A Multi-View Pedestrian Tracking Framework Based on Graph Matching

Fanyi Duanmu[1], Xin Feng[2], Xiaoqing Zhu[3], Wai-tian Tan[3], and Yao Wang[1]
[1]New York University, [2]Chongqing University of Technology, [3]Cisco Systems Inc.
{fanyi.duanmu, yw523}@nyu.edu; xfeng@cqut.edu.cn; {xiaoqzhu,dtan2}@cisco.com.

Abstract

In the applications of video monitoring over large public or private spaces, multiple cameras are required to cover the entire space and resolve the problems of occlusion, object intersection and so on. In this work, a novel multi-view pedestrian tracking framework is proposed to simultaneously detect and associate human objects across views using graph matching techniques to fully exploit the object features and the spatial/temporal relationships among the objects. Experimental results are provided to demonstrate the accuracy of our proposed framework.

Keywords: Pedestrian Detection, Graph Matching, Multi-view Tracking, Object Association

1. Introduction

Multi-view pedestrian tracking is a very popular and useful computer vision research topic with a broad impact on applications such as intelligent surveillance, human machine interaction, etc. There are several technical challenges, including the occlusion, trajectory intersection, the detection failures due to illumination change or object size, the cross-camera correspondence creation (especially without prior camera or scene knowledge). To address these topics, there have been a large number of tracking algorithms proposed in the past two decades to tackle such challenges from both detection and tracking aspects.

From the detection perspective, numerous approaches have been proposed and can be categorized as follows:

Category 1: Holistic Detection. Detectors are trained based on global features such as edge template [1] or local features such as histogram of oriented gradients (HOG) [2]. Whenever the features in local search window meet certain criteria, the detector will catch it and categorize this search window as a detected human. For instance, in OpenCV [3] package, a pre-trained HOG-based linear support vector machine (SVM) [4] is used for detection. However, such algorithms are sensitive to occlusions.

Category 2: Part-based Detection. Human objects can be modeled as a collection of parts (as in [5]). Edge and orientation features can be derived over each part. The joint hypotheses from the parts are used to classify the assembled object block. The algorithms in this category often have better resistance to occlusions. However, the part processing and detection can be very complex (e.g., computing features per scale, performing classification per possible location and post-processing, such as non-maximal suppression, etc.).

Category 3: Motion-based Detection. In such kind of algorithms, the moving pedestrians are usually modeled as foreground while the static regions are classified as background, through background-subtraction algorithms (e.g., [6]). However, the algorithms from this category are intrinsically sensitive to the illumination change and background noise.

Category 4: Deep learning-based Detection. In the past a few years, advanced deep learning techniques are widely used in object recognition areas. Basically, a large image dataset is prepared beforehand and used for training the classifier(s) from pixel domain. The resultant classifier is able to derive the bounding area of the target object and decide the class of this object (e.g., car, human, etc.) For example, Region-based Convolutional Neural Network (RCNN) proposed in [7] can mostly detect objects from an input image and classify them into the corresponding categories with very high accuracy.

From the tracking point of view, some classical approaches such as Meanshift [8], Kalman Filtering [9] perform well for single-view tracking but may fail in complex scenarios with frequent object disappearances and occlusions. For a long-term tracking, the most typical framework is tracking-by-detection. In such scheme, a human detector is firstly utilized to localize the object in each frame. Then a tracker algorithm is used to find correspondence among objects and associate the objects across frames (e.g., [10] [11]). For the multi-camera

pedestrian tracking, two major schemes are proposed and referred as "fusion-first" and "tracking-first". In "fusion-first" framework (such as [12]), the detected objects are projected to a common view and associated. Then the tracking is triggered over each view, whereas in "tracking-first" framework, tracking is completed either independently over each view or collaboratively across views, then the estimated tracks from each view can be combined and associated (such as [13]). To associate the objects across views, different approaches are proposed. Some solutions are purely based on image features (e.g., color histograms, HOG, etc). For instance, in [14], Kang et al. formulate the tracking as a problem of maximum joint probability model based purely on color. It uses the probabilistic data filtering and multi-camera homography to track pedestrians across cameras. In [15], Fleuret et al. proposes to predict people occlusions using a generative model and corresponding probabilistic occupancy map, and demonstrates successful tracking of up to 6 people. In [16], Berclaz et al. proposes a k-shortest paths (KSP) algorithm and formulates the trajectory linking and data association as a global network flow convex optimization problem and significantly reduces the tracking complexity.

Different from the traditional people re-identification problem, for cross-view pedestrian tracking, relationships among objects are consistent over 3D world coordinate and serve as strong constraints for object association. In recent years, several graph-based solutions have been proposed to associate objects not only based on object features but also relationships. In [17], tracking graph is firstly created over each view and later a separate cross-view graph is created to impose the spatial constraints for the corresponding view pair. Li et al. improve upon [17] and propose to introduce "virtual nodes" in [18] from each view onto another to ensure the amount of detections in each view pair are the same. Further, the graph weights are refined using world coordinate distances. However, in this work, only graph node similarity is considered without fully exploiting the edge similarity. Nie, et al. propose a unified graph matching framework [19] for single and cross camera pedestrian tracking. In this work, graph matching is applied to associate object "tracklets" using both person-wise and part-wise features. The two graphs before and after occlusion and scene transition are matched to link "tracklets".

Inspired by the previous works, in this paper, we propose to combine the state-of-art deep convolutional neural network (CNN) based object detection and graph matching (GM) based object association within the same framework. Each camera is used as a smart sensor and processor to identify and track pedestrians. The post-processed data (e.g., trajectory, object descriptors, etc.) after compression are up-streamed to an edge server for data synthesis and final cross-view object association. During the graph derivation, we fully exploit the object-level features and spatial/temporal features as node and edge attributes in the graph. The experimental results demonstrate that the proposed framework can outperform recent work in terms of tracking accuracy and precision.

The rest of this paper is structured as follows. Section 2 provides a brief background review of graph matching algorithm, a key component of our system. Section 3 details our system design, including pedestrian detection, single-view pedestrian tracking and cross-view pedestrian association. Section 4 presents the experimental results. This paper concludes in Section 5 with tentative future work summarized.

2. Background Review of Graph Matching

In this work, the state-of-art factorized graph matching (FGM) [20] algorithm is used for both the single-view pedestrian tracking and the cross-view object association, as detailed in Section 3.

Similar to conventional graph matching (GM), given two graph structures and the associated affinity matrices, the problem of FGM also aims at deciding the optimal correspondence X between nodes, to ultimately maximize the global sum of the node and edge compatibilities, as shown in (1), where $KP(n_i, n_j)$ denotes the similarity between node i from graph A and node j from graph B, $KQ(e_{ik}, e_{jl})$ measures the similarity between edge e_{ik} linking node i and k in graph A and edge e_{jl} linking node j and l in graph B.

$$\mathrm{J(X)} = \sum_{i,j} x_{i,j} KP(n_i, n_j) + \sum_{e_{ik} \in E_1, e_{jl} \in E_2} x_{ij}\, x_{kl} KQ(e_{ik}, e_{jl}) \quad (1)$$

In FGM, the pair-wise affinity matrix K is factorized, to provide light-weight and unified representation of conventional GM methods, as shown in (2), where KP and KQ are node and edge affinity matrices, G_1 and H_1 represent the incidence matrices of graph A and G_2 and H_2 represent the incidence matrices of graph B. Namely, the non-zero elements in each column of G_1/G_2 and H_1/H_2 indicate the starting and ending nodes in the corresponding directed edges, respectively.

$$\mathrm{K} = diag\big(vec(KP)\big) + (G_2 \times G_1)\, diag\big(vec(KQ)\big)(H_2 \times H_1)^T \quad (2)$$

FGM has the following advantages, compared with previous GM algorithms:

Firstly, FGM operates over smaller matrices. Therefore, the computational cost is significantly reduced.

Secondly, FGM internally introduces a path-following algorithm and convex relaxation to better approximate the GM problem solution.

Experimental results have demonstrated that FGM outperforms the previous GM frameworks with proven performance margin and is therefore chosen as the object association tool for our pedestrian association task.

3. System Overview

Our proposed multiview tracking framework consisits of three key components: Pedestrian Detection (in Section 3.A), Single-view Tracking (in Section 3.B) and Cross-view Tracking (in Section 3.C).

A. *Pedestrian Detection*

Our pedestrian detection is based on RCNN detector [7] as introduced in Section 1. Among 20 pre-defined training categories, we only enable the "person" category (with a confidence threshold=80%) and use this classifier to detect pedestrian objects and corresponding bounding boxes. The RCNN detector is very robust to illumination change and the output at the fully-connected layer can be used as feature vectors (totally 4096 entries) for the following pedestrian tracking and cross-view object association tasks.

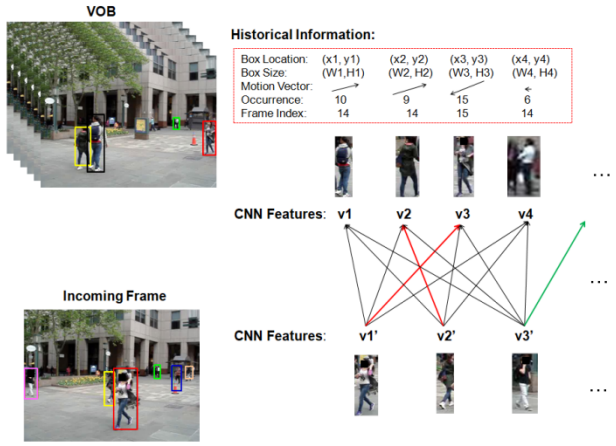

Figure 1. Single-View Pedestrian Tracking Algorithm
Red Arrow: VOB match found and object unified. **Green Arrow**: New VOB entry to create. Pedestrian faces are masked to protect privacy.

B. *Single-View Object Tracking*

Our single-view pedestrian tracking framework is illustrated in Figure 1. In this framework, a virtual object buffer (VOB), namely a dynamic pedestrian lookup table, is used to document the historical information of each detected object, such as CNN features, bounding box, detected frame indices, motion vector, object label, last detected frame index, occurrence (i.e., number of total detections), etc. Each new pedestrian object from the incoming frame will be compared against each VOB candidate using the RCNN features and be associated with the one which satisfies the following two criteria:

Criterion 1: The bounding box intersection-over-union (IOU) ratio between the incoming pedestrian object and the motion-compensated VOB object is greater than a pre-defined threshold (i.e., 0.3). Please note that the motion-compensated object is the previous detected VOB object translated by $(mv \cdot \Delta f)$, where mv is the most recent VOB object motion vector and Δf is the time distance between the current frame and the most recent frame the target VOB candidate appeared.

Criterion 2: The RCNN feature Euclidean distance between the current object and the VOB candidate is minimal among all VOB candidates and at the same time this VOB object has not been associated with any other incoming frame object previously.

If a match is found, then the incoming frame object is associated with the corresponding VOB object and this object entry is updated (e.g. occurrence, motion vector, trajectory, etc.). Otherwise, the incoming object is defined as a new object and a new entry will be created and inserted into the VOB.

This framework can tackle the infrequent RCNN detection failure (i.e., object is missing temporarily and later re-appears within a small number of frames, e.g., 1~2 frames), as shown in Figure 2 (top).

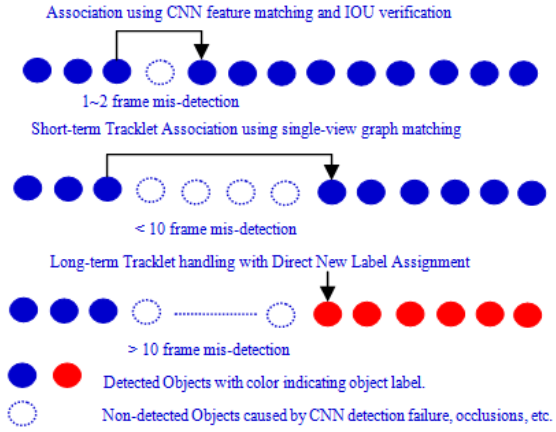

Figure 2. Single-View Tracking Mis-detection Handling strategies
Top: RCNN mis-detection (1~2 frames) solved by VOB lookup. ***Mid***: Short-term mis-detection (<10 frames) solved by single-view graph matching between last detected object frame and first re-detected object frame. ***Bottom***: Long-term mis-detection (>10 frames) handling using new label assignment.

On the other hand, the mis-detections caused by trajectory intersection or occlusion can be either short-term or long-term. If the occlusion lasts only a few frames (e.g., within 5 frames), after reappearance, the object still resembles the corresponding one from the last detected frame and the relationships to other objects remain. However, if the occlusion or intersection lasts longer (e.g., above 20 frames), we cannot confidently predict its location. In this work, we provide two different strategies to address the long-term and the short-term intersection and occlusion.

For the long-term "tracklets", whose frame-stamps are non-overlapping and the frame difference between the last detection and first re-detection is above 10 frames, we simply assign a new object label, as shown in Figure 2

(bottom). Later, we will refine the labels through cross-view tracking, as described in Section 3.C.

Over the short-term "tracklets", whose frame-stamps are non-overlapping and the frame distance between the last detection and the first re-detection is within 10 frames, we apply the FGM [20] algorithm (introduced in Section 2) to associate, as shown in Figure 2 (middle). In our single-view object association across frames, each detected pedestrian object is defined as a node in the graph, as illustrated in Figure 3. The relationship between two objects in the frame is defined as an edge in the graph. For object tracklets with non-overlapping frame-stamps (e.g., one tracklet ending in frame 11 and the other tracklet starting in frame 17), we trigger the graph matching over the two most adjacent frames (i.e., frame 11 and frame 17). For the associated objects, we further compare the spatial offset (i.e., displacement between associated objects bounding box centers) of the two associated objects against a threshold, i.e., $\text{s\_factor} \cdot mv \cdot \Delta f$, where mv is the motion vector of the ending frame object in the anchor object trajectory and Δf is the frame distance between the ending frame-stamp in the anchor tracklet and the starting frame-stamp of the other tracklet. s_factor is a scaling controller. A larger s_factor makes it more likely to link "broken" tracklets but simultaneously increases the risk of false-linking. In our configuration, s_factor=5. If two objects satisfy the spatial validation, we will link the corresponding tracklets and unify their labels. Otherwise, the two objects are assigned with different labels.

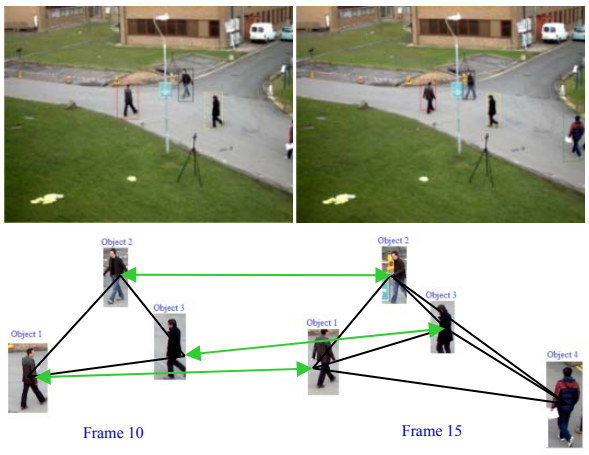

Figure 3. Single-View Graph Matching between frames
Solid black lines indicate the edge simlarity in a frame; ***Green lines*** indicate the associated objects through graph matching.

Under our configuration, the RCNN features are used as node attributes and the spatial relationships are used as edge attributes. The node similarity is defined in (3), where $\overrightarrow{CNN}_t^i$ and $\overrightarrow{CNN}_{t+\Delta t}^j$ denote the RCNN feature vector for node i and node j at time t and $t+\Delta t$. The edge similarity is defined in (4), where $\overrightarrow{d_{ik}}(t) = \overrightarrow{X_i}(t) - \overrightarrow{X_k}(t)$ is the displacement vector between node i position and node k position, at timestamp t and $\overrightarrow{d_{jl}}(t+\Delta t)$ is the displacement vector between node j position and node l position at timestamp $t+\Delta t$. In both (3) and (4), "$dist$" denotes the L-2 distance between two vectors.

$$KP(n_i, n_j) = exp(-dist(\overrightarrow{CNN}_t^i, \overrightarrow{CNN}_{t+\Delta t}^j)) \quad (3)$$

$$KQ(e_{ik}, e_{jl}) = exp(-dist\left(\overrightarrow{d_{ik}}(t), \overrightarrow{d_{jl}}(t+\Delta t)\right)) \quad (4)$$

C. *Cross-View Object Tracking*

The cross-view object tracking consists of three major steps: homography projection, graph matching, and label unification.

Step 1: Homography Projection. The goal of this step is to project objects standing on the same ground plane from different views onto a reference common plane to impose spatial constraints. In our implementation, we select the view with the largest visible area as the reference view and derive the homography matrix based on the pre-selected corresponding feature points on the ground between the reference and non-reference views. Without precise calibration parameters, the estimated homography parameters already provide relatively accurate projections for ground points (e.g., the points close to the feet are projected accurately after homography projection), as illustrated in Figure 4, where the original ground positions are transformed according to the derived homography matrix before further applying graph matching.

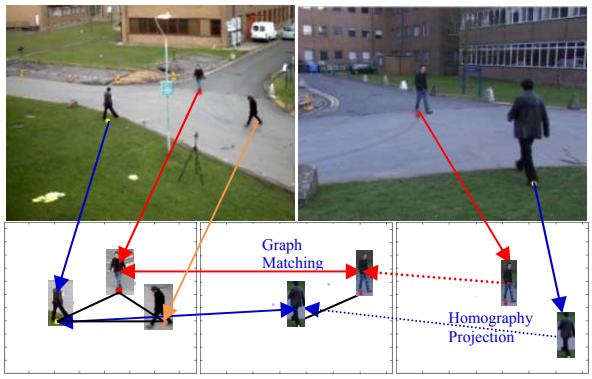

Figure 4. Cross-View Homography Mapping and GM Illustration
Top: two frames captured at the same time but from different cameras. The left view is used as reference view. **Bottom left and right**: original spatial locations in left and right views. **Bottom middle:** homography projected spatial locations from top right view to top left view. **Dashed lines** denote spatial homography transformation. **Bi-directional headed arrow** denote associated objects through cross-view graph matching.

Step 2: Cross-view Graph matching (CV-GM) between non-reference view and reference view. Similar to single-view object association between frames, we also apply the graph matching algorithm [20] to associate objects across views, using object-level features and inter-object relationships, as illustrated in Figure 4. For each view, each node represents a tracklet. The node attributes are RCNN features from the frames that the object is detected and the object trajectories. The node similarity between object i in one view and object j in another view is

$KP(n_i, n_j) = KP(CNN_{ij}) \cdot KP(S_{ij})$, where $KP(CNN_{ij})$ denotes the tracklet CNN feature similarity and $KP(S_{ij})$ represents the object trajectory similarity, as defined in (5) and (6). Here O_i and O_j represent the set of detection frame-stamps for object i and j. In (6), (x_t^i, y_t^i) denotes the 2D coordinates of the bounding box lower boundary middle point at time t for object i. N_t is the co-detected object number. The $T(.)$ represents the homography transform from object j view to object i view. In both (5) and (6), distance is measured using Euclidean norm.

$$KP(CNN_{ij}) = exp(-\frac{1}{N_t}\sum_{t\in(O_i \cap O_j)} dist(\overrightarrow{CNN_t^i}, \overrightarrow{CNN_t^j})) \quad (5)$$

$$KP(S_{ij}) = exp(-\frac{1}{N_t}\sum_{t\in(O_i \cap O_j)} dist(x_t^i, T(x_t^j)) \quad (6)$$

The edge similarity, $KQ(\overrightarrow{e_{ik}}, \overrightarrow{e_{jl}})$, is used to measure the object location and trajectory similarity between node i and node k in view 1 and that between node j and node l in view 2, as defined in (7), where cdf is the co-detected frame set between two views, t is the frame index, $\overrightarrow{d_{ik}}(t)$ and $\overrightarrow{d_{jl}}(t)$ are defined similarly as (4) but extended to 3-dimensional vector by concatenating all temporally-detected samples.

$$KQ(\overrightarrow{e_{ik}}, \overrightarrow{e_{jl}}) = exp(-\frac{1}{N_t}\sum_{t\in cdf} dist(\overrightarrow{d_{ik}}(t), \overrightarrow{d_{jl}}(t))) \quad (7)$$

For simplicity, the graph matching is triggered when a new object appears in one view (either entering the scene or being re-detected). After association, we re-calculate the matching distance D_t (i.e., sum of the node and edge distance against the associated nodes and edges in other views) for the new object and compare with the minimum distance (D_{min}) from the previously-associated objects. If $D_t \leq \rho \cdot D_{min}$, we assume the association is correct. Otherwise, we assume that the association is incorrect and assign a different label. Here ρ is a confidence parameter. A larger ρ will associate objects more easily but increases the risk of cross-view mismatch. In our setting, ρ = 10.

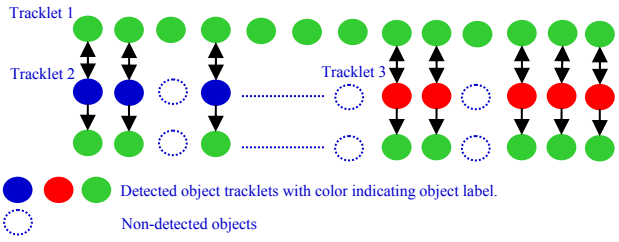

Figure 5. Cross-View Graph Matching Illustration
Top: Tracklet 1 captured from View 1; ***Mid***: Two tracklets (2 and 3) captured from View 2. ***Bottom***: After cross-view graph matching, tracket 2 and 3 are both matched with tracklet 1 for the overlapping frames and therefore the labels of tracklet 1, 2 and 3 are unified (eventually all marked in green).

Step 3: Label Unification. According to the cross-view graph matching results, we unify the labels, as illustrated in Figure 5. The object with shorter tracklet in one view will carry over the label from the associated object with longer tracklet in the other views, as long as it does not overlap with any previously-associated tracklet(s) in the same view. Automatically, the falsely-labeled tracklets in the single-view tracking are recognized and re-labeled.

4. Experimental Results

We evaluate our proposed tracking framework on the public PETS benchmark dataset [21]. The scenes in this cross-view dataset are very challenging with multiple pedestrians constantly interacting with each other and different kinds of occlusions. We simulate the multi-view tracking among *View* 1, 5 and 7 for 200 frames, using *View* 1 as the reference view. Objects from *View* 5 and *View* 7 are projected onto *View* 1 and associated and then the labels are unified across views.

We report our tracking results using the MOT metrics [22], i.e., multiple object tracking precision (MOTP) and multiple object tracking accuracy (MOTA), summarized in Table I. Previous works using the same dataset ([18], [23]) are provided for comparison.

TABLE I CROSS-VIEW TRACKING RESULTS OVER *PETS* DATASET

Method	[23] (3-view)	[23] (2-view)	[18] (2-view)	Proposed (3-view)
MOTP	**53.4%**	**60.0**	**79.9%**	**83.6%**
MOTA	**74.1%**	**76.0**	**93.2%**	**94.8%**

The experimental results demonstrate that the RCNN detector can mostly detect the pedestrian objects but may occasionally fail on partially-occluded or small objects. The RCNN features do not suffice to associate objects across views. However, after combining the spatial and temporal constraints (e.g., locations, trajectories, etc.), the cross-view object association is significantly improved. Compared with [18] and [23], the MOTP and MOTA are both improved. A sample output from our tracking system is provided in Figure 6.

5. Conclusions

In this paper, we propose a novel multi-view pedestrian detection and tracking framework based on smart camera network for surveillance applications. We use the RCNN detector to identify pedestrians in each view and track the detected objects by matching the CNN features. Then we formulate and solve the cross-view object association as a graph matching problem using the object trajectories. The experimental results demonstrate the improvement over previous works. Future studies include (1) the training of a CNN classifier specifically for pedestrian detection and (2) incorporation with additional visual features for more accurate cross-view object association.

Figure 6. Proposed Multiview Pedestrian Tracking System Output Visualization

6. References

[1] C. Papageorgiou and T. Poggio, "A Trainable Pedestrian Detection system", International Journal of Computer Vision (IJCV), pp: 15-33,2000.

[2] N. Dalal, B. Triggs, “Histograms of oriented gradients for human detection”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp:886-893, 2005.

[3] B. Samarth. "Practical OpenCV", Berkeley, California: Apress, 2013.

[4] C. Cortes, V. Vapnik, "Support-vector Networks". Machine Learning. 20 (3): 273–297, 1995.

[5] K. Mikolajczyk, C. Schmid, and A. Zisserman, "Human detection based on a probabilistic assembly of robust part detectors", The European Conference on Computer Vision (ECCV), volume 3021/2004, pages 69-82, 2005.

[6] Y. Xu, L. Xu, D. Li, and Y. Wu, “Pedestrian detection using background subtraction assisted Support Vector Machine”, International Conference on Intelligent Systems Design and Applications (ISDA), 2011.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", Advances in Neural Information Processing Systems (NIPS), 2015.

[8] J. Tu, H. Tao, T. Huang, “Online Updating Appearance Generative Mixture Model for Meanshift Tracking”, in Machine Vision and Applications, Vol: 20, Issue: 3, pp: 163-173, 2009.

[9] Z. Han, Q. Ye, J. Jiao, "Online Feature Evaluation for Object Tracking using Kalman Filter",in International Conference on Pattern Recognition (ICPR), 2008.

[10] A. Bewley, Z. Ge, L. Ott, F. Ramos, B. Upcroft, “Simple Online and Realtime Tracking”, Proc. IEEE International Conference on Image Processing (ICIP), pp. 3464 - 3468, Arizona, USA, September, 2016.

[11] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah,"Part-based Multiple-person Tracking with Partial Occlusion Handling", International Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[12] P. Jahanshahi, A. Masoud, “Multi-view tracking using Kalman filter and graph cut”, in AI & Robotics (IRANOPEN), pp. 1-5, 2015.

[13] X. Wang ,"Intelligent multi-camera video surveillance: A review”, Journal of Pattern Recognition Letter, Vol. 34, pp. 3–19, 2013.

[14] Kang, J., Cohen, I., Medioni, G., “Persistent objects tracking across multiple non overlapping cameras”, In: Proceedings of IEEE Workshop on Motion and Video Computing, pp. 112–119, 2005.

[15] F. Fleuret, J. Berclaz, R. Lengagne and P. Fua, “Multicamera People Tracking with a Probabilistic Occupancy Map”, in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 30, no. 2, pp. 267-282, Feb. 2008.

[16] J. Berclaz, F. Fleuret, E. Turetken and P. Fua, “Multiple Object Tracking Using K-Shortest Paths Optimization”, in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 33, no. 9, pp. 1806-1819, 2011.

[17] L. Leal-Taixe,G. Pons-Moll, B. Rosenhahn, "Branch-and-price global optimization for multiview multi-target tracking", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[18] C. Li, S. Ping, H. Sheng, J. Chen, and Z. Xiong, "Multi-view Multi-object Tracking Based on Global Graph Matching Structure." Pacific Rim Conference on Multimedia, 2016.

[19] Nie W, Liu A, Su Y, Luan H, Yang Z, Cao L, Ji R. Single/cross-camera multiple-person tracking by graph matching. Neurocomputing, 2014.

[20] F. Zhou and F. De la Torre, “Deformable Graph Matching”, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 38(9):1774-1789, 2016.

[21] J. Ferryman,, A. Shahrokni, “PETS-2009: dataset and challenge”, 20th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, pp. 1–6, 2009.

[22] K. Bernardin, and S. Rainer, "Evaluating multiple object tracking performance: the CLEAR MOT metrics.", EURASIP Journal on Image and Video Processing, Issue: 1, pp: 1-10, 2008.

[23] Z. Wu, N.I. Hristov, T.H. Kunz, M. Betke, “Tracking-reconstruction or reconstruction tracking Comparison of two multiple hypothesis tracking approaches to interpret 3D object motion from several camera views”, Workshop on Motion and Video Computing (WMVC), pp. 1–8, 2009.