Distributed Multi-Class Road User Tracking in Multi-Camera Network for Smart Traffic Applications

Nyan Bo Bo[1,2], Maarten Slembrouck[2,1], Peter Veelaert[2,1], and Wilfried Philips[2,1]

[1] imec, Kapeldreef 75, B-3001 Leuven, Belgium
[2] TELIN-IPI, Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium
Nyan.BoBo@ugent.be, Maarten.Slembrouck@ugent.be, Peter.Veelaert@ugent.be, Wilfried.Philips@ugent.be

Abstract. Reliable tracking of road users is one of the important tasks in smart traffic applications. In these applications, a network of cameras is often used to extend the coverage. However, efficient usage of information from cameras which observe the same road user from different view points is seldom explored. In this paper, we present a distributed multi-camera tracker which efficiently uses information from all cameras with overlapping views to accurately track various classes of road users. Our method is designed for deployment on smart camera networks so that most computer vision tasks are executed locally on smart cameras and only concise high-level information is sent to a fusion node for global joint tracking. We evaluate the performance of our tracker on a challenging real-world traffic dataset in an aspect of *Turn Movement Count* (TMC) application and achieves high accuracy of 93% and 83% on vehicles and cyclist respectively. Moreover, performance testing in anomaly detection shows that the proposed method provides reliable detection of abnormal vehicle and pedestrian trajectories.

Keywords: road user tracking · smart camera network · distributed computing · trajectory analysis · road traffic statistics · smart traffic.

1 Introduction

Automatic detection and tracking of road users, *i.e.*, vehicles, cyclists and pedestrians is an active research topic in computer vision since it is one of the essential building blocks in smart traffic and intelligent surveillance applications. Traffic features extracted from trajectories of road users are vital for understanding the behavior of road users, modeling of traffic, evaluation of traffic scenes and eventually automatic traffic flow control to obtain optimal efficiency. Many existing trackers use information from a single camera to track one or more classes of road users. To tackle coverage limitation of a single camera view tracking, a network of multiple cameras is used. A considerable number of methods have been proposed on tracking of road users across multiple cameras with overlapping/non-overlapping views. However, all these methods boil down to a single tracking

with camera handover or target re-identification to associate local trajectories of each camera view to produce global trajectories.

A relatively small amount of work, for instance [4, 15, 8, 7], has exploited the advantages of using information from cameras observing the same traffic scene with high overlapping view from different angles. By observing the same road user from different view angles, the occlusion problem may be significantly mitigated. Centralized multi-camera tracking methods [4, 15, 7] require images from all views in order to localize and track road users. In these methods, the number of cameras is limited by computational and communication bottlenecks. Fortunately, the introduction of smart cameras [12] allows the execution computer vision algorithms locally on cameras and only compact high-level information is sent to a decision node for joint trajectory estimation.

In this paper, we propose a distributed multi-camera road user tracking system which is capable of simultaneously tracking pedestrians, cyclists and vehicles with high accuracy. Each smart camera locally tracks road users on its image plane using recursive Bayesian estimation. Local 2D estimates are then sent to a fusion node on which ground plane positions are jointly estimated. For the performance evaluation of our proposed method, we captured a 90 minutes long multi-camera dataset of a real traffic scene at the intersection of five streets using four cameras. Instead of directly measuring the accuracy of the trajectories, which involves exhaustive manual annotation, we perform an experimental analysis on the performance of high-level tasks such as turning movement counting and anomaly detection. The experimental analysis shows that our method provides trajectories for turning movement count with high accuracy of 93% on vehicles and 83% on cyclists, and reliable anomaly detection.

The rest of this paper is organized as follows: Section 2 briefly discusses the related work on tracking of road users. Section 3 describes our proposed tracking method in details. Section 4 explains how do we experimentally measure the performance of the proposed tracker and presents our findings. Section 5 summarizes this paper.

2 Related Work

Numerous road user tracking methods have been proposed over the past two decades. Early methods use feature points [14], foreground blobs [16, 4] and histograms [5] in combination with various data association/filtering frameworks to track road users. With an increase in available computation power and the introduction of more efficient classifier algorithms, many methods [6, 17] utilize object detector responses as observations in trajectory estimation. This approach is called *tracking-by-detection* in visual tracking literature since an object of interest is first detected and then its trajectory is estimated by associating subsequent detector responses through time.

The popularity of the *tracking-by-detection* approach was boosted by the introduction of more accurate object detectors based on *Convolutional Neural Networks* (CNN). A vehicle tracker proposed by Qui *et al.*[10] deploys the *You Only*

Look Once YOLO object detector [11] and objects detected at two different time instances are associated by finding the optimal matches of *Kanade-Lucas-Tomasi* (KLT) feature points. Their work does not address partial/occlusion of vehicles which is very common in practical traffic surveillance. Ooi *et al.*[9] first detects road users with a *Region-based Fully Convolutional Network* (RFCN) [1]. Then data associating is performed by minimizing a cost function which incorporates object type (car, bicycle, pedestrian, etc.), position, size and color between objects detected at different time instances using the Hungarian algorithm. Missing detection due to occlusion, noise, etc. are handled by the prediction of a Kalman filter. However, if the detector fails to localize an object being tracked for an extended period of time, the prediction of the Kalman filter without measured evidences may drift from an actual trajectory.

Road users can be observed from different viewpoints using a network of cameras to tackle the occlusion problem since it is quite unlikely that a particular road user is occluded in all views. Tang [15] performs an inverse projection of foreground pixels onto the common plane and then deploys an overlap reasoning of the inverse projected blobs for joint position estimation. Instead of an inverse projection, the *Probabilistic Occupancy Mapping* (POM) approach computes the probability of positions on the ground plane being occupied by one or more road users. This is done by projecting a hypothesized volume of road users on the image plane and measuring how well it matches to observations from one or more cameras. The original work of Fluret *et al.*[2] uses foreground blobs as observations in POM computation. A recent work of Nishikawa [7] utilizes responses from the YOLO object detector as observations and *K-Shortest Path* (KSP) optimization to fit trajectories to the computed POM through time. This approach achieves its optimal performance when the KSP trajectory fitting is done on all frames in video, making it an offline tracker,*i.e.*, tracks after the whole video is captured. It is possible to deploy it as an online tracker with small delay by performing trajectory fitting in a batch of a few frames but the performance is then often suboptimal.

3 Proposed Method

Our distributed tracking system consists of K smart cameras observing a traffic scene from different angles and a fusion node as depicted in Figure 1. Internal parameters such as camera matrix and distortion coefficients as well as external parameters such as rotation matrix and translation vectors are obtained during the camera network installation. In visual tracking literature, an object being tracked is usually denoted as a target. Therefore, the word road user and target will be used interchangeably throughout this paper. Each smart camera locally tracks all targets in its view by using recursive Bayesian state estimation on its image plane and sends local position estimates to the fusion node. Local position estimates of the same target from different viewpoints are fused as a single joint position on the global ground plane. These joint position estimates are associated through time using recursive Bayesian state estimation on the

ground plane to produce optimal global trajectories. The following subsections describes both local tracking on smart cameras and global tracking on the fusion node in detail.

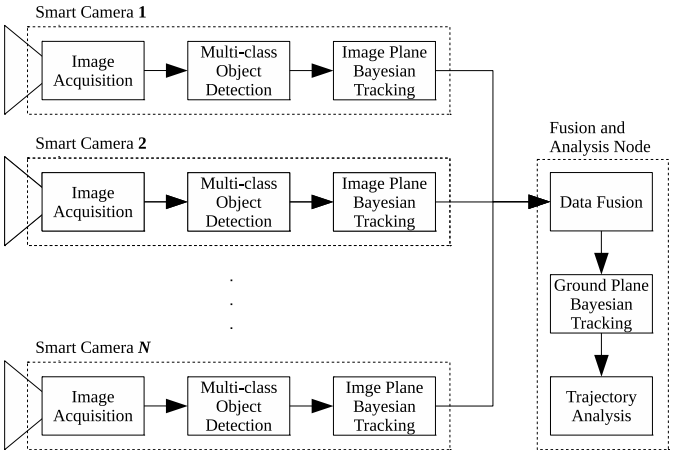


Fig. 1. Building blocks of our proposed tracker.

3.1 Tracking on Smart Camera

At a give time t, a smart camera captures an image observed in its view and feeds it to the YOLO object detector. Output of the object detector is a set of N detections $\boldsymbol{d}_i = (u_i, v_i, w_i, h_i, \alpha_i, \lambda_i) : i \in \{1, 2, \ldots, N\}$ on its image plane where (u_i, v_i), w_i, h_i, α_i and λ_i are the detection center, width, height, object class (pedestrian, cyclist and vehicle) and reliability score respectively. Our tracker keeps the state of M targets, each of which is represented with a vector. Given a set of current detections $D_t = \{\boldsymbol{d}_{1,t}, \boldsymbol{d}_{2,t}, \ldots, \boldsymbol{d}_{N,t}\}$ and a set of previously known states of all targets $K_{t-1} = \{\boldsymbol{k}_{1,t-1}, \boldsymbol{k}_{2,t-1}, \ldots, \boldsymbol{k}_{M,t-1}\}$, the task of a local tracker is to estimate a new state of each target as $K_t = \{\boldsymbol{k}_{1,t}, \boldsymbol{k}_{2,t}, \ldots, \boldsymbol{k}_{M,t}\}$. The state of each target is predicted and updated using recursive Bayesian estimation as:

$$\text{predict} : P(\boldsymbol{k}_t|\boldsymbol{d}_{1:t-1}) = \int P(\boldsymbol{k}_t|\boldsymbol{k}_{t-1})P(\boldsymbol{k}_{t-1}|\boldsymbol{d}_{t-1})d\boldsymbol{k}_{t-1} \tag{1}$$

and

$$\text{update} : P(\boldsymbol{k}_t|\boldsymbol{d}_{1:t}) = \frac{P(\boldsymbol{d}_t|\boldsymbol{k}_t)P(\boldsymbol{k}_t|\boldsymbol{d}_{1:t-1})}{P(\boldsymbol{d}_t|\boldsymbol{d}_{1:t-1})}. \tag{2}$$

Using Equation (1) together with a constant acceleration motion model, state of targets $\hat{K}_t = \{\hat{\boldsymbol{k}}_{1,t}, \hat{\boldsymbol{k}}_{2,t}, \ldots, \hat{\boldsymbol{k}}_{M,t}\}$ at time t can be predicted. These predicted

states have to be associated with corresponding detections in D_t so that Equation (2) can be used to update the predicted state using detections as observations. When observing a traffic scene, due to physical constraints, the variation in displacement and size of a target in terms of pixels in two consecutive frames are relatively small. Therefore, a detection $\boldsymbol{d}_t$ corresponding to a target j will be very close to the predicted $\hat{\boldsymbol{k}}_{j,t}$. Moreover, the object class α of the corresponding detection and target's state should be the same. Therefore, we define a function which calculates the association cost between a detection and a target as follows:

$$\delta(\boldsymbol{d}_i, \boldsymbol{k}_j) = \sqrt{(u_i - u_j)^2 + (v_i - v_j)^2} + |w_i - w_j| + |h_i - h_j| + \delta_{type}(\alpha_i, \alpha_j). \quad (3)$$

The first term in Equation (3) is simply the Euclidean distance while the second and third term compute the absolute differences in height and width respectively. The last term makes sure that $\delta(\boldsymbol{d}_i, \boldsymbol{k}_j)$ is very large when the detected class is different from target's class, formally expressed as:

$$\delta_{type}(\alpha_i, \alpha_j) = \begin{cases} 0 & \alpha_i \text{ and } \alpha_i \text{ are the same road user class} \\ \xi & \text{otherwise} \end{cases} \quad (4)$$

where ξ is a very large constant: $\xi \gg 1$.

The Hungarian method is one of the most popular methods to obtain the assignment of detections to targets with total minimal cost. However, it often occurs that the YOLO detector fails to detect one or more targets while new road users entering the scene are detected. This sometimes causes mismatch errors in the Hungarian method although total matching cost is at its minimum. Therefore, we propose to use a greedy matching algorithm as described in Algorithm 1. Only matched pairs with cost lower than a threshold Λ are added to the

Algorithm 1 Greedy matching.

Input: D_t: detections, $\hat{K}_t$: predictions
Output: *pairs*: matched detection and target pairs

Initialize *pairs* as a list
for $k \leftarrow 1$ to $minimum(N, M)$ **do**
  $\boldsymbol{d}, \boldsymbol{k} \leftarrow \underset{\boldsymbol{d} \in D_t, \boldsymbol{k} \in \hat{K}_t}{\arg\min} \ \delta(\boldsymbol{d}, \boldsymbol{k})$
  $D_t \leftarrow D_t \setminus \boldsymbol{d}, \ \hat{K}_t \leftarrow \hat{K}_t \setminus \boldsymbol{k}$
  if $\delta(\boldsymbol{d}, \boldsymbol{k}) < \Lambda$ **then**
    Append tuple $(\boldsymbol{d}, \boldsymbol{k})$ to the list *pairs*
  end if
end for

list *pairs*. Detections without matching target are initialized as new targets. For targets without any matching detection, a correlation-based template matching

method similar to Guan *et al.*[3] is used to generate a corresponding detection. Finally, the predicted states of all targets are updated using their corresponding detections. Under an assumption of the Gaussian distribution of noise in motion and observation models, aforementioned recursive Bayesian state estimation simply becomes a Kalman filtering. Center points on upper and lower edges of bounding boxes $\boldsymbol{r}^{upper} = (u^{upper}, v^{upper})$ and $\boldsymbol{r}^{lower} = (u^{lower}, v^{lower})$ of all targets are then sent to the fusion node.

3.2 Joint Position Estimation and Tracking

Consider a road user in the scene observed by a set of N smart cameras $C = \{c_1, c_2, \ldots, c_N\}$ from different angles. Only a subset of cameras $C_{vis} \subseteq C$ may be able to track the target. A target may be outside of the view of some cameras, occluded, or the detector may simply fails to detect it. In this case, only cameras in the subset C_{vis} are able to estimate the position of the targets in their own image coordinates. Suppose that a smart camera c accurately estimates the position of a target. A line connecting the two center points $\boldsymbol{r}_c^{upper}$ and $\boldsymbol{r}_c^{lower}$ must be the best approximation of the projection of a hypothetical vertical line, which length is the height h of the target, placed at a true physical position $(x, y, 0)$ of the target. In the ideal situation,

$$L(x, y, h) = |\boldsymbol{r}_c^{lower} - \rho_c(x, y, 0)|^2 + |\boldsymbol{r}_c^{upper} - \rho_c(x, y, h)|^2 \approx 0, \tag{5}$$

where a projection function $\rho_c(x, y, z)$ projects a point in 3D world coordinates on to the image coordinates of the camera c.

However, due to the presence of uncertainty in the detection and camera calibration, $L(x, y, h)$ will be usually not zero, but will only attain a minimum value greater than zero. When local estimates from a set of cameras C_{vis} are available, the error function $L(x, y, h)$ can be extended to the multi-camera case:

$$L(x, y, h) = \sum_{c \in C} (|\boldsymbol{r}_c^{lower} - \rho_c(x, y, 0)|^2 + |\boldsymbol{r}_c^{upper} - \rho_c(x, y, h)|^2). \tag{6}$$

The joint estimate of the target's position and height is then found by minimizing the error function $L(x, y, h)$ over all positions (x, y) and possible heights:

$$\hat{x}, \hat{y}, \hat{h} = \underset{x,y,h}{\arg\max}\ L(x, y, h). \tag{7}$$

Subsequently, using x and y as state variables, a discrete Bayes filter with constant velocity motion model is applied to suppress the noise in the join joint estimation.

4 Experimental Analysis

4.1 Traffic Dataset

For the performance evaluation of our method, we capture a multi-camera video using four *GoPro HERO 4* cameras aiming at an intersection of five streets in

the city of Ghent, Belgium. Camera positions and streets layout are depicted in Figure 2 where top view of the intersection is obtained from *Google Maps*. All streets allow two-way traffic except *Street 4* which does not allow incoming traffic from the junction. Videos are captured at 30 FPS with the HD Ready resolution of 1280×720 pixels. All four video streams are loosely time synchronized and the duration is approximately one and a half hour. All types of usual road users such as cars, trucks, bus, cyclists and pedestrians go trough the intersection causing partial/full occlusion in one or more camera views making it a very challenging dataset.

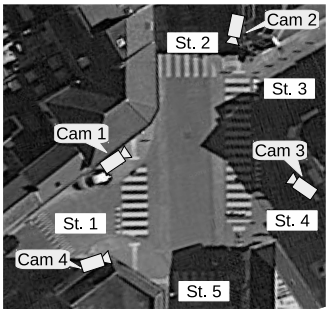


Fig. 2. Camera layout plotted on Google Maps.

4.2 Automatic Turning Movement Count of Vehicles and Cyclists

In smart traffic application, turning movement count (TCM) at intersection is essential information to understand the traffic flow for optimal traffic management. For example, more efficient signal timing plan can be derived from TMC information. Therefore, we measure the performance of our tracker in a TMC application for vehicles and cyclists. First, trajectories of road users denoted as $S = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_N\}$ are generated using the proposed tracker. Then we define regions at the beginning of each streets: Ω_1, Ω_2, Ω_3, Ω_4 and Ω_5 for *Street 1*, *2*, *3*, *4* and *5* respectively. If a trajectory passes through a region Ω_f first and a region Ω_l last, it is classified as a trajectory coming from *Street f* and going into *Street l*. This simple rule-based method produces the TMC as listed in Table 1.

Both tables in Table 1 clearly show that *Street 2* and *5* are the main streets as TMC between two streets for both vehicles and cyclists are much higher than others. Approximately 60% of traffic (both vehicles and cyclists) at the intersection goes trough *Street 2* and *5*. The second most used path is between *Street 1* and *2* constituting approximately 10% of the total traffic flow. The street with the lowest incoming vehicle traffic is *Street 3* since only three incoming vehicles are detected). However, there should be no incoming vehicle in *Street 4* since it is just a one-way street allowing only outgoing traffic. From visual

inspection of those trajectories, we learn that all three trajectories are wrong. In fact, two trajectories are false positive trajectories and one is the first part of segmented trajectories of a vehicle. The vehicle turns into the *Street 4* from *Street 5* but the driver realizes that *Street 4* is a one-way street and incoming traffic is not allowed. Therefore, the vehicle turns back and goes into the *Street 3*. This results in two isolated trajectories of the same target causing error in automatic TMC application.

		Destination				
		St. 1	**St. 2**	**St. 3**	**St. 4**	**St. 5**
Origin	**St. 1**	0	7	6	0	12
	St. 2	28	5	15	0	142
	St. 3	8	10	0	0	6
	St. 4	9	8	12	0	8
	St. 5	14	122	8	3	4

(a) Vehicle trajectories

		Destination				
		St. 1	**St. 2**	**St. 3**	**St. 4**	**St. 5**
Origin	**St. 1**	1	25	6	5	7
	St. 2	33	2	2	6	137
	St. 3	9	2	0	2	9
	St. 4	4	24	2	1	5
	St. 5	4	117	10	7	6

(b) Cyclist trajectories

Table 1. Results of automatic turning movement count at five arms intersection.

To obtain numerical results of TMC's accuracy, we randomly selected 100 trajectories and visually inspected in the videos if they are correct. We achieve an accuracy as high as 93% on vehicles and 83% on cyclists. The key of achieving high accuracy in turning movement count is to be able to track targets while avoiding tacking loss (resulting segmented trajectories instead of a complete trajectory for a target) and identity switches. Figure 3 illustrates the example of typical trajectories produced by our tracker. Two cyclists are fully occluded by a white van in the view of *Camera 1* and *2*. However, they are visible in the view of *Camera 3* and *4* (partially occluding each other). Although there is no local estimate available from *Camera 1* and *2*, our tracker fuses local estimates from *Camera 3* and *4* to produce accurate joint estimates and tracks two cyclists without any tracking lost.

4.3 Anomaly Detection of Road Users

Road users usually follow similar paths as they move about in traffic. For example, the majority of pedestrians walk on the road side pavement and cross the road along a pedestrian crossing. A trajectory which is very different from the common trajectories is regarded as an abnormal trajectory. For instance, a pedestrian may cross the road without using a pedestrian crossing. If trajectories are clustered based on their similarity, common trajectories form clusters which contain the majority of the trajectories while abnormal trajectories form clusters containing only a few trajectories. In this subsection, we assess the reliability of our tracker in trajectory anomaly detection application. For this purpose, we

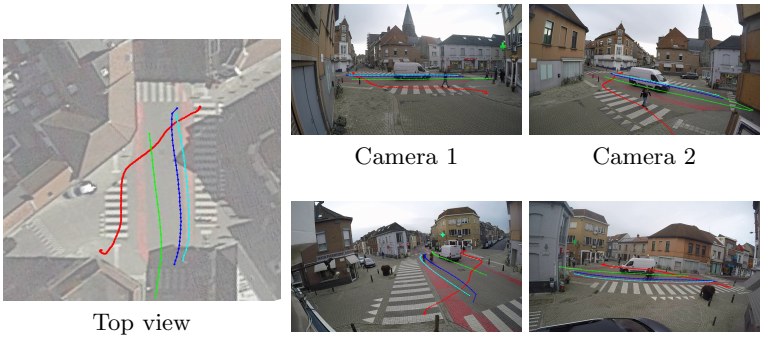


Fig. 3. Example tracking results of a pedestrian (red trajectory), two cyclists (blue and cyan trajectories) and a van (green trajectory).

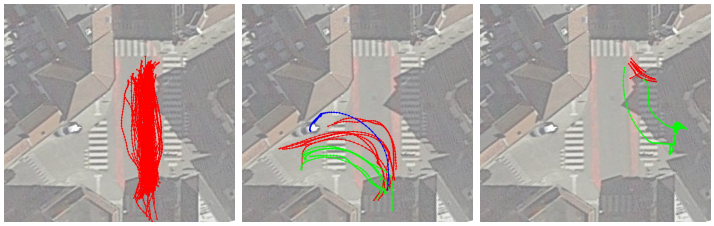
propose a greed clustering algorithm as described in Algorithm 2. The algorithm uses the Euclidean distance between point pairs from r_j and r_k defined by Dynamic Time Warping algorithm [13] as a dissimilarity measure, which is denoted as $\delta_{DWT}(r_j, r_k)$ in Algorithm 2. The operator $|.|$ computes the cardinality of a set.

Algorithm 2 Greedy clustering.

Input: $S = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_N\}$: a set of trajectories
Output: $S_1, S_2, \ldots, S_M$: sets of trajectories
$i \leftarrow 1,\ S_i \leftarrow \emptyset$
while $S \neq \emptyset$ **do**
    $S_i' \leftarrow S_i$
    for each $\boldsymbol{r_j} \in S$ **do**
        if $\frac{\sum_{r_k \in S_i} \delta_{DWT}(r_j, r_k)}{|S_i|} < \Upsilon$ **then**
            $S_i \leftarrow S_i \cup \{r_j\},\ S \leftarrow S \setminus \{r_j\}$
        end if
    end for
    if $S_i' = S_i$ **then**
        $i \leftarrow i + 1,\ S_i \leftarrow \emptyset$
    end if
end while

All trajectories from *Street 5* to *2* are grouped into a single cluster and no possible anomaly is detected as shown in Figure 4a. For trajectories coming from *Street 5* and going into *Street 1*, two usual clusters are formed and one anomaly is detected. When there is no vehicle/cyclist waiting at the mouth of *Street 1* to go into other streets, vehicles coming from *Street 5* tend to make smaller turns to go into *Street 1*. These trajectories form a cluster of common trajectories which are

shown as green trajectories in Figure 4b. Vehicles make bigger turns when there are vehicles/cyclists waiting at the mouth of *Street 1* resulting in another cluster of common trajectories shown as red trajectories in 4b. The abnormal trajectory shown as blue trajectory in 4b is the trajectories of a car which drives onto a pavement as it turns into *Street 1* and parks for a while before continue driving down the same street. Figure 4c depicts the detected abnormal trajectory which is a results of a car coming out of *Street 3*, which turns into *Street 4* and parks at the mouth of the street for about five minutes. Then the car drives out of *Street 4* and goes into *Street 2*. The other trajectories shown in red are usual trajectories from *Street 3* to *2*.



(a) (b) (c)

Fig. 4. Examples of anomaly detection in vehicle trajectories.

Pedestrian trajectories are also clustered using Algorithm 2 and examples of the resulting clusters are shown in Figure 5. As expected, common trajectories are clustered as big clusters which usually are along pedestrian crossings (as shown in Figure 5a and 5b) and pavements. Three similar abnormal trajectories are formed when pedestrians cross the road from the corner of *Street 1* and *5* to the corner of *Street 2* and *3*, as shown in Figure 5c. A pedestrian walks to the middle of the junction and comes back while crossing *Street 1* using the pedestrian crossing. This results in a trajectory which is quite different from the other common trajectories as shown in Figure 5d. A trajectory shown in Figure 5e is formed by a person getting out of a parked car (the same car which causes the abnormal vehicle trajectory shown in Figure 4b) and walked straight to a shop at the corner of *Street 4* and *5*. A trajectory of the same person returning from the shop to the car is also detected as abnormal trajectory in another cluster as depicted in Figure 5f.

5 Conclusion

This paper presented the multi-camera tracking method which simultaneously tracks multiple classes of road users such as pedestrians, cyclists and vehicles. It provides reliable trajectories for subsequent smart traffic applications such as turning movement count and anomaly detection. The distributed design of

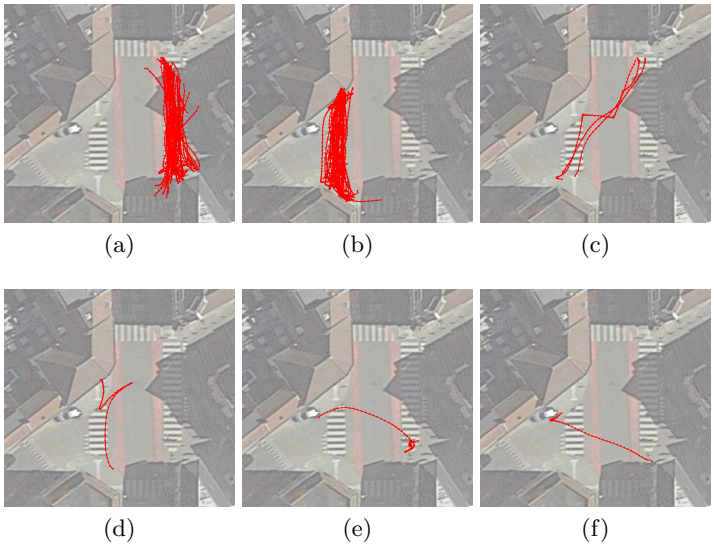


Fig. 5. Examples of anomaly detection in pedestrian trajectories.

the proposed tracker allows the deployment on smart camera networks, keeping all local computer vision tasks on smart cameras. Only concise high-level information is exchanged for joint position estimation. Each smart camera locally estimates the image plane position of targets by recursive Bayesian estimation using YOLO detector responses and template matching as observations. Locally estimated image positions of targets are then transmitted to the fusion node where corresponding ground plane positions are jointly estimated by minimizing proposed cost function. Performance of the proposed method was assessed in context of the turning movement count application and achieved an accuracy as high as 93% on vehicles and 83% on cyclists. Abnormal itineraries of vehicles and pedestrians were also detected with high reliability by clustering trajectories produced by the proposed tracker.

6 Acknowledgement

This research received funding from the Flemish Government under the "*Onderzoeksprogramma Artificile Intelligentie (AI) Vlaanderen*" programme.

References

1. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: Object detection via region-based fully convolutional networks. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. pp. 379–387. NIPS'16, Curran Associates Inc., USA (2016)

2. Fleuret, F., Lengagne, R., Fua, P.: Fixed point probability field for complex occlusion handling. In: Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1. vol. 1, pp. 694–700 Vol. 1 (Oct 2005)
3. Guan, J., Van Hese, P., Niño Castañeda, J., Nyan, B.B., Grünwedel, S., Van Haerenborgh, D., Van Cauwelaert, D., Veelaert, P., Philips, W.: Template matching based people tracking using a smart camera network. In: Proceedings of SPIE. vol. 9026, pp. 1–9. SPIE-INT SOC OPTICAL ENGINEERING (2014)
4. Hu, Z., Wang, C., Uchimura, K.: 3d vehicle extraction and tracking from multiple viewpoints for traffic monitoring by using probability fusion map. In: 2007 IEEE Intelligent Transportation Systems Conference. pp. 30–35 (Sep 2007)
5. Lee, S., Baik, H.: Origin-destination (o-d) trip table estimation using traffic movement counts from vehicle tracking system at intersection. In: IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics. pp. 3332–3337 (Nov 2006)
6. Liu, L., Xing, J., Ai, H.: Multi-view vehicle detection and tracking in crossroads. In: The First Asian Conference on Pattern Recognition. pp. 608–612 (Nov 2011)
7. Nishikawa, Y., Sato, H., Ozawa, J.: Multiple sports player tracking system based on graph optimization using low-cost cameras. In: 2018 IEEE International Conference on Consumer Electronics (ICCE). pp. 1–4 (Jan 2018)
8. Nyan, B.B., Veelaert, P., Philips, W.: Occlusion robust symbol level fusion for multiple people tracking. In: Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017), VOL 6. vol. 6, pp. 216–226. SCITEPRESS Science and Technology Publications, Lda. (2017)
9. Ooi, H.L., Bilodeau, G.A., Saunier, N., Beaupré, D.A.: Multiple object tracking in urban traffic scenes with a multi-class object detector. In: Advances in Visual Computing. pp. 727–736. Springer International Publishing, Cham (2018)
10. Qiu, H., Liu, X., Rallapalli, S., Bency, A.J., Chan, K., Urgaonkar, R., Manjunath, B.S., Govindan, R.: Kestrel: Video analytics for augmented multi-camera vehicle tracking. In: 2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI). pp. 48–59 (April 2018)
11. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 779–788 (June 2016)
12. Rinner, B., Wolf, W.: An introduction to distributed smart cameras. Proceedings of the IEEE **96**(10), 1565–1575 (Oct 2008)
13. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing **26**(1), 43–49 (February 1978)
14. Saunier, N., Sayed, T.: A feature-based tracking algorithm for vehicles in intersections. In: Proceedings of the The 3rd Canadian Conference on Computer and Robot Vision. pp. 59–. CRV '06, IEEE Computer Society, Washington, DC, USA (2006)
15. Tang, H.: Development of a multiple-camera tracking system for accurate traffic performance measurements at intersections. Tech. rep., Intelligent Transportation Systems institute, Center for Transaction Studies, University of Minnesota
16. Weiming Hu, Xuejuan Xiao, Zhouyu Fu, Xie, D., Tieniu Tan, Maybank, S.: A system for learning statistical motion patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(9), 1450–1464 (Sep 2006)
17. Zhang, H., Geiger, A., Urtasun, R.: Understanding high-level semantics by modeling traffic patterns. In: 2013 IEEE International Conference on Computer Vision. pp. 3056–3063 (Dec 2013)