Supplementary Material

The appendix complements our method by a detailed comparison with prior works, transformations between image coordinates and the world coordinates, a pseudocode representation of the proposed 3D geometry pre-clustering step, an additional sampling strategy for spatial-temporal edge costs, our lifted multicut solver, and a method to create tracking predictions in 3D space. Along with it, we present the performance of pre-processing the spatial-temporal tracking graph and ablation studies on the final edge costs accuracy, which depends on our pre-clustering algorithm. Lastly, we show the performance of LMGP compared with state-of-the-art single-camera methods, computational time, and some qualitative tracking results on WILDTRACK and Campus dataset.

A. LMGP Compared with Other Multi-Camera Trackers

We present in Table 5 the main differences between our LMGP method with other baselines. In summary, we utilize image correspondences for each object at each timeframe as the *Centralized Representation* strategy to correct ID-Switch errors generated by tracklets in *Single View-based methods*. On the contrary, using nodes as tracklets allows us to reduce the computational cost in the association step significantly, implying our tracker runs reasonably fast and, therefore, be feasible to deploy it in real-world tracking applications (Table 10).

Method	Single-Camera Tracklets	Centralized Representation	Online
LMGP (Ours)	✓	✓	
MLMRF [28]	✓		✓
STVH [42]	✓		
DMCT [47]		✓	✓
GLMB [32]		✓	✓
DyGLIP [33]	✓		✓

Table 5. Comparison of our LMGP w.r.t. recent multi-camera trackers. Single camera tracklets signifies for trackers whose inputs can be used as tracklets. Centralized representation refers to usage of relationships among detections at each timeframe. Online indicates for only current frames are used.

B. Transformation From Camera Coordinates to Word Coordinates on The Ground Plane

In this section, we describe two transformations which compute a corresponding 3D position $\tilde{\mathbf{X}}_\pi$ (inhomogeneous) on a ground plane τ for each point $\mathbf{x}$ in the image coordinate

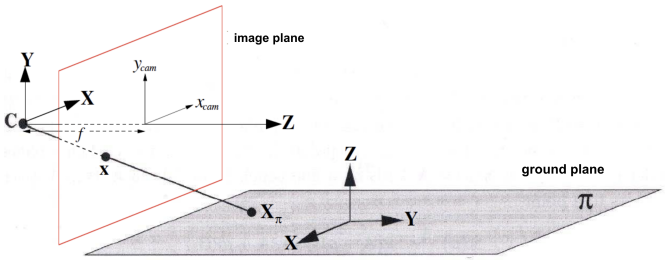

Figure 5. Projection scheme for the pinhole camera model. Given a $\mathbf{x}$ point in the image plane, $\mathbf{X}_\tau$ is its spatial point (homogeneous coordinate) in the τ ground plane. Image taken and adapted from [15].

and the world coordinate $\tilde{\mathbf{C}}$ of the camera centre $\mathbf{C}$ for the pinhole camera model (Figure 5).

We denote by $\mathbf{X}$ a point in space with world coordinates represented by a homogeneous vector $(\mathrm{X}, \mathrm{Y}, \mathrm{Z}, 1)^T \in \mathbb{R}^4$. Its homogeneous coordinates on the image plane $\mathbf{x} \in \mathbb{R}^3$ can be obtained by:

$$\mathbf{x} = \mathbf{P}\mathbf{X} \tag{16}$$

where $\mathbf{P}$ is the 3×4 homogeneous *camera projection matrix* written as:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}], \tag{17}$$

with $\mathbf{K}$ being the 3×3 *intrinsic matrix* and $[\mathbf{R}|\mathbf{t}]$ being the 3×4 *extrinsic matrix* where $\mathbf{R}$ is the 3×3 rotation matrix and $\mathbf{t}$ is the 3×1 translation vector.

Let $\mathbf{M} = \mathbf{K}\mathbf{R}$ and $\mathbf{p_4}$ is the last column vector of $\mathbf{P}$, the matrix $\mathbf{P}$ then can be written in the form

$$\mathbf{P} = [\mathbf{M}|\mathbf{p_4}]. \tag{18}$$

Since $\mathbf{K}$ and $\mathbf{R}$ in the finite projective camera model [15] are assumed to be non-singular, this means that $\mathbf{M}$ is also non-singular. We then can compute $\tilde{\mathbf{C}}$ by:

$$\tilde{\mathbf{C}} = -\mathbf{M}^{-1}\mathbf{p_4} \tag{19}$$

To specify $\mathbf{X}_\pi$ on the ground plane π, we need further constraints [15, 36]. In this setting, the coordinate systems of the camera and the surrounding space both have the z-axis as the height axis and the back projection line of an image point $\mathbf{x}$ hits the ground exactly when the z-coordinate of this line is set to zero since the origin of the spatial coordinate system is lying on the ground. The inhomogeneous coordinate $\tilde{\mathbf{X}}_\pi \in \mathbb{R}^3$ of $\mathbf{X}_\pi$ given a point $\mathbf{x}$ in the image coorindate is estimated by:

$$\tilde{\mathbf{X}}_\pi = -\frac{\tilde{C}_3}{\tilde{X}_3}\mathbf{M}^{-1}\mathbf{x} + \tilde{\mathbf{C}} \tag{20}$$

where $\tilde{C}_3$ and $\tilde{X}_3$ are obtained by:

$$\begin{aligned} [\tilde{X}_1, \tilde{X}_2, \tilde{X}_3]^T &= \mathbf{M}^{-1}\mathbf{x} \\ [\tilde{C}_1, \tilde{C}_2, \tilde{C}_3]^T &= \tilde{\mathbf{C}}. \end{aligned} \tag{21}$$

C. Pre-Clustering Algorithm

We present in Algorithm 1 the 3D geometry projection-based pre-clustering whose input is a set of detections across cameras at each time frame t and the output is a set of clusters for each object appearing in the scene. The linear assignment problem at line 8 exploits the 3D geometry constraints among points in the same object depicted in Figure 6-a. The code line 13 in algorithm 1 will verify in reverse directions (confident connections) for nodes in an initial cluster I_b to alleviate errors attained in inaccurately calibrated cameras and noisy detections. Figure 6-b illustrates this procedure with $I_1 = \{2, 3, 5, 6, 7\}$.

Algorithm 1: Pre-Clustering

Input: Timeframe t, detections B, r is a radius to scan nearby detections.
Output: $C = \{C_b : b \in B, \text{time}(b) = t\}$ is a set of clusters for each detection at timeframe t.

1 $D^t \leftarrow \{b \in B : \text{time}(b) = t\}$
2 $C \leftarrow \emptyset$
// Generate matches for each object
3 **for** $b \in D^t$ **do**
// Initialize
4 $j \leftarrow \text{cam}(b)$, $I_b \leftarrow \emptyset$, $C_b \leftarrow \emptyset$
// Define matching candidates in cam j
5 $B^{t,j}(b) = \{b' \in D^t : \text{dist}(h(b), h(b')) \leq r, \text{cam}(b') = j\}$
// Check one way connection
6 **for** *camera* $j' \neq j$ **do**
// Define matching candidates in cam j'
7 $B^{t,j'}(b) = \{b' \in D^t : \text{dist}(h(b), h(b')) \leq r, \text{cam}(b') = j'\}$
8 Solve linear assignment problem between $B^{t,j}(b)$ and $B^{t,j'}(b)$ with costs $\text{dist}(h(b_1), h(b_2))$ for $b_1 \in B^{t,j}(b)$, $b_2 \in B^{t,j'}(b)$
9 **if** b *is matched to some node* $b' \in B^{t,j'}(b)$ **then**
10 $I_b \leftarrow I_b \cup \{b'\}$
11 **end**
12 **end**
// Clustered points must be matched both ways (confident connections)
13 $C_b = \{b' \in I_b : b \in I_{b'}\}$
// Update set of clusters
14 $C \leftarrow C \cup \{C_b\}$
15 **end**
16 **return** C

D. Learning Spatial-Temporal Affinity and Correcting ID-Switch Positions

D.1. Sampling Training Data

We describe in Algorithm 3, 4 the training data sampling for f_{spatial}, f_{temporal} networks used in generating affinity costs (Eqs. 23, 24), where $\text{Uni}(a, b)$ is the uniform distribution between a and b, $\text{id}(x)$ is the single ID of tracklet x. Pos_s, Neg_s are sets of samples in the same and different objects connected by spatial edges. Likely, Pos_t, Neg_t are outputs for temporal edges. For f_{split}, we create the training data by running the trained CenterTrack on the validation set in each dataset.

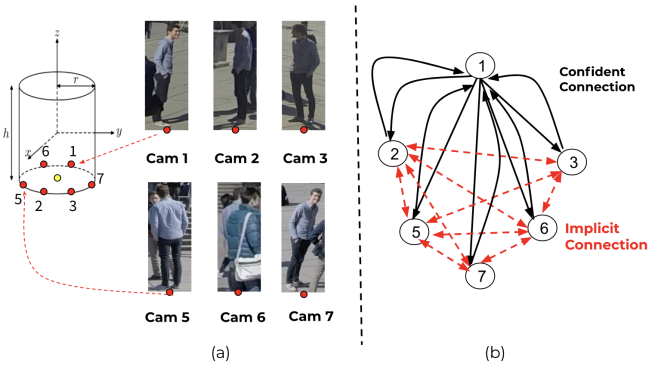

Figure 6. (a) The geometry constraint of 2D foot points belong to the same object on the 3D space, (b) An illustration of our pre-clustering in WILDTRACK with 7 cameras where the object is not visible at camera 4.

D.2. Network Architecture

For both f_{spatial}, f_{temporal} and f_{split}, we employed the same architecture as depicted in Figure 7 given the input feature vectors with d dimensions. We train f_{spatial}, f_{temporal} with 2 epochs and f_{split} with 30 epochs using Adam optimizer [27]. A modified F1 loss function [24] is also applied during the training process to mitigate problems caused by imbalanced training data.

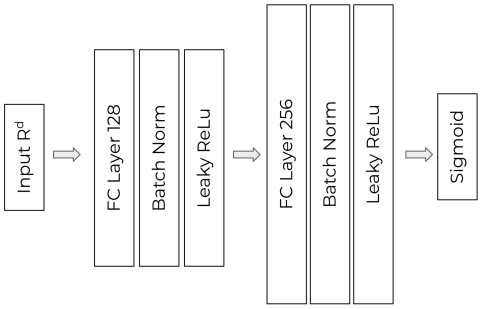

Figure 7. The binary network architecture used to train for f_{spatial}, f_{temporal} and f_{split}.

E. Solver for Lifted Multicut for Tracking

Since the lifted multicut problem is NP-hard to solve [25], we resort to efficient heuristics. We first compute a preliminary partition with the Greedy Additive Edge Contraction (GAEC) algorithm, followed by an improvement step with the Kernighan-Lin local search (KL-Local) procedure [25]. We summarize the used solver in an Algorithm 2. These heuristics have been shown to yield high quality solutions in practice [29].

Algorithm 2: Solver for LMGP Tracker

Input: Spatial-temporal graph
$G = (V, F)$, $F = E^t \cup E^s \cup E^c$
Edge costs c_{e^t}, c_{e^s}, M
Output: Edge label function $y : F \to \{0, 1\}$

$y \leftarrow \text{Run GAEC}\,(G,\, c_{e^t},\, c_{e^s},\, M)$
$y \leftarrow \text{Run KL} - \text{Local}\,(G,\, c_{e^t},\, c_{e^s},\, M,\, y)$
return y

F. Generating Tracking Predictions in 3D Space

Given clusters of the spatial-temporal tracking graph, we interpolate trajectories in 3D-coordinates. Since a cluster consists of a set of tracklets, which in turn consist of a set of detections, we directly associate a cluster with its underlying set of detections. Hence, let a cluster consisting of detections $\{b_1, \ldots, b_s\}$ be given. At each timestep t such that there exists at least one detection in that timestep, we obtain a 3D position through

$$
\begin{aligned}
p^t_{avg} &= \frac{\sum_{i:\text{time}(b_i)=t} h(b_i)}{|\{i : \text{time}(b_i) = t\}|} \\
p^t_{3D} &= \underset{p:\|p - p^t_{avg}\| \leq r}{\arg\max} \sum_{i:\text{time}(b_i)=t} \|\pi_{\text{cam}(b_i)}(p), b_i\|^2_{iou}
\end{aligned}
\tag{22}
$$

where $\pi_j(p)$ is the projection of a 3D cylinder centering at p to a rectangle bounding box on 2D coordinates of camera j. The above interpolation step in Equation 22 first takes the average position of the 3D-projections of all relevant detections of a given timepoint. Second, a 3D position in the vicinity of the mean position is calculated such that the reprojection error w.r.t. IoU is minimized.

G. Experimental Results

G.1. Implementation Details

Generating Tracklets For both PETS-09 [12], Campus [44], and WILDTRACK [7] datasets, we use CenterTrack [53] trained on the CrowdHuman dataset [37] and pre-trained it on training sequences in each dataset with the following settings: heatmap noise 0.05, tracklets confidence 0.4, false positive rate 0.1, and batch size of 32 images. All other parameters are identical with the default settings. Given trained models, tracklets are generated by running CenterTrack with the provided detections and employ a tracking threshold 0.2, pre-threshold 0.4, which are scores for predicting a bounding box and feeding the heatmap to the next frame, respectively.

Algorithm 3: Sampling spatial edges

Input: $\Gamma = \{\tau : \text{cam}(\tau) \in \{1, 2, \ldots, m\}\}$ is a set of all ground truth trajectories across m cameras.
n is the number of random sampling.
Output:

$$\text{Pos}_s = \{(x, y) : \text{cam}(x) \neq \text{cam}(y), \text{id}(x) = \text{id}(y), \text{time}(x) \cap \text{time}(y) \neq \varnothing\}$$
$$\text{Neg}_s = \{(x, y) : \text{cam}(x) \neq \text{cam}(y), \text{id}(x) \neq \text{id}(y), \text{time}(x) \cap \text{time}(y) \neq \varnothing\}$$

// Initialize
$\text{Pos}_s \leftarrow \emptyset$, $\text{Neg}_s \leftarrow \emptyset$
repeat
  for each *camera* $j \in \{1, \ldots, m\}$ **do**
    for each $\tau = (b_1, b_2, \ldots, b_{|\tau|}) : \text{cam}(\tau) = j$ **do**
      // Randomly remove an interval in τ
      $i \sim \text{Uni}(1, |\tau|)$, $k \sim \text{Uni}(i, |\tau| - i - 1)$
      $x \leftarrow (b_1, \ldots, b_{i-1})$, $y \leftarrow (b_{i+k+1}), \ldots, b_{|\tau|})$
      $\text{R}^j_\tau \leftarrow \{x, y\}$
    end
  end
  // Initialize balanced sampling
  $n_{\text{balance}} \leftarrow 0$
  // Generate positive pairs
  for each $\tau, \tau' : \text{cam}(\tau) \neq \text{cam}(\tau')$ ***and*** $\text{id}(\tau) = \text{id}(\tau')$ ***and*** $\text{time}(\tau) \cap \text{time}(\tau') \neq \emptyset$ **do**
    for each $(x, y) \in \text{R}^{\text{cam}(\tau)}_\tau \times \text{R}^{\text{cam}(\tau')}_{\tau'}$ **do**
      $\text{Pos}_s \leftarrow \text{Pos}_s \cup \{(x, y)\}$
      $n_{\text{balance}} \leftarrow n_{\text{balance}} + 1$
    end
  end
  // Generate negative pairs
  for each $\tau, \tau' : \text{cam}(\tau) \neq \text{cam}(\tau')$ ***and*** $\text{id}(\tau) \neq \text{id}(\tau')$ ***and*** $\text{time}(\tau) \cap \text{time}(\tau') \neq \emptyset$ **do**
    for each $(x, y) \in \text{R}^{\text{cam}(\tau)}_\tau \times \text{R}^{\text{cam}(\tau')}_{\tau'}$ **do**
      // Balanced Sampling
      if $n_{balance} > 0$ **then**
        $\text{Neg}_s \leftarrow \text{Neg}_s \cup \{(x, y)\}$
        $n_{\text{balance}} \leftarrow n_{\text{balance}} - 1$
      end
    end
  end
until *n times*
return Pos_s, Neg_s

Multi-view Appearance Feature We utilize DG-Net [52], one of the best performing methods for re-identification to extract embedding vectors for detections. To train DG-Net, the training data is selected from pre-clustering of unoccluded detections C'_b for each object (Section 3.1, main paper). We use the best models trained on Market-151 [51] and DukeMTMC-Reid [34] as the "teacher network" in DG-Net. Noting that pre-training on other datasets is a common practice in MOT, and pre-training on the above datasets was e.g. done in [21]. While our method benefits from strong appearance features from DG-Net, these features alone are not mainly responsible for our performance, see Table 1 and 2 in the main paper with `LMGP w/o Pre-Clustering`, which mainly relies on appearance features and does not include 3D-information.

Algorithm 4: Sampling temporal edges

Input: $\Gamma = \{\tau : \text{cam}(\tau) \in \{1, 2, \dots, m\}\}$ is a set of all trajectories across m cameras.
n is the number of random sampling.

Output:

$$\text{Pos}_t = \{(x, y) : \text{cam}(x) = \text{cam}(y), \text{id}(x) = \text{id}(y), \text{time}(x) \cap \text{time}(y) = \varnothing\}$$
$$\text{Neg}_t = \{(x, y) : \text{cam}(x) = \text{cam}(y), \text{id}(x) \neq \text{id}(y), \text{time}(x) \cap \text{time}(y) = \varnothing\}$$

```
// Initialize
Pos_t ← ∅, Neg_t ← ∅
repeat
    Select randomly camera j
    // Initialize balanced sampling
    n_balance ← 0
    for each τ = (b_1, b_2, ..., b_|τ|) : cam(τ) = j do
        // Randomly remove an interval in τ
        i ~ Uni(1, |τ|), k ~ Uni(i, |τ| − i − 1)
        x ← (b_1, ..., b_{i−1}), y ← (b_{i+k+1}, ..., b_|τ|)
        // Generate positive pairs
        Pos_t ← Pos_t ∪ {(x, y)}
        R_τ ← {x, y}
        n_balance ← n_balance + 1
    end
    // Generate negative pairs
    for each τ, τ' : id(τ) ≠ id(τ') do
        for each (x, y) ∈ R_τ × R_τ' : time(x) ∩ time(y) = ∅ do
            // Balanced Sampling
            if n_balance > 0 then
                Neg_t ← Neg_t ∪ {(x, y)}
                n_balance ← n_balance − 1
            end
        end
    end
until n times
return Pos_t, Neg_t
```

G.2. Ablation Study of Affinity Costs

Since the performance of a tracking system depends highly on the accuracy of local connections, this section validates the effect of several proposed affinities, which form our edge costs c_{e^t} and c_{e^s}. In particular,

$$c_{e^t} = f_{\text{temporal}}(c^{\text{app}}_{\text{index}}(\tau, \tau'),\ c^{\text{fw,t}}(\tau, \tau'),\ c^{\text{bw,t}}(\tau, \tau'),\ t(\tau, \tau')) \tag{23}$$

$$c_{e^s} = f_{\text{spatial}}(c^{\text{fw,s}}(\tau, \tau'),\ c^{\text{bw,s}}(\tau, \tau'),\ c^{\text{app}}(\tau, \tau'),\ c^{\text{avg3D}}(\tau, \tau'), c^{\text{pc}}(\tau, \tau')) \tag{24}$$

where $\text{index} \in \{\text{best}, \text{min}, \text{max}, \text{avg}, \text{std}\}$, described at the Equation (4), Subsection 3.2 in the main paper. For spatial cost in Equation 24, the appearance part $c^{\text{app}}(\tau, \tau')$ only uses the visible detections at current cameras.

Table 6 and 7 present ablation studies on WILDTRACK dataset for different settings where positive/negative refers to edge pairs whose nodes are in the same/distinct pedestrians.

In Table 6, we compare our full settings for temporal edge costs c_{e^t} in Equation 23 and compare with: (i) without using forward/backward prediction (second row) and (ii) without utilizing pre-clustering results to query images at other cameras (third row), i.e., we only measure appearance affinities for τ and τ' in the current camera. It can be observed that pre-clustering is the most critical factor; thereby, the total accuracy will decline from 99% down to 94% in the Acc metric. Secondly, the forward/backward affinities also contribute significantly to the final performance with a 2% improvement.

For spatial edges in Table 7 we examine the impact of two ingredients: (i) forward/backward prediction (second row) and (ii) without applying our novel prior-3D based on pre-clustering agreement affinity (third row). The obtained results confirm that prior-3D affinity is the most influential affinity that is able to boost edge costs accuracy up to 7%, yielding 100% accuracy for both positive and negative edges. Similarly as for spatial edges, the forward/backward affinities also contribute to complement our prediction with a growth of approximately 2%.

Method	Type	Precision %	Recall %	F1 %	Acc %
Full settings	positive	98	90	94	99
	negative	99	100	100	
W/o Forward, backward	positive	95	89	92	97
	negative	97	98	98	
W/o Pre-clustering	positive	94	88	91	94
	negative	95	94	95	

Table 6. Temporal edge costs performance with variant settings over 50565 samples in which 3678 edges are positive and the remaining of 46887 are negative.

Method	Type	Precision %	Recall %	F1 %	Acc %
Full settings	positive	100	100	100	100
	negative	100	100	100	
W/o Forward, backward	positive	97	99	98	98
	negative	98	99	98	
W/o Prior-3D	positive	92	94	93	93
	negative	94	93	94	

Table 7. Spatial edge costs performance with different configurations evaluated over 11026 samples with 3215 edges are positive and 7811 remaining are negative.

G.3. Performance of Splitting Tracklets in the Spatial-Temporal Tracking Graph

We now demonstrate that the pre-clustering is helpful during the construction of the spatial-temporal tracking graph. In particular, we experimented on the WILDTRACK dataset due to its severe occlusions with two current state of the art single-camera pedestrian trackers: CenterTrack [53]

Figure 8. Qualitative tracking results on WILDTRACK at three overlapping camera views (zoom in for better visualization). The superiority of multi-cameras can be seen in the two following scenarios. First, even an object 20 is fully occluded at frame 12 in Cam 5 (green arrow), our tracker could still track this target by establishing inter-camera correspondences in Cam 1 and 7 (red arrow). Second, at frame 22 in Cam 1, the ID-Switch error will occur between ID 141 and 204 (yellow and orange arrows) due to ambiguous appearance affinities; fortunately, we could avoid this error by harnessing corresponding detections for those objects in Cam 7, which are separable.

and Tracktor++ [1]. Given tracklets computed by these trackers, we go through each tracklet and assign "correct" label for a pair of two consecutive detections if they are of the same object and as an "ID-Switch" label otherwise. Table 8 shows our performance. The results indicate that we can detect and correct up to 97% of ID-Switches w.r.t. the F1 score for both CenterTrack and Tracktor++ over a total of 310 resp. 280 tracklets on the testing frames. Furthermore, we were also able to retain correct consecutive detection for both trackers (100% F1 score). Obtaining such high-quality tracklets not only allows us to reduce ID-Switch errors significantly but also allows for better affinity computations, resulting in an overall better final tracking result.

Method	Type	Precision %	Recall %	F1 %	#Samples
CenterTrack [53]	Correct	100	100	100	3268
	ID-Switch	96	98	97	310
Tracktor++ [1]	Correct	100	100	100	3256
	ID-Switch	97	96	97	280

Table 8. Our performance in reducing ID-switch errors and retaining correct detection pairs in the tracklet splitting step. The result is measured with two state-of-the-art single-camera trackers.

Method	IDF1 ↑	MOTA ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDs ↓
Deep MCD + ptrack [7]	64.3	52.4	83.4	10.7	1023	2239	711
Tracktor++ [1]	65.0	60.4	68.3	8.7	819	637	272
CenterTrack [53]	67.4	67.5	74.6	3.4	649	494	278
LMGP-single cam, detection	69.6	66.4	75.3	5.2	690	575	128
LMGP-single cam, tracklets	71.2	72.8	82.5	2.7	755	394	154
LMGP-multi-cam, tracklets	**98.2**	**97.1**	**97.6**	**1.3**	**71**	**7**	**12**

Table 9. Our tracking performance compared to state of the art single camera methods on WILDTRACK.

G.4. Single-Camera Benchmark

We provide experimental results on WILDTRACK (using the multi-camera setup and then projecting the result to a single camera, namely `LMGP-Multi-cam, tracklets`) compared to three modern single camera pedestrian tracking methods in Table 9. Furthermore, we also evaluate LMGP with a pure single-camera approach, including (i) using detections in the tracking graph (which reverts to the model of [40]) and without pre-clustering or other 3D-geometry based components (`LMGP-single cam, detection`); (ii) using CenterTrack tracklets instead of detections and our full model (`LMGP-single cam, tracklets`).

As we mentioned, even with the best methods, the single-camera approach is still insufficient under severe occlusions. Considering large improvements on all metrics, we argue that the multi-camera strategy convincingly im-

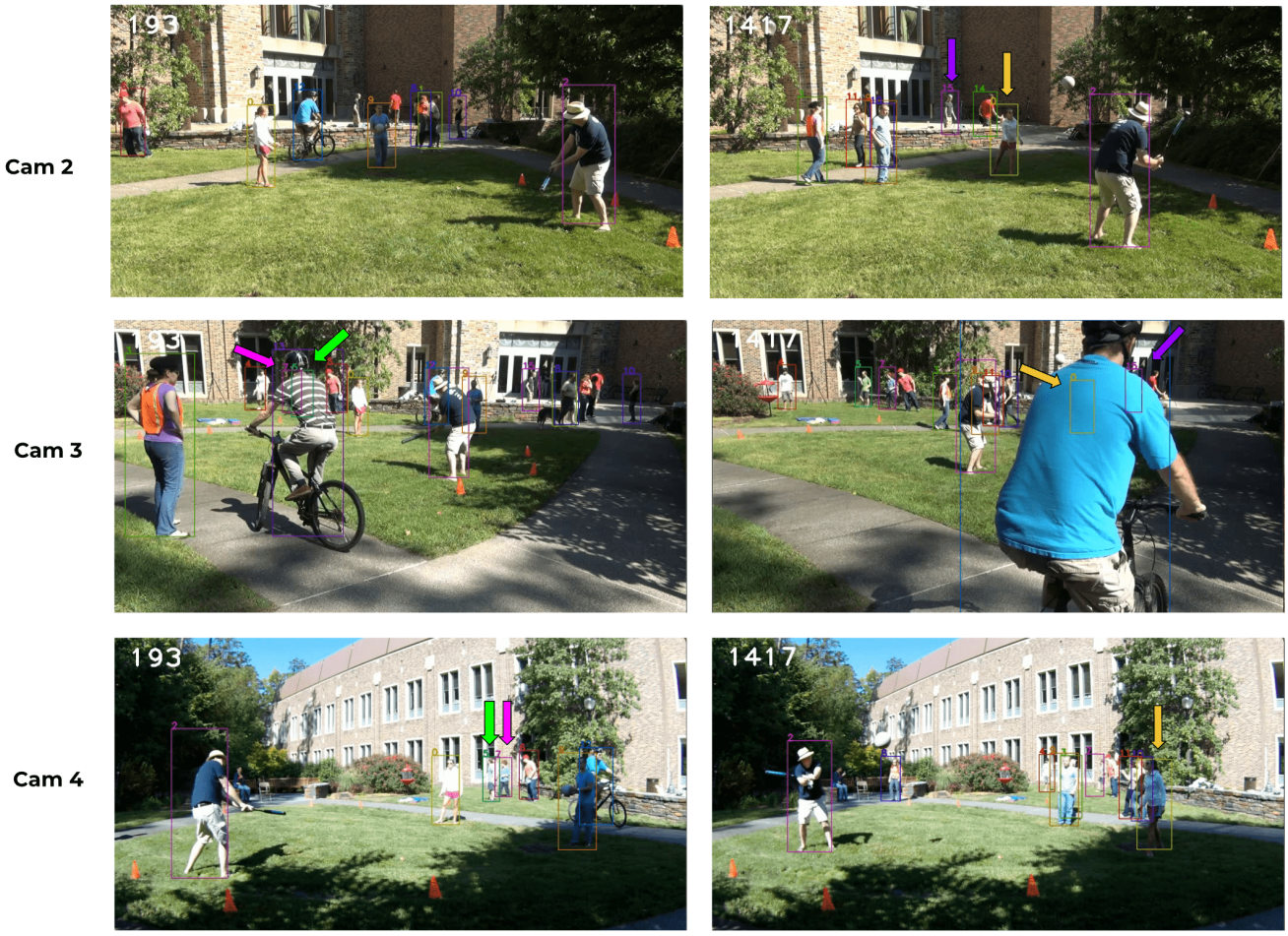

Figure 9. Qualitative tracking results in the Garden 1 in Campus dataset at three overlapping camera views (zoom in for better visualization). The benefit of the multi-camera approach can be found in two typical situations. First, at frame 193 (left column), two objects at the pink and green arrows in Cam 3 were not visible in this view; however, we could overcome these missing positions by using information from Cam 4. Similarly, at frame 1417, two objects at yellow and purple arrows are occluded in Cam 3. Fortunately, leveraging visible detections in Cam 4 and Cam 2, we can continue tracking these objects.

proves upon the single-camera one in crowded settings.

G.5. Running Time

Dataset	#Camera	#Frame/Cam	Running Time	Speed (fps)
WILDTRACK	7	400	213	1.9
PETS-09 S2-L1	3	795	147	5.4
PETS-09 S2-L2	3	436	135	3.2
PETS-09 S2-L3	3	240	111	2.2

Table 10. Running time and speed of the LMGP tracker on multi-camera sequences.

We show the running time of our tracker on two multi-camera datasets in Table 10, which includes both feature extraction and data association linking steps. We implemented our tracker in C++ on a Intel(R) Core(TM) i7-9800X CPU machine with 16 cores and 126 GB memory. Besides, four NVIDIA TITAN RTX GPUs with 24GB memory are employed for parallel computation in appearance feature extractions and running the pre-clustering. In short, we conclude that our tracker runs reasonably fast; therefore, it is feasible to deploy it in real-world tracking applications.

H. Qualitative Results

Our qualitative tracking results are illustrated in Figure 8 and Figure 9 for WILDTRACK and Campus, respectively, with three overlapping camera views. It is evident that monocular pedestrian tracking is insufficient to capture all objects due to highly crowded and cluttered scenes. For example, the target at the green arrow in Cam 5 at frame 12 in WILDTRACK is entirely occluded by other objects (Figure 8). In such situations, incorporating multiple views is a reasonable way to improve the total tracking performance.