 0.8$, $\lambda_2 = 0.1$, and $\lambda_3 = 0.1$.

Secondly, we evaluate the influence of the balance parameters between three types of affinity scores, i.e., appearance affinity, motion affinity, and trajectory smoothness affinity in Eq. (10), λ_8, λ_9, and λ_{10}, on tracking performance, which is shown in Fig. 5b. To handle the tracking task in crowded scenes, we take the motion affinity as a more important fac-

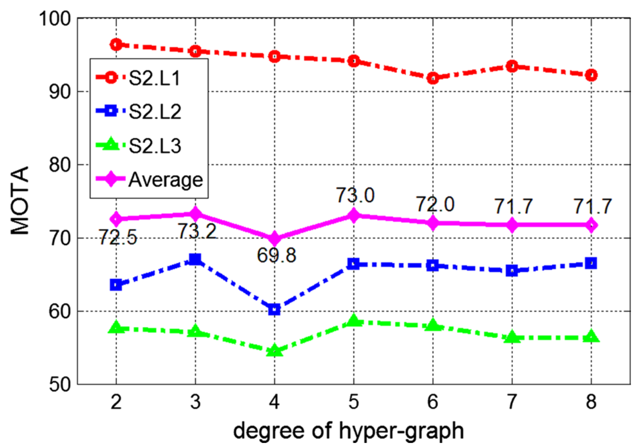

Fig. 6 Effect of the degree of hyper-graph k (number of nodes associated with each hyper-edge) on the tracking performance. The MOTA score is used to indicate the overall performance of the tracker. Note that the hyper-graph degenerates to a graph when $k = 2$

tor. Thus, we keep $\lambda_9 \geq \lambda_8 > 0$ and $\lambda_9 \geq \lambda_{10} > 0$. Since $\lambda_8+\lambda_9+\lambda_{10} = 1$, we vary $\lambda_8 \in [0.1, 0.4]$ and $\lambda_9 \in [0.4, 0.8]$ with interval 0.1, and $\lambda_{10} = 1 - \lambda_8 - \lambda_9$, while keep all other parameters fixed. The average MOTA scores over three sequences in PETS 2009 dataset with two camera views are presented in Fig. 5b. Our algorithm is relative stable to these parameters with 1.63 % standard deviation of the average MOTA score. Based on the maximal value in Fig. 5(b), we set $\lambda_8 = 0.3$, $\lambda_9 = 0.6$, and $\lambda_{10} = 0.1$.

Finally, we conduct experiments to validate the influence of the degree of hyper-edge k on tracking performance. We construct STV hyper-graph with $k = 2, \cdots, 8$ while keeping all other parameters fixed, and report the changes in the average MOTA score over three sequences in PETS 2009 dataset in Fig. 6. As these results show, tracking performance decreases as k increases when $k \geq 6$, because STV hyper-graph with hyper-edge degree that is too high fails to describe the motion pattern well enough, for the case of targets moving in drastically different speed and directions. Thus, we choose the degree of the hyper-edge $k = 3$ in our experiments.

For other parameters, we use the following default values for the parameters in our algorithm. The sensitivity controlling parameters of the affinity score to the prediction errors and the deviation of smooth trajectories in the motion and trajectory smoothness affinity calculations in Eqs. (7) and (8) are set as $\lambda_4 = 0.01$ and $\lambda_5 = 0.05$, respectively. Meanwhile, the sensitivity controlling parameter of the weight to the reliability score in Eq. (9) is set as $\lambda_6 = 1.0$, and the trade-off parameter between the scattering and number of associated 2D views is set as $\lambda_7 = 0.01$. The minimal size of subgraph is set as $\beta^* = 2$. We set the score threshold as $\mu = 0.2$ and the $\xi = 5$ in CS generation. The minimal length of the target trajectory $\ell = 5$, i.e., each tracked trajectory must contain 5 detections. These parameters are chosen empirically, i.e., make grid search of one parameter over a range of values while keep other parameters fixed, and we find that the performance of our algorithm are relatively insensitive to small perturbations of the parameters.

4.4 Performance Evaluation and Comparison

We evaluate our method on both single-camera and multi-camera multi-target tracking tasks and discuss the results in the following sections. We use the same input frame detections obtained by the DPM algorithm (Felzenszwalb et al. 2008) and the ground truth annotation provided in Milan et al. (2011) for all the evaluated methods on the PETS 2009 dataset. For the MOTChallenge 2015 3D benchmark, the publicly provided input detection results (Leal-Taixé et al. 2015) are adopted to complete the tracking task. Our main purpose here is to discount the difference in the detection methods, so as to perform a comprehensive evaluation of our method on the data association part, and provide fair comparison with other methods. On the other hand, because of this setting, performances of many evaluated methods may differ from their published results.

4.4.1 Multi-Camera Tracking Evaluation

In Table 2 and Fig. 7, we compare quantitative performance of our method with several state-of-the-art multi-camera multi-target tracking methods on the PETS 2009 dataset. The quantitative results of the trackers shown in Table 2 are calculated with the hit/miss distance threshold 1 m. We include performances from two existing multi-camera multi-target tracking methods (Leal-Taixé et al. 2012; Hofmann et al. 2013) for comparison. For a fair comparison, we use the same frame detections obtained with the DPM algorithm (Felzenszwalb et al. 2008) as the input to all methods. However, we cannot obtain the source code or binary executable that can reproduce the performances reported in Hofmann et al. (2013). As such, results in Table 2 are based on our own implementation of this work, with our best effort to follow the steps given in the original paper, for comparison[6]. In addition, for clarification and completeness, we also report the tracking results presented in Hofmann et al. (2013) in Table 2[7]. Some qualitative tracking results of our STV hyper-graph are presented in Fig. 9.

We highlight several points regarding the quantitative results in Table 2. Tracking performance is improved by using multi-cameras for all three datasets, where the performance

[6] We will make our method and our implementation of Hofmann et al. (2013) along with the tracking results available after the paper decision.

[7] Since different input detections and ground truth are used, it is unfair to directly compare the tracking results of the proposed method with the results presented in Hofmann et al. (2013).

Table 2 Multi-camera multi-target tracking results in the PETS 2009 dataset. The tracking results of the methods are obtained by running the publicly available codes with the same detection results and ground truth used in our tracker. The number in the bracket of average performance indicates the number of cameras used in each tracking scenario. The symbol ↑ means higher scores indicate better performance while ↓ means lower scores indicate better performance

Sequence	Method	Camera IDs	AvgRank ↓	MOTA[%] ↑	MOTP[%] ↑	GT	MT[%] ↑	ML[%] ↓	FM ↓	IDS ↓
PETS S2.L1	Berclaz et al. (2009)†	1+3+5+6+8	-	82.00	56.00	19	-	-	-	-
	Hofmann et al. (2013)†	1+5	-	99.40	82.90	-	100.00	0.00	1	1
	Hofmann et al. (2013)†	1+5+7	-	99.40	83.00	-	100.00	0.00	1	2
	Leal-Taixé et al. (2012)	1+5	-	85.74	67.87	19	89.47	0.00	115	150
	Leal-Taixé et al. (2012)	1+5+7	-	82.06	66.23	19	89.47	0.00	125	270
	Hofmann et al. (2013)*	1+5	-	91.89	79.50	19	94.74	0.00	29	41
	Hofmann et al. (2013)*	1+5+7	-	91.66	79.40	19	94.74	0.00	31	45
	Ours	1+5	-	95.51	80.60	19	100.00	0.00	12	14
	Ours	1+5+7	-	95.08	79.80	19	100.00	0.00	13	13
PETS S2.L2	Hofmann et al. (2013)†	1+2	-	87.60	73.50	-	86.00	0.00	128	111
	Hofmann et al. (2013)†	1+2+3	-	79.70	74.20	-	69.80	2.30	129	132
	Leal-Taixé et al. (2012)	1+2	-	40.14	54.13	43	4.65	9.30	581	621
	Leal-Taixé et al. (2012)	1+2+3	-	36.38	53.83	43	2.33	9.30	678	865
	Hofmann et al. (2013)*	1+2	-	58.97	65.80	43	25.56	2.33	288	385
	Hofmann et al. (2013)*	1+2+3	-	58.85	66.00	43	30.23	2.33	293	388
	Ours	1+2	-	67.00	61.50	43	51.16	0.00	239	239
	Ours	1+2+3	-	65.24	61.80	43	44.19	0.00	246	249
PETS S2.L3	Hofmann et al. (2013)†	1+2	-	68.50	72.30	-	54.50	20.50	149	156
	Hofmann et al. (2013)†	1+2+4	-	65.40	73.90	-	40.90	25.00	88	116
	Leal-Taixé et al. (2012)	1+2	-	48.49	51.74	44	22.73	9.09	250	279
	Leal-Taixé et al. (2012)	1+2+4	-	40.22	49.46	44	9.09	15.91	234	300
	Hofmann et al. (2013)*	1+2	-	54.39	60.20	44	25.00	25.00	67	106
	Hofmann et al. (2013)*	1+2+4	-	49.79	63.00	44	29.55	25.00	80	123
	Ours	1+2	-	57.06	59.30	44	38.64	15.91	120	129
	Ours	1+2+4	-	54.39	54.90	44	29.55	20.45	99	92
Average	Hofmann et al. (2013) (2)†	-	-	85.17	76.23	-	80.17	6.83	92.67	89.33
	Hofmann et al. (2013) (3)†	-	-	81.50	77.03	-	70.23	9.10	72.67	83.33
	Leal-Taixé et al. (2012) (2)	-	4.50	58.12	57.91	-	38.95	6.13	315.33	350.00
	Leal-Taixé et al. (2012) (3)	-	5.67	52.89	56.51	-	33.63	8.40	345.67	478.33
	Hofmann et al. (2013) (2)*	-	3.33	68.42	68.50	-	48.43	9.11	128.00	177.33
	Hofmann et al. (2013) (3)*	-	3.50	66.77	69.47	-	51.51	9.11	134.67	185.33
	Ours (2)	-	1.67	73.19	67.13	-	63.27	5.30	123.67	127.33
	Ours (3)	-	2.17	71.57	65.50	-	57.91	6.82	119.33	118.00

† The tracking results of the methods are copied directly from the published papers. Since different input detections and ground truth are used, it is unfair to directly compare the tracking results of the proposed method with these results directly copied from the published papers. For clarification and completeness, we also report them in the table

* The tracking results are based on our own implementation of (Hofmann et al. 2013), with our best effort to follow the steps given in the original paper, and using the same input detections and ground truth as our tracker

gains for videos with higher target densities (S2.L2 and S2.L3) are particularly significant. This is due to the complementary information provided from multiple camera views, which helps to resolve appearance ambiguity and occlusions. However, performance gain with multi-cameras decreases in the cases of low target density (S2.L1), where single-camera tracking already saturates the performance metric. Yet, further increasing the number of camera views does not usually lead to a monotonic increase in performances, e.g., Leal-Taixé et al. (2012) (2) produces 5.23 % larger average MOTA score than Leal-Taixé et al. (2012) (3), and Ours (2) produces 1.62 % larger average MOTA score than Ours (3). This is due to errors in camera calibration that lead to inaccuracies in the mapping function ϕ^v. These errors result in

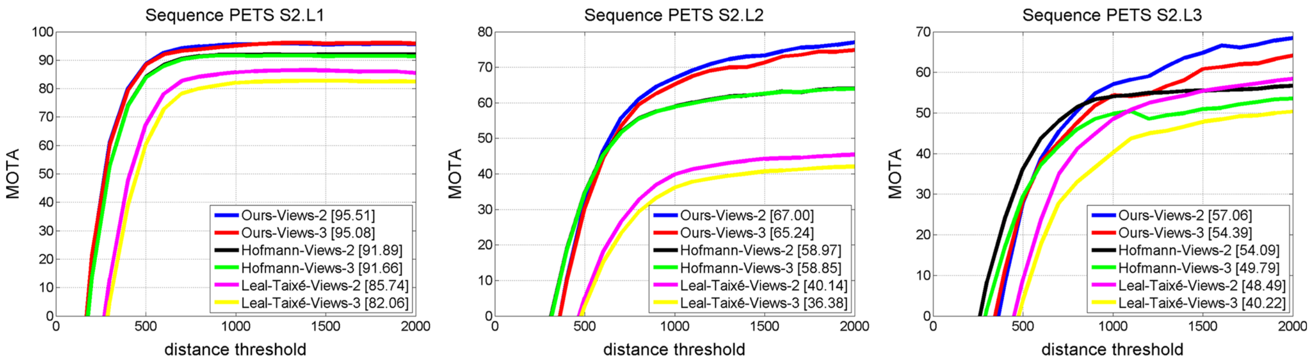

Fig. 7 Plots of the MOTA score with different hit/miss distance thresholds, i.e., varying from 0 to 2000 mm, of our approach and two state-of-the-art methods, i.e., Leal-Taixé et al. (2012) and Hofmann et al. (2013), for multi-camera multi-target tracking in PETS 2009 S2.L1, S2.L2 and S2.L3 sequences. The performance score for each tracker is presented in the *legend*

Table 3 Single-camera multi-target tracking results in the PETS 2009 dataset. All methods use the video sequence capture by camera #1 to complete the tracking task. The tracking results of the methods are obtained by running the publicly available codes with the same detection results and ground truth used in our tracker. The symbol ↑ means higher scores indicate better performance while ↓ means lower scores indicate better performance

Sequence	Method	AvgRank ↓	MOTA [%] ↑	MOTP [%] ↑	GT	MT [%] ↑	ML [%] ↓	FM ↓	IDS ↓
PETS S2.L1	Breitenstein et al. (2011)†	-	75.00	60.00	19	-	-	-	-
	Kuo and Nevatia (2011)†	-	-	-	19	78.90	0.00	23	1
	Yang and Nevatia (2012a)†	-	-	-	19	89.50	0.00	9	0
	Shi et al. (2014)†	-	96.10	81.80	19	94.70	0.00	6	4
	Dehghan et al. (2015)†	-	90.40	63.12	19	95.00	0.00	-	3
	Hofmann et al. (2013)†	-	98.00	82.80	-	100.00	0.00	11	10
	Berclaz et al. (2011)	-	75.05	77.00	19	63.16	0.00	63	38
	Andriyenko and Schindler (2011)	-	73.44	78.20	19	52.63	15.79	15	34
	Andriyenko et al. (2012)	-	89.05	78.10	19	84.21	0.00	21	26
	Pirsiavash et al. (2011)	-	81.59	71.80	19	68.42	0.00	71	63
	Wen et al. (2014)	-	94.43	74.50	19	94.74	0.00	16	13
	Leal-Taixé et al. (2012)	-	84.90	67.95	19	84.21	0.00	107	101
	Hofmann et al. (2013)*	-	91.57	80.30	19	94.74	0.00	38	52
	Ours	-	95.44	80.80	19	100.00	0.00	10	10
PETS S2.L2	Hofmann et al. (2013)†	-	75.80	72.10	-	65.10	0.00	252	234
	Berclaz et al. (2011)	-	41.60	63.00	43	2.33	13.95	416	244
	Andriyenko and Schindler (2011)	-	35.21	69.50	43	9.30	25.58	91	118
	Andriyenko et al. (2012)	-	49.99	64.30	43	9.30	2.33	261	292
	Pirsiavash et al. (2011)	-	34.50	69.90	43	9.30	4.65	793	2509
	Wen et al. (2014)	-	55.32	58.40	43	11.63	2.33	205	141
	Leal-Taixé et al. (2012)	-	36.03	53.56	43	4.65	11.63	514	508
	Hofmann et al. (2013)*	-	55.11	70.30	43	9.30	6.98	303	350
	Ours	-	59.15	65.70	43	34.88	0.00	259	239
PETS S2.L3	Hofmann et al. (2013)†	-	62.80	70.50	-	54.50	11.40	217	225
	Berclaz et al. (2011)	-	39.85	60.60	44	15.91	18.18	159	196
	Andriyenko and Schindler (2011)	-	51.65	57.20	44	29.55	18.18	99	153
	Andriyenko et al. (2012)	-	46.12	58.90	44	20.45	20.45	126	168
	Pirsiavash et al. (2011)	-	49.79	65.40	44	27.27	25.00	149	172
	Wen et al. (2014)	-	50.30	55.10	44	22.73	22.73	47	38

Table 3 continued

Sequence	Method	AvgRank ↓	MOTA [%] ↑	MOTP [%] ↑	GT	MT [%] ↑	ML [%] ↓	FM ↓	IDS ↓
	Leal-Taixé et al. (2012)	-	48.90	51.67	44	22.73	11.36	241	224
	Hofmann et al. (2013)*	-	46.60	65.20	44	20.45	34.09	64	88
	Ours	-	53.36	59.20	44	25.00	18.18	115	100
Average	Berclaz et al. (2011)	6.33	52.17	66.87	-	27.13	10.71	212.67	159.33
	Andriyenko and Schindler (2011)	4.83	53.43	68.30	-	30.49	19.85	68.33	101.67
	Andriyenko et al. (2012)	4.17	61.72	67.10	-	37.99	7.59	136.00	162.00
	Pirsiavash et al. (2011)	5.83	55.29	69.03	-	35.00	9.88	337.67	914.67
	Wen et al. (2014)	3.00	66.68	62.67	-	43.03	8.35	89.33	64.00
	Hofmann et al. (2013)†	-	78.87	75.13	-	73.20	3.80	160.00	156.33
	Leal-Taixé et al. (2012)	5.83	56.61	57.73	-	37.20	7.66	287.33	277.67
	Hofmann et al. (2013)*	4.00	64.43	71.93	-	41.50	13.69	135.00	163.33
	Ours	2.00	69.32	68.57	-	53.29	6.06	128.00	116.33

† The tracking results of the methods are copied directly from the published papers. Since different input detections and ground truth are used, it is unfair to directly compare the tracking results of the proposed method with these results directly copied from the published papers. For clarification and completeness, we also report them in the table

* The tracking results are based on our own implementation of (Hofmann et al. 2013), with our best effort to follow the steps given in the original paper, and using the same input detections and ground truth as our tracker

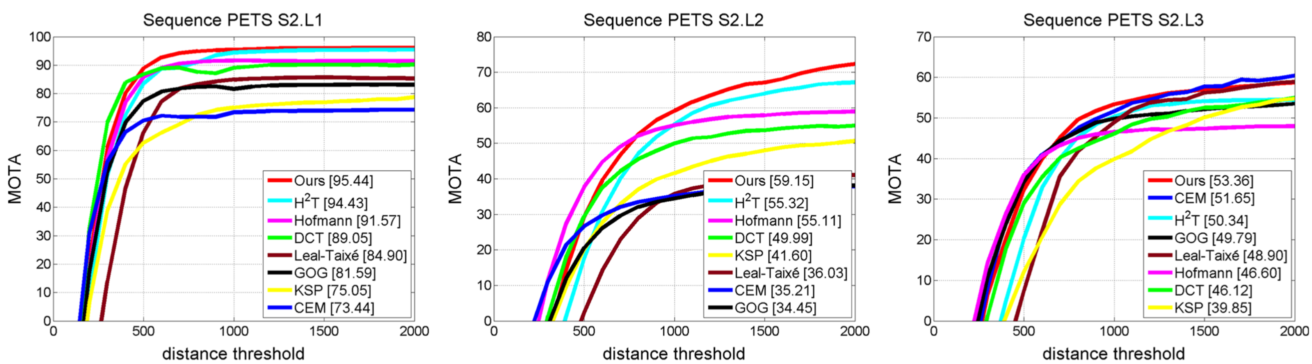

Fig. 8 Plots of the MOTA score with different hit/miss distance thresholds, i.e., varying from 0 to 2000 mm, of our method and several state-of-the-art methods, i.e., H^2T (Wen et al. 2014), DCT (Andriyenko et al. 2012), CEM (Andriyenko and Schindler 2011), GOG (Pirsiavash et al. 2011), KSP (Berclaz et al. 2011), Leal-Taixé et al. (2012), and Hofmann et al. (2013), for single-camera multi-target tracking in PETS 2009 S2.L1, S2.L2 and S2.L3 sequences. The performance score for each tracker is presented in the *legend*

incorrect associations that accumulate with increasing frame detections in multiple views, and lead to incorrect couplings (false positives), which greatly influence the performance of the trackers. Although our method is relative more robust to camera calibration errors than (Leal-Taixé et al. 2012) (the average MOTA score gap between two camera views and three camera views is reduced to 1.62 from 5.23 %), by integrating the higher-order dependencies among couplings, it is not entirely satisfactory. Thus, the way to restrain the errors in camera calibration while exploring effective information in multi-camera to help tracking is still worth study.

In comparison with the state-of-the-art multi-camera multi-target tracking algorithms (Leal-Taixé et al. 2012; Hofmann et al. 2013) based on associating pairs of 3D couplings, our method achieves better performance as reflected by lowest AvgRank score, which is determined by higher MOTA and MT scores and lower IDS and FM scores. This shows the effectiveness of using higher-order temporal correlations among couplings encoded by STV hyper-graph, which greatly reduces the association ambiguities, indicated by the lower IDS and FM scores.

4.4.2 Single-Camera Tracking Evaluation

PETS 2009 dataset We first evaluate our method in handling the single-camera multi-target tracking task on the first view of each sequence in PETS 2009 dataset (i.e., S2.L1, S2.L2, and S2.L3 sequences). We compare our approach with several state-of-the-art single-camera multi-target tracking algorithms (Breitenstein et al. 2011; Kuo and Nevatia 2011;

Fig. 9 STV hyper-graph tracking results on PETS 2009 videos. We show results using three camera views and two different frames, as well as the top down view of the overall tracking results. This figure is better viewed in *color* (Color figure online)

Table 4 Quantitative results on the single-camera MOTChallenge 3D benchmark (Leal-Taixé et al. 2015)

Method	AvgRank	MOTA [%] ↑	MOTP [%] ↑	FAF ↓	MT [%] ↑	ML [%] ↓	FP ↓	FN ↓	IDS ↓	FM ↓
DBN Klinger et al. (2015)	1.3	51.1	61.0	2.3	28.7	17.9	2077	5746	380	418
LPSFM Leal-Taixé et al. (2011)	2.7	35.9	54.0	2.3	13.8	21.6	2031	8206	520	601
LP3D Milan et al. (2015)	3.6	35.9	53.3	4.0	20.9	16.4	3588	6593	580	659
KalmanSFM Pellegrini et al. (2009)	4.1	25.0	53.6	3.6	6.7	14.6	3161	7599	1838	1686
Ours	3.3	34.2	55.8	3.5	11.2	25.4	3057	7454	532	611

Yang and Nevatia 2012a; Shi et al. 2014; Dehghan et al. 2015; Berclaz et al. 2011; Leal-Taixé et al. 2012; Hofmann et al. 2013; Andriyenko and Schindler 2011; Andriyenko et al. 2012; Pirsiavash et al. 2011; Wen et al. 2014), with the results presented in Table 3 and Fig. 8. The quantitative tracking results shown in Table 3 are calculated with the hit/miss distance threshold 1 m. As previously mentioned, for a fair comparison, we use the same frame detections obtained with the DPM algorithm (Felzenszwalb et al. 2008) as the input to all methods.

As shown in Table 3 and Fig. 8, our method performs the best in two sequences, i.e., S2.L2 and S2.L3, while performs competitively in the sequence S2.L1, and achieves the best performance with the lowest AvgRank score comparing with the state-of-the-art trackers. Comparing with the previous methods (Berclaz et al. 2011; Leal-Taixé et al. 2012; Andriyenko and Schindler 2011; Andriyenko et al. 2012; Pirsiavash et al. 2011; Hofmann et al. 2013) merely using the pairwise similarities between tracklets, our method exploits the higher-order similarities among multiple tracklets in a hyper-graph such that full motion information can be used to improve the performance, especially in the crowded scenes, e.g., S2.L2 (Fig. 9b) and S2.L3 (Fig. 9c) sequences.

In addition, compared to Wen et al. (2014), the other hypergraph based single-camera tracking method, our method also achieves better performance for single-camera tracking with 2.64 and 10.26 % gain of MOTA and MT scores on average. This is due to the use of calibrated cameras in 3D space, as depth information from the ground-plane assumption improves the motion and trajectory smoothness affinity estimations, which improves overall tracking performance.

MOTChallenge 2015 3D benchmark We also report the experiment results on the MOTChallenge 3D benchmark (Leal-Taixé et al. 2015) in Table 4. As presented in Table 4, our approach achieves competitive performance with the state-of-the-art single-camera 3D multi-target tracking methods (Leal-Taixé et al. 2011; Milan et al. 2015; Pellegrini et al. 2009; Klinger et al. 2015) according to the AvgRank score. The algorithm (Klinger et al. 2015) integrates some prior knowledge of the scenes and learns the object appearance online using the online random forest classifier, which makes the tracker achieve the best performance. However, using the complex object appearance model will bring the huge computational load, which affects the runtime of the algorithm (Klinger et al. 2015), i.e., it runs 0.1 Frame-Per-Second (fps). Incorporating higher-order connections among tracklets (i.e., motion constraints can be fully exploited) makes our approach achieves relative lower IDS and FM scores, which promotes its performance on single-camera multi-target tracking scenarios.

4.5 Discussion

Effectiveness of dense subgraphs To have a detailed understanding of the contribution of each component of our method, we construct a baseline tracker that uses pairwise correlation of 3D couplings (tracklets) as the method (Hofmann et al. 2013), and apply the dense subgraph search (Liu et al. 2012) as the tracking solution. We compare this baseline algorithm (marked as *Ours-P*) in Table 5 with our own implementation of the network flow based optimization method (Hofmann et al. 2013), for tracking with two camera views. This baseline algorithm improves 4.17 % MOTA and 12.55 % MT scores, and reduces 4.57 % ML and 29.5 % IDS scores on average performance compared to the network flow optimization based method (Hofmann et al. 2013), showing that our formulation of tracking as searching subgraphs or sub-hypergraphs is important in improving the overall performance.

Effectiveness of hyper-graph representation In addition, to exploit the effectiveness of hyper-graph representation, we compare our approach with the baseline tracker, i.e., Ours-P. Our full method improves 0.7 % MOTA and 2.28 % MT score, and reduces 3.38 % FM score on average performance in comparison with the baseline algorithm, which demonstrates that using hyper-graph as a representation can reduce the FM score and improve MT score to promote the multi-target tracking performance.

Running time In addition to its performance, our method also affords efficient running time. Table 6 reports the running time measured in fps on the PETS2009 dataset over 1–3 camera views after given the detection results. These running time is based on an implementation with unoptimized C++ code, single thread execution on a workstation with Intel 2.67GHz CPU and 128 GB memory. Note that, as shown in Table 6, our method runs faster in S2.L3 than S2.L2 over all

Table 5 Effect of different components in the proposed tracker. The symbol ↑ means higher scores indicate better performance while ↓ means lower scores indicate better performance

Sequence	Method	Camera IDs	MOTA [%] ↑	MOTP [%] ↑	GT	MT [%] ↑	ML [%] ↓	FM ↓	IDS ↓
PETS S2.L1	Hofmann et al. (2013)*	1+5	91.89	79.50	19	94.74	0.00	29	41
	Ours-P	1+5	96.40	80.80	19	100.00	0.00	10	6
	Ours	1+5	95.51	80.60	19	100.00	0.00	12	14
PETS S2.L2	Hofmann et al. (2013)*	1+2	58.97	65.80	43	25.56	2.32	288	385
	Ours-P	1+2	63.50	61.70	43	51.16	0.00	252	249
	Ours	1+2	67.00	61.50	43	51.16	0.00	239	239
PETS S2.L3	Hofmann et al. (2013)*	1+2	54.39	60.20	44	25.00	25.00	67	106
	Ours-P	1+2	57.60	59.40	44	31.82	13.63	122	120
	Ours	1+2	57.06	59.30	44	38.64	15.91	120	129
Average	Hofmann et al. (2013)*	-	68.42	68.50	-	48.43	9.11	128.00	177.33
	Ours-P	-	72.50	67.30	-	60.99	4.54	128.00	125.00
	Ours	-	73.19	67.13	-	63.27	5.30	123.67	127.33

* The tracking results are based on our own implementation of Hofmann et al. (2013), with our best effort to follow the steps given in the original paper, and using the same input detections and ground truth as our tracker

Table 6 The running speed of our method in different sequences with different camera views. Frame-Per-Second (fps) is used to measure the speed of the tracker. In comparison, we also show the frame rate of the original PETS 2009 videos

Sequence	1-view	2-views	3-views	PETS frame rate
S2.L1	30.6	16.8	10.9	7.0
S2.L2	7.1	1.8	0.9	7.0
S2.L3	9.0	3.2	2.5	7.0

camera views. Although S2.L3 has higher target density than S2.L2, the highly complex pedestrian interactions in S2.L2 result in more hyper-edges are included in STV hyper-graph. Thus, the dense sub-hypergraph search is slower in S2.L2 than that in S2.L3.

5 Conclusion

Incorporating multiple cameras is an effective solution to improve the performance and robustness of multi-target tracking to occlusion and appearance ambiguities. In this paper, we propose a new multi-camera multi-target tracking method based on a space-time-view hyper-graph that encodes higher-order constraints (i.e., beyond pairwise relations) on 3D geometry, appearance, motion continuity, and trajectory smoothness among 2D tracklets within and across different camera views. We solve tracking in each single view and reconstruction of tracked trajectories in 3D environment simultaneously by formulating the problem as an efficient search of dense sub-hypergraphs on the space-time-view hyper-graph using a sampling based approach. Experimental results on the PETS 2009 benchmark dataset and MOTChallenge 2015 3D benchmark demonstrate that our method performs favorably against the state-of-the-art methods in both single-camera and multi-camera multi-target tracking, while achieving close to real-time running efficiency. We also provide experimental analysis of the influence of various aspects of our method to the final tracking performance.

There are several directions we would like to further improve the current work. First, the current method relies on the knowledge of camera parameters, it is useful to be able to recover camera parameters along with multi-target tracking and 3D reconstruction. This is possible with recent advances that recover camera parameters from multiple image sequences (Kim et al. 2013); Kostrikov et al. 2014). Second, the current method also assumes a static camera, and a more challenging scenario that we will explore is when some views are from cameras with ego motion (e.g., PTZ cameras). Also, there exist alternative formulations of the sub-hypergraph search algorithm such as those based on hyper-graph Laplacians (Zhou et al. 2006). Subsequently we would like to investigate and compare different optimization strategies to solve the dense sub-hypergraph search problem. Last, we would like to push the limit test of multi-camera tracking methods, and extend similar methods to scenarios where camera views have less overlapping.

Acknowledgments We would like to thank Dawei Du for a number of suggestions that considerably improved the quality of this paper. Longyin Wen and Siwei Lyu were supported by US National Science Foundation Research Grant (CCF-1319800). Zhen Lei was supported by the National Key Research and Development Plan (Grant No. 2016 YFC0801002), the Chinese National Natural Science Foundation Projects #61375037, #61473291. Honggang Qi was supported by National Nature Science Foundation of China #61472388.

References

Andriyenko, A., & Schindler, K. (2011). Multi-target tracking by continuous energy minimization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1265–1272).

Andriyenko, A., Schindler, K., & Roth, S. (2012). Discrete-continuous optimization for multi-target tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1926–1933).

Attanasi, A., Cavagna, A., Castello, L. D., Giardina, I., Jelic, A., Melillo, S., et al. (2015). GReTA—a novel global and recursive tracking algorithm in three dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(1), 1.

Berclaz, J., Fleuret, F., & Fua, P. (2009). Multiple object tracking using flow linear programming. In *Winter-PETS* (pp. 1–8). Snowbird: IEEE.

Berclaz, J., Fleuret, F., Türetken, E., & Fua, P. (2011). Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(9), 1806–1819.

Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., & Gool, L. J. V. (2011). Online multi-person tracking-by-detection from a single, uncalibrated camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(9), 1820–1833.

Brendel, W., Amer, M. R., & Todorovic, S. (2011). Multiobject tracking as maximum weight independent set. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1273–1280).

Chari, V., Lacoste-Julien, S., Laptev, I., & Sivic, J. (2015). On pairwise costs for network flow multi-object tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5537–5545).

Dehghan, A., Tian, Y., Torr, P. H. S., & Shah, M. (2015). Target identity-aware network flow for online multiple target tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1146–1154).

Dollár, P., Wojek, C., Schiele, B., & Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(4), 743–761.

Felzenszwalb, P. F., McAllester, D. A., & Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8).

Ferryman, J. M., & Shahrokni, A. (2009). PETS2009: Dataset and challenge. In *Winter-PETS* (pp. 1–6).

Fleuret, F., Berclaz, J., Lengagne, R., & Fua, P. (2008). Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(2), 267–282.

Hofmann, M., Wolf, D., & Rigoll, G. (2013). Hypergraphs for joint multi-view reconstruction and multi-object tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3650–3657).

Hong, L., & Cui, N. (2000). An interacting multipattern joint probabilistic data association (imp-jpda) algorithm for multitarget tracking. *Signal Processing*, *80*(8), 1561–1575.

Huang, C., Li, Y., & Nevatia, R. (2013). Multiple target tracking by learning-based hierarchical association of detection responses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(4), 898–910.

Isard, M., & Blake, A. (1998). Condensation—conditional density propagation for visual tracking. *International Journal of Computer Vision*, *29*(1), 5–28.

Izadinia, H., Saleemi, I., Li, W., & Shah, M. (2012) (MP)^{2}T: Multiple people multiple parts tracker. In *Proceedings of European Conference on Computer Vision* (pp. 100–114).

Jiang, H., Fels, S., & Little, J. J. (2007). A linear programming approach for multiple object tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8).

Khan, Z., Balch, T. R., & Dellaert, F. (2005). MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(11), 1805–1918.

Kim, J., Dai, Y., Li, H., Du, X., & Kim, J. (2013). Multi-view 3D reconstruction from uncalibrated radially-symmetric cameras. In *Proceedings of IEEE International Conference on Computer Vision* (pp. 1896–1903).

Klinger, T., Rottensteiner, F., & Heipke, C. (2015). Probabilistic multi-person tracking using dynamic bayes networks. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *II–3/W5*, 435–442.

Kostrikov, I., Horbert, E., & Leibe, B. (2014). Probabilistic labeling cost for high-accuracy multi-view reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1534–1541).

Kuhn, W., & Tucker, A. (1951) Nonlinear programming. In *Proceedings of 2nd Berkeley Symposium* (pp. 481–492).

Kuo, C. H., & Nevatia, R. (2011). How does person identity recognition help multi-person tracking? In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1217–1224).

Leal-Taixé, L., Milan, A., Reid, I.D., Roth, S., & Schindler, K. (2015). Motchallenge 2015: towards a benchmark for multi-target tracking. CoRR abs/1504.01942.

Leal-Taixé, L., Pons-Moll, G., & Rosenhahn, B. (2011). Everybody needs somebody: modeling social and grouping behavior on a linear programming multiple people tracker. In *Workshops in Conjunction with IEEE International Conference on Computer Vision* (pp. 120–127).

Leal-Taixé, L., Pons-Moll, G., & Rosenhahn, B. (2012) Branch-and-price global optimization for multi-view multi-object tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1987–1994).

Leven, W. F., & Lanterman, A. D. (2009). Unscented kalman filters for multiple target tracking with symmetric measurement equations. *IEEE Transaction on Automatic Control*, *54*(2), 370–375.

Liu, H., & Yan, S. (2012). Efficient structure detection via random consensus graph. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 574–581).

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

Liu, Y., Li, H., & Chen, Y. Q. (2012). Automatic tracking of a large number of moving targets in 3d. In *Proceedings of European Conference on Computer Vision* (pp. 730–742).

Marchesotti, L., Marcenaro, L., Ferrari, G., & Regazzoni, C. S. (2002) Multiple object tracking under heavy occlusions by using kalman filters based on shape matching. In *Proceedings of IEEE International Conference on Image Processing* (pp. 341–344).

Milan, A. (2011) Continuous energy minimization tracker. http://www.milanton.de/contracking/index.html.

Milan, A., Leal-Taixé, L., Schindler, K., Roth, S., & Reid, I.D. (2015). Multiple object tracking benchmark: 3d mot. https://motchallenge.net/results/3D_MOT_2015/.

Milan, A., Roth, S., & Schindler, K. (2014). Continuous energy minimization for multitarget tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(1), 58–72.

Ojala, T., Pietikäinen, M., & Mäenpää, T. (2000). Gray scale and rotation invariant texture classification with local binary patterns. In *Proceedings of European Conference on Computer Vision* (pp. 404–420).

Pellegrini, S., Ess, A., Schindler, K., & Gool, L. J. V. (2009). You'll never walk alone: modeling social behavior for multi-target tracking.

In *Proceedings of IEEE International Conference on Computer Vision* (pp. 261–268).

Pirsiavash, H., Ramanan, D., & Fowlkes, C. C. (2011). Globally-optimal greedy algorithms for tracking a variable number of objects. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1201–1208).

Reid, D. B. (1979). An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, *24*, 843–854.

Shi, X., Ling, H., Hu, W., Yuan, C., & Xing, J. (2014). Multi-target tracking with motion context in tensor power iteration. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3518–3525).

Shu, G., Dehghan, A., Oreifej, O., Hand, E., & Shah, M. (2012). Part-based multiple-person tracking with partial occlusion handling. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1815–1821).

Smith, K., Gatica-Perez, D., & Odobez, J. M. (2005). Using particles to track varying numbers of interacting people. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 962–969).

Stiefelhagen, R., Bernardin, K., Bowers, R., Garofolo, J. S., Mostefa, D., & Soundararajan, P. (2006). The CLEAR 2006 evaluation. *CLEAR* (pp. 1–44). Berlin: Springer.

Wen, L., Li, W., Yan, J., Lei, Z., Yi, D., & Li, S. Z. (2014) Multiple target tracking based on undirected hierarchical relation hypergraph. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 3457–3464).

Wu, Z., Hristov, N.I., Kunz, T. H., & Betke, M. (2009). Tracking-reconstruction or reconstruction-tracking? Comparison of two multiple hypothesis tracking approaches to interpret 3D object motion from several camera views. In *Proceedings of the IEEE Workshop on Motion and Video Computing* (pp. 1–8).

Wu, Z., Kunz, T. H., & Betke, M. (2011). Efficient track linking methods for track graphs using network-flow and set-cover techniques. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1185–1192).

Yang, B., & Nevatia, R. (2012). Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1918–1925).

Yang, B., & Nevatia, R. (2012). An online learned CRF model for multi-target tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2034–2041).

Yang, M., Liu, Y., Wen, L., You, Z., & Li, S. Z. (2014). A probabilistic framework for multitarget tracking with mutual occlusions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8).

Yu, Q., & Medioni, G. G. (2009). Multiple-target tracking by spatiotemporal monte carlo markov chain data association. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(12), 2196–2210.

Zhang, L., Li, Y., & Nevatia, R. (2008). Global data association for multi-object tracking using network flows. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8).

Zhou, D., Huang, J., & Schölkopf, B. (2006). Learning with hypergraphs: Clustering, classification, and embedding. *Advances in Neural Information Processing Systems* (pp. 1601–1608). Cambridge: MIT Press.