Moving Vehicle Tracking Using Disjoint-View Multicameras

E. Shabani Nia* and Sh. Kasaei*

Abstract: Multicamera vehicle tracking is a necessary part of any video-based intelligent transportation system for extracting different traffic parameters; such as link travel times and origin/destination counts. In many applications, it is needed to locate traffic cameras disjoint from each other to cover a wide area. This paper presents a method for tracking moving vehicles in such camera networks. The proposed method introduces a new method for handling inter-object occlusions; as the most challenging part of the single camera tracking phase. This approach is based on coding the silhouette of moving objects before and after occlusion and separating occluded vehicles by computing the longest common substring of the related chain codes. In addition, to improve the accuracy of the tracking method, in the multicamera phase, a new feature based on the relationships among surrounding vehicles is introduced. The proposed feature is modeled by an exponential distribution and can efficiently improve the efficiency of the appearance (space-time) features when they cannot discriminate between correspondent and non-correspondent vehicles; due to noise or dynamic condition of traffic scenes. A graph-based approach is then used to track vehicles in the camera network. Experimental results show the efficiency of the proposed method.

Keywords: Disjoint-Views, Multicameras, Occlusion, Tracking, Neighbors' Relationships.

1 Introduction

Interests in intelligent transportation systems (ITS), as an efficient way of monitoring and controlling traffic conditions, come from problems caused by traffic congestions; such as increase of travel time, air pollution, fuel consumption, and reduce of transportation infrastructure efficiency. In ITS, different sensors provide raw data for the transportation management center (TMC) via a communication network. At the TMC, traffic parameters of each site are analyzed and used in controlling signals and message displays along with other traffic control devices.

Usage of video cameras, as powerful sensors for collecting different data, has some main advantages over traditional sensors (e.g., loop detectors). These include the ease of installation, maintenance, and usage. In addition, the cameras provide information that is conceivable by human operators. By the extend of using cameras in ITS, the role of computer vision for automating the process of information extraction from videos has become very crucial.

Evaluating most traffic parameters in a scene requires finding and tracking moving vehicles in a network of cameras. Traffic cameras, depending on their applications, may have overlapped or non-overlapped field of views. Camera overlap is helpful for solving occlusion and depth estimation problems. But, traffic cameras are usually placed non-overlapped (especially in highways) and in a wide distance from each other to cover wide areas to provide more data. Tracking the objects as they move across disjoint camera views is a challenging task and many factors (including changing illumination conditions, different viewing angles, shadows, occlusions, and environmental noise) introduce major challenges in the process.

In this paper, we present a method for tracking vehicles in a network of disjoint-view cameras. The method provides different modules for tracking vehicles in videos captured by single cameras and tracks them in a multicamera fashion. It presents new methods for handling occlusion and improves the precession of multicamera tracking process. The rest of the paper is organized as follow. Section II reviews some related work on vehicle tracking. In Section III, the proposed multi-tracking method is explained. Experimental evaluations of the proposed methods are presented in Section IV. Finally, the paper conclusion is derived in Section V.

Iranian Journal of Electrical & Electronic Engineering, 2011.
Paper first received 14 Nov. 2010 and in revised form 25 June 2011.
* The Authors are with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.
E-mails: shabaninia@ce.sharif.edu, skasaei@sharif.edu.

2 Related Work

There are several frameworks for visual surveillance reported in the literature [1], [2] and [3]. In this paper, the framework reported in [1] is employed; because of its completeness and widespread usage. This framework is divided into several stages. Reviewing the work related to each individual stage is a rather comprehensive task. Several surveys exist which can offer a proper introduction to the subject [4, 5, 6].

There are many tracking methods introduced in the literature. They exhibit different abilities for handling dynamic situations. For tracking multiple vehicles, it is important to maintain the trajectory of moving vehicles before, during, and after an occlusion situation. In [7], three different general tracking methods are proposed; namely, point tracking, kernel tracking, and silhouette tracking.

In point tracking [8], [9] and [10], objects are represented by some points that are corresponded in consecutive frames. But, the method encounters difficulties in handling the occlusion.

In kernel tracking [11], [12] and [13], the representation of object is based on the primitive object region and the parametric motion of the object is computed from one frame to the next. The advantages are in the use of simple geometric shapes and the applicability of kernel tracking for both rigid and non-rigid objects. A well-known kernel tracker is based on the mean-shift method [24]. Mean-shift tracking algorithm is an iterative scheme that is based on histogram comparison of the original object in the current frame and the candidate regions in the target frame. The aim is to maximize the correlation between these two histograms. Hybrid trackers that integrate the respective advantages of mean-shift and particle filter (MSPF) have achieved impressive success in robust tracking [25]. The pivot of MSPF is to sample fewer particles using particle filters. The particles are then shifted to their respective local maximum of target searching space by mean-shift. MSPF not only greatly reduces the number of required particles, but also remedies the deficiency of mean-shift. In the experimental result section we will compare the performance of our proposed tracker with mean-shift and MSPF methods.

For objects with complex shapes (such as hands, head, and shoulders) that cannot be well described by simple geometric shapes, silhouette tracking provides an accurate shape description of objects. The goal of a silhouette-based object tracker is to find the object region in each frame by means of object edges or contour generated by using previous frames [14], [15] and [16]. However, these methods often need a training phase and are highly sensitive to the initialization step.

Although many works are reported for tracking vehicles in a single camera phase, there are little works that address multicamera tracking especially with non-overlapped field of views. In [17], Shan et al. presented an unsupervised learning approach for measuring edges and matching the appearance between non-overlapping views. The matching is based on computing the probability of two observations from different views. Since the method compares edge images of vehicles, the images should be registered together. This constrains the views of cameras to be somewhat similar. In [18], Maden et al. proposed an algorithm for tracking pedestrians in disjoint-view cameras that is based on appearance representation and is capable of dealing with small changes of pose. A matching strategy then extends along the whole available tracks. Ellis et al. [27] determined the topology of a camera network by finding the entry and exit zones of each camera and the links between them using the cooccurrence of entry and exit events, assuming that correct correspondences will cluster in the feature space (location and time).

Other type of trackers is based on Bayesian formulation [19] and [20]. In [19], Huang and Russell defined a physical event space over which probabilities are defined. Then, given that a stream of observation of many objects is available and by introducing an identity criterion they were able to compute the probability that any two objects are the same. In [20], Javed et al. proposed an algorithm for tracking objects in a network of nonoverlapping cameras. By using the kernel density estimation, the algorithm learns the camera topology in the form of multivariate probability density of space-time variables. It also learns the subspace of inter-camera brightness transfer functions to handle the appearance change of an object as it moves from one camera to another. That framework has been followed in [28] and [29] with some novelties in modeling features.

In this paper, we present an efficient method for tracking moving vehicles in a network of disjoint-view cameras, where more attention is paid on occlusion handling in the single camera tracking phase along with modeling new features for improving the accuracy of multicamera tracking. This reasoning is valid especially in highways where groups of vehicles tend to keep their relationships through the road. Matching is then performed separately for each pair of cameras by using the assignment procedure. The proposed method, in contrast to [19], learns the topology of movements and presents a general approach that works not only for two cameras, but also for a network of multi-cameras.

3 Proposed Multicamera Tracking Method

Figure 1 shows the overall block diagram of the proposed multicamera tracking method. It contains the stages of single camera tracking and fusion of multicamera information.

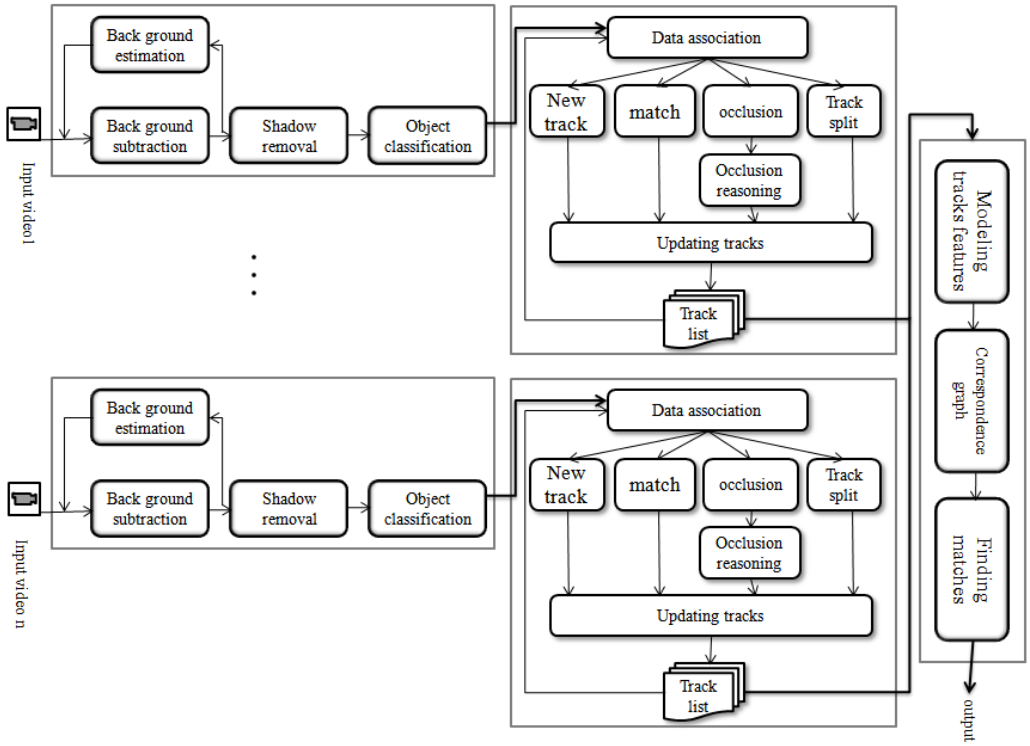

Fig. 1 Overall block diagram of proposed multicamera tracking methods.

3.1 Single Camera Tracking

Tracking as the process of finding and following interested objects in video sequences, is one of the most popular tasks in video surveillance methods. The input to a single camera tracking module is a binary mask of detected vehicles and the output is the trajectory of vehicles. These trajectories are then used for extracting traffic parameters. After detecting and classifying objects as vehicles, it is time to track vehicles in a sequence of frames. This is called data association; which is the strategy of finding the best match for obtained tracks. In this paper, we present a kernel tracker for tracking the bounding box of objects using Kalman filtering. The Kalman filter builds a model for the state of the method that maximizes the a posteriori probability of previous measurements.

Figure 2 shows the pseudo code of the proposed vehicle tracking algorithm, which should be run on each frame. First, an m×n match matrix is considered for m tracks and n detected blobs. Then, this matrix is filled by the overlap of predicted tracks and detected blobs. This matrix can be used to analyze different cases that have been occurred (including appearance of new objects, object lost, matching, object split/merge, or any combination of these cases). A good tracking algorithm must efficiently handle these different situations. Figure 3 shows the pseudo code of the proposed process.

```
If (time = = 0) { // the first frame
      Consider a new track for each detected vehicle
}
Else {
       For (all existing tracks) {
        Predict new location and size of track by Kalman filter
        Overlap new location with all detected blobs
        If (overlap (track (i), blob (j))! =0)
        Mark match_matrix[i][j];
        Analyze ( match-matrix);}
}
```

Fig. 2 Pseudo code of proposed vehicle tracking method.

When no match is found for a track, the track is lost. But, the track should be kept alive for some frames while the Kalman filter continues predicting the new location of the track. If no match is found for that track, during a period of time, it will be ignored. In the case of object match, the Kalman filter and the track mask are updated. We keep a binary mask for each track for the occlusion reasoning algorithm that will be discussed next.

Object splitting is resolved by assigning the largest split part with the track and creating new tracks for the remaining parts. Track merging or occlusion is the most challenging part of data association. Inter-object occlusion occurs when at least two blobs of tracks merge together.

```
Analyze (match_matrix){

For (each row of match_matrix){
  If (row has no mark) // track lost
    If (Tdead > threshold){
              Delete track;
             Tdead++;}
  Else if (row has one mark){  // object match
          Update mask & mean color;
          Correct kalman filter;}
   Else {  //object split
     if (this track is not virtual){
     Assign the most similar detected object to it;
     Create new track for remaining object ;}

     Else {//an occluded object is splitting
     Assign detected objects with their parents ;}
      }
For (each column of matrix){
  If (column has no mark) //new object
  Consider a new track for this detected object;
  Else if(column has more than one mark)//object merging
   occlusion_reasoing();
  }
}
```

Fig. 3 Pseudo code of proposed match matrix analyzer.

When a new track is created, a binary mask of object is kept for it. To be more resistant against noise, a Gaussian filter is used to smooth the mask. This mask is updated when a match is found for the track. Thus, the mask presents the most recent object shape. When occlusion occurs, first we try to resolve the occlusion by detecting tracks that are involved in that occlusion situation and split the region of each track. If it is not possible or reliable, we keep the information of involving tracks to be able to recognize objects after occlusion finishes. We explain these two steps in the following subsections.

3.1.1 Usage of Chain Codes for Occlusion Resolving

An occlusion case is recognized when a column of match matrix has more than one mark (i.e., a detected blob has overlaps with more than one existing track). In this case, occlusion has occurred among overlapping tracks and thus we first split them.

The directional chain code has been widely used in image retrieval for its simplicity and low storage requirement. The first approach for representing digital curves using chain code was introduced by Freeman in 1961 known as Freeman chain code (FCC). This code follows the object contour in a counter clockwise manner and keeps the track of directions as we go from a pixel to the next. Here, an 8-connected FCC is used to represent the vehicle contour.

By keeping an updated version of the track mask, one can code the object contour by FCC. The shape of a solid object does not change widely in consecutive frames. Here, we use the Kalman filter not only for predicting the new track location but also for predicting

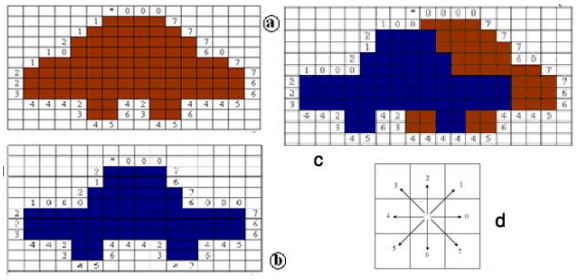

Fig. 4 a) FCC of red car before occlusion {000777607766544 4654324654324443221102l1}. b) FCC of blue car before occlusion{000767600076664467432444654324432210002l}. c) FCC of detected blob in occlusion interval {000077760776654446544444 3 2465412443221000212.

the new object size or even rotation angle. This prediction helps us to scale and rotate the mask, to compensate these changes.

After updating the mask of involved tracks in occlusion situation, based on predicted size of the track, the FCC of tracks and the detected blob are extracted. One can expect common substrings between FCCs of tracks and detected blob. Figure 4 shows the vehicle mask before occlusion that scales in consecutive frames. For this mask and detected blob, FCCs are extracted after bring smoothed by a Gaussian filter. The next step is the comparison of FCCs. We used the longest common substring (LCS) algorithm for this comparison. The LCS problem finds the longest string that is a substring of two or more strings. One can find these substrings in θ(m+n) and compute them by dynamic programming costs θ(mn), where n and m are the lengths of two codes. Note that as the LCS algorithm is only run on the boundary of foreground mask and only in the occlusion situation it is not time consuming.

The algorithm then finds other points of boundary, which are not visible due to occlusion, based on the common substring. It is probable to have more than one LCS in FCCs. The question is then "Which LCS is the best match for the track?". Each common substring separates a set of points with a bounding box, which is assigned as the new track location. Consequently, the best match is the one with the least distance to the predicted location of the track.

3.1.2 Occlusion Management

Sometimes when the length of LCS is not so large or when the distance of predicted track is far from the best set of points, it is not reliable to split tracks. In these cases, we propose an occlusion reasoning strategy as follow (see Figure 5).

Once it is recognized that the object splitting is not reliable, the reasoning algorithm can create a new virtual track of the merged objects and assign these objects as parents of a virtual track. The reasoning algorithm tracks the merged object until the end of

occlusion, and simultaneously predicts the objects involved in occlusion using the parameters obtained before occlusion. If a virtual track tends to split, these parts are compared with track parents and the algorithm decides two trajectories that have the minimum feature distance. In essence, this occlusion reasoning does not make any assumption on the number of objects involving in occlusion. In addition, since it is based on the object shape, it is useful for traffic sequences where the range of vehicle colors is limited.

3.2 Multi View Vehicle Tracking

Tracking vehicles in a network of cameras with disjoint views is the last module of the tracker. The input to this module is the track list of different cameras and the output is a list of correspondent vehicles in different sites. In this paper, we use the formulation of [20] to define the event space and to convert the multicamera tracking problem into a minimization problem. The probability of correspondence of two given observations from two cameras is calculated based on appearance and space-time features of observations. Dynamic condition of environment such as sudden changes of illumination, congestion, or stop lights may invalidate this probability of correspondences.

Thus, we suggest using the relationship among vehicle neighbors to improve the accuracy of the modeling. This reasoning is valid especially in highways where groups of vehicles tend to keep their relationships through the road. Matching is performed separately for each pair of cameras using the assignment problem.

3.2.1 Formulation

The multicamera tracking problem and the matching algorithm are formulated for two cameras. Obviously, this two-camera solution can be generalized for a multicamera problem by considering each two pairs of cameras, separately.

We start by defining the used notations. Here, O_a^i denotes the observation of object a, in camera i (C_i), O^i denotes the set of all observations of all objects passing C_i, O is the set of all observations in C_1 and C_2 (i.e., $O = O^1 \cup O^2$), K_a^b is the correspondence of object a in C_1 with object b in C_2. Then, K_a^b is true if O_a^1 corresponds to O_b^2, and false otherwise. K is a matrix representing a feasible solution of problem with m rows and n columns where these are the number of observations in $C_1$and C_2, respectively. The entry of row i and column j of matrix K is denoted by K_i^j.

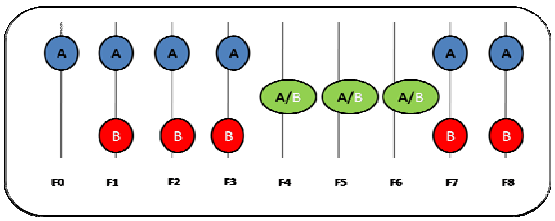

Fig. 5 Occlusion handling example. Two vehicles A and B occlude each other at frame F4. A virtual track is kept for them during occlusion. Virtual track splits at Frame F6. Split parts are assigned with parents of virtual track.

Now, consider two cameras C_1 and C_2 and the list of observed vehicles tracks obtained from the single camera tracking phase. The problem is then finding a matrix K among all feasible solutions that maximizes P(K|O); i.e., given all observations of tracks in C_1 and C_2 if a set of correspondences K gives the maximum probability that it is the solution. This problem is equivalent to

$$K' = \arg\min_{K \in S}(-\log(P(K \mid O))) \quad (1)$$

in which S is the set of all possible correspondence matrixes. Assuming that each matching between two observations is conditionally independent of other matches, we have

$$P(K \mid O) = \prod_{\forall \; K_i^j \in K \;\text{where}\; K_i^j = \text{true}} P(K_i^j \mid O_i^1, O_j^2) \quad (2)$$

Now, the main questions are how to compute the probability of correspondence of two given observations O_i^1 and O_j^2 and how to solve this minimization problem. We will address these two topics in the subsequent sections.

When a certain camera C_i observes some vehicles, it generates a report consisting of various features. Thus, the observation O_a^i in our method forms a feature vector. We call the features of three first rows as space-time (st) features and the mean color of vehicles during the tracking process as the appearance (app) feature. In addition, we use the appearance of a vehicle neighbors as another clue to find the correspondence. Vehicles that have a similar velocity and their time distance is less than a threshold are hypothesized to be neighbors. To compute the probability of (2) it is needed to estimate the probability density function (PDF) of different features. Since features listed in Table 1 are in three distinct categories and are different in nature, three different probability density functions are needed. Here, the Parzen window is used as the correspondence probability estimator of space-time features, while the appearance is modeled by a multivariate Gaussian distribution, and a heuristic method is applied to model the probability of similarity within neighbors.

Modeling the space-time and appearance features, by the Parzen window and Gaussian distribution, requires a training phase to learn some correspondent observations in that environment. Learning is done by assuming that the correspondence is known. One way to achieve this is to manually select the tracks of similar vehicles. To facilitate this task, we developed a GUI tool (see Figure 6). Having the probability of appearance as well as the space-time and neighbors' relationships, the probability of correspondence of two given observations is given by

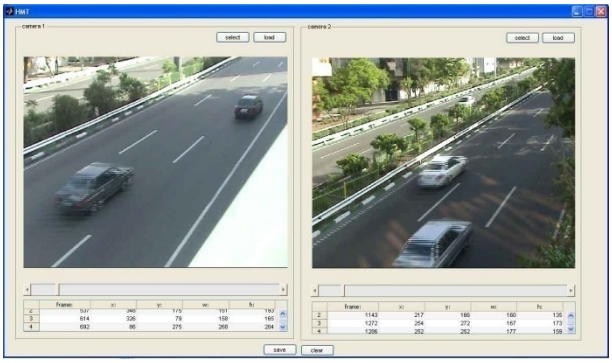

Fig. 6 GUI tool for creating samples or ground-truth.

$$P\left(K_i^j \middle| O_i^1, O_j^2\right) = \left(P\left(K_i^j(app) \middle| O_i^1, O_j^2\right) \times P\left(K_i^j(st) \middle| O_i^1, O_j^2\right)\right)^{\propto} \times P\left(K_i^j(N) \middle| O_i^1, O_j^2\right)^{1-\propto} \quad (3)$$

in which α is the weight of space-time and appearance features against the neighbors' relationships. This probability is high when i) two observations have a high similarity of appearance and their space-time features match together, or ii) the similarity of neighbors of two observations is high enough. Note that using the similarity of neighbors is helpful due to the fact that traffic conditions are dynamic and space-time features may not follow a single distribution.

Table 1 Various features of an observation.

Name	Description	Category
T_{exit} /T_{enter}	Time of exit from region of interest (ROI) of camera or Time of entrance to ROI.	Space-time
L_{exit} /L_{enter}	(x,y) location of exit/enter to the ROI.	Space-time
V	Mean velocity of vehicle in the ROI.	Space-time
Col	Mean color of vehicle in the ROI (HSV).	Appearance
N	Set of neighbors of vehicle in ROI.	Neighbors' Relationship

3.2.2 A Brief Review on Modeling Space-Time Similarity of Observations

Kernel density estimation (or Parzen window) method is a nonparametric way of estimating the PDF of a random variable. Kernel density estimation (KDE) makes it possible to extrapolate the data to the entire population. If X_1, X_2, …, X_n is a d-dimensional independent and identically-distributed (i.i.d.) samples of a random variable, then the kernel density approximation of its probability PDF is given by

$$F(X) = \frac{1}{n|H|^{1/2}} \sum_{i=1}^{n} \ker(H^{-\frac{1}{2}}(X - X_i)) \quad (4)$$

where n is the number of samples obtained from the learning phase, ker is a kernel, and H is a d×d smoothing matrix, called the bandwidth. To reduce the complexity, H is assumed to be diagonal (i.e., $H=diag[h_1^2,h_2^2,\ldots,h_d^2]$), and the diagonal elements are estimated using the correspondent objects in the training set. The multivariate kernel can be generated from the product of symmetric univariate kernels

$$\ker(X) = \prod_{j=1}^{d} \ker(x_j) \quad (5)$$

where x_j are the d-components of X. Quite often ker is taken to be a standard zero-mean and unit-variance Gaussian function. Thus, the variance is controlled indirectly through the parameter h by

$$\ker\left(\frac{x - x_i}{h_i}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h_i^2}}. \quad (6)$$

We use F(X) for calculating $P(K_i^j(st)|O_i^1, O_j^2)$ in (3). In our method, X is a 7D vector comprising space-time features of two given observations in C_1 and C_2, including L_{exit} of O_i^1, L_{enter} of O_j^2 (x,y), difference of T_{enter} of O_j^2 and T_{exit} of O_i^1, and V of O_i^1 and O_j^2.

3.2.3 A Brief Review on Estimating the PDF of Appearance Features

Given two observations O_i^1 and O_j^2, we define a random variable Y that is the difference of the color of O_i^1 and O_j^2. In our method, the mean and variance of this variable is learned during the training phase by observing similar objects in that environment. Therefore, Y is a 3D vector representing the difference of HSV color model components. Fitting a Gaussian kernel to the mean and variance of Y, the PDF is obtained by

$$F(Y) = \frac{1}{\sqrt{(2\pi)^3 |\Sigma|^{1/2}}} \exp\left(-\frac{1}{2}(Y-\mu)^T \Sigma^{-1}(Y-\mu)\right) \quad (7)$$

where μ is the mean and Σ is the variance of Y, assuming that color channels are independent. We use F(Y) to compute $P(K_i^j(app)\ |O_i^1, O_j^2)$ in (3).

3.2.4 Probability of Neighbors Similarity

In this section, we explain how to model the neighbors' similarity as a new feature introduced in our formulation. Two vehicles are considered neighbors, if their time distance and velocity difference are smaller than a threshold. Vehicles tend to keep their relationship as they move through highways (because of constrains of speed in roads and less ramps between camera sites). If N_i^1 denote the set of all vehicles that are the neighbors of vehicle i in C_1 and N_j^2 denote the set of all vehicles neighboring vehicle j in C_2, the similarity of these two groups is defined as

$$Z = |N_i^1| + |N_j^2| - 2n_{match} \quad (8)$$

$$P\left(N_i^1 \text{ similar } N_j^2 \middle| O_i^1, O_j^2\right) = \beta e^{-\beta Z} \quad (9)$$

where "|S|" denotes the cardinality of set S, and n_{match} is the number of correspondent vehicles in two sets with close appearance similarity. Figure 7 shows the pseudo-code of n_{match} calculation.

```
n_match = 0
For (all a in N_i^1)
{
    If ( max_{∀ b∈N_j^2} (P(K_a^b(app) | O_a^1, O_b^2)) > Threshold)
        n_match = n_match + 1
}
```

Fig. 7 Proposed pseudo code for computing n_{match} in (11).

As this figure shows, n_{match} increases when the maximum appearance similarity of a member of N_i^1 with all members of N_j^2 resides in an acceptable range. By having the number of matches in two sets, a new variable, called Z, is computed. Since n_{match} is at most $\max(|N_i^1|, |N_j^2|)$, Z is always nonnegative, and thus we can model the probability of Z by an exponential density function (9). This function reaches its maximum value when the number of matches in two sets is the most, and decreases when n_{match} reduces. β adjusts the slop of the exponential function and should be set so that when n_{match} equals zero the function value vanishes. At this stage we can model the probability of similarity of neighbors of two observations by this exponential function. Figure 8 shows an example of the probability calculation.

3.2.5 Finding Correspondence

Having the probability of correspondence of each two corresponding observations in C_1 and C_2, it is time to find the solution matrix K, such that it minimizes (1). We begin with the simplest case where all vehicles detected in C_1 are also detectable in C_2. In this case, a most probable assignment (pairing all vehicles) can be found by formulating the problem as a weighted bipartite matching problem and using any of several well-known algorithms of bipartite graph matching. To do this, a bipartite graph is constructed where nodes in the left and right partitions are represented by O_i^1 and O_j^2, respectively, and the edge between these nodes is represented by -log $P(K_i^j | O_i^1, O_j^2)$.

A match in a graph G = (V, E) is a subset of M edges E such that no two edges in M share a common vertex V; V= (A∪B) denotes a bipartite graph, where A and B are two kinds of vertices. A match in a bipartite graph assigns vertices of A to vertices of B. A maximum cardinality match is a match with the maximum number of edges. If the edges of the graph have associated weights, then a minimum weight match

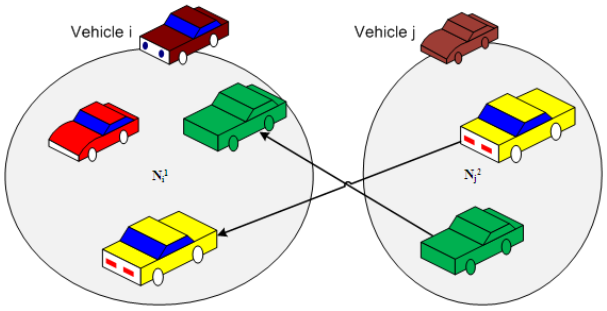

Fig. 8 Example of calculating probability of similarity of neighbors of two vehicles. Vehicles i and j are observed by C_1 and C_2 cameras. N_i^1 and N_j^2 are neighbors of i and j. Here, n_{match} is set to 2 (because green vehicle in N_i^1 has maximum appearance similarity with green vehicle in N_j^2) that is more than threshold. This also is true for two yellow vehicles. From Eq. (8), Z=(3+2)-4 =1, so similarity of this two sets is $\beta e^{-\beta}$.

is a match for which the sum of edge weights is minimized. A minimum-weight maximum-cardinality match is a match with the least weight. The best known algorithm with polynomial time bound for weighted bipartite match is the classical Hungarian method due to Kuhn [23], which runs in time O(|V| (|E |+ |V| log|V|)). Weighted bipartite match algorithms can be implemented efficiently and can be applied to graphs with reasonably large sizes.

In a general case, vehicles can appear or disappear in connections between cameras (i.e., number of observations is not equal in two cameras). A minimum weight maximum cardinality match, on the other hand, would always return the match with maximum cardinality. It guarantees that every vertex in A is matched to a vertex in B. Thus, to handle this case, we used the method in [19] that adds extra nodes to each partition. With m and n vehicles in C_1 and C_2, respectively, the bipartite graph has m+ n nodes in each part to allow all possibilities. Extra nodes are called virtual nodes. In this form, three kinds of edges are distinguishable; namely, edge between two real nodes, edge from a real node to a virtual node, edge from a virtual node to a real node, and edge between two virtual nodes. The edge weight of the first kind is as the case for which there is no virtual nodes in the method (described above). Edge weights of the second and third groups are equal to the –log of the probability of exit between two camera sites and –log of the probability of entrance between two cameras. The probability of exit/enter can be assessed as a prior knowledge in the training phase. The forth group has edges of zero weight.

4 Experimental Results

As there is no standard database available for evaluating the performance of disjoint-view multicamera trackers, in this paper, the proposed tracking method was tested on two provided datasets. The first dataset is a set of videos captured from Resalat

tunnel prepared by Tehran traffic control center. Each sequence in this dataset has a frame rate of 25 frames/second and resolution of 640×480 pixels/frame. The sequences are captured by two cameras placed in east to west of the tunnel in opposite directions. This set is in low quality and populous. The second dataset is recorded by our group for this work. It contains a set of videos captured from Kordestan highway by three cameras placed in a distance of about 500 m apart. Each sequence has a frame rate of 25 frames/second and resolution of 576×720 pixels/frame. Vehicles move in one direction from north to south. There are some ramps between camera sites. Also, some vehicles miss or appear in successive cameras. Figure 9 shows the location of these cameras.

Conducted experiments consider two steps of single camera tracking and multicamera information fusion. Each step has its related metric for evaluation.

Fig. 9 Location of cameras in Kordestan highway (Resalat tunnel is also visible).

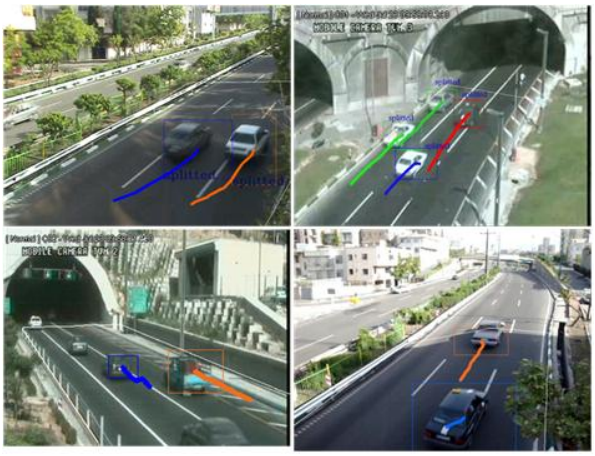

Fig. 10 Single camera tracking results.

In the single camera tracking method, the efficiency of tracker is defined by the ratio of the number of vehicles tracked correctly to the actual number of vehicles as defined in [20]

$$efficiency = \frac{\text{Number of Vehicles correctly tracked by method}}{\text{actual number of vehicles}} \times 100 \quad (10)$$

Table 2 lists the result of this criterion along with the comparison of our proposed method with [24] and [25], on the two datasets. As these results show, the efficiency of the proposed single camera tracker is the same or even more than the hybrid method which is based on mean-shift and particle filtering. As expected, the results on Resalat tunnel are low due to low resolution of the frames. Figure 10 shows the result of this tracker on some frames of this dataset.

Table 2 Result of Eq. (25) on two datasets.

Camera Site	Number of Frames	Mean-shift Method [30]	Mean-Shif + Particle Filtering [31]	Proposed Method
Kordestan Highway, Camera 1	855	%68	%93	%93
Kordestan Highway, Camera 2	1300	%76	%95	%95
Kordestan Highway, Camera 3	1770	%70	%82	%88
Resalat Tunnel, Camera 1	260	%68	%73	%89
Resalat Tunnel, Camera 2	305	%50	%62	%75

For multicamera tracker, a training phase is required to learn the corresponding vehicles in that environment. As shown in Figure 6, a GUI was developed to facilitate the manually creation of correspondent vehicles. In the training phase, it is not needed to find correspondence for all vehicles, but the number of samples should be high enough to estimate the space-time and appearance probabilities properly. In the testing phase,

correspondences are computed using the proposed multicamera vehicle tracking algorithm. In our assignment strategy, a real node may correspond to a real or virtual node. Therefore, it is probable that the number of matches be less than the actual number of matches. We use the recall and precision criteria to evaluate the process [7]

$$recall = 100 \times \frac{\text{true number of matches method finds}}{\text{actual number of matches}} \quad (11)$$

$$precision = 100 \times \frac{\text{true number of matches method finds}}{\text{total number of matches method finds}} \quad (12)$$

Tables 3 and 4 summarize these results for two scenarios of passing from Camera 1 to 2, and from 2 to 3. In each case, about 35 vehicles pass from each camera where about 12 vehicles are common between Cameras 1 and 2, and about 15 vehicles are common between cameras 2 and 3. Results of Equation (3) for different values of α are presented, where α=1 is true for the case in which neighbors' relationships are not considered at all; which is equal to the method of Javed in [20]. For brevity, formulas are written concisely. To better decide on the best α, the F-score measure is used that is in fact the harmonic mean of precision and recall.

$$F - score = \frac{2\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (13)$$

Figure 11 shows the result of the F-score measure on the data of Table 3 and 4. As the results show, for Camera 1 and 2 the value of 0.95 for α gives the best result while this happens in α=0.9 for Camera 2 and 3. In addition, α should be high; because the appearance and space-time features are important hints and as Figure 11 shows selecting α less than 0.85 can result worse than the case in which neighbors' relationships are not considered at all. Generally, the selection of α depends to the used database and environmental condition. Figure 12 shows the result of our matcher for three vehicles in the environment.

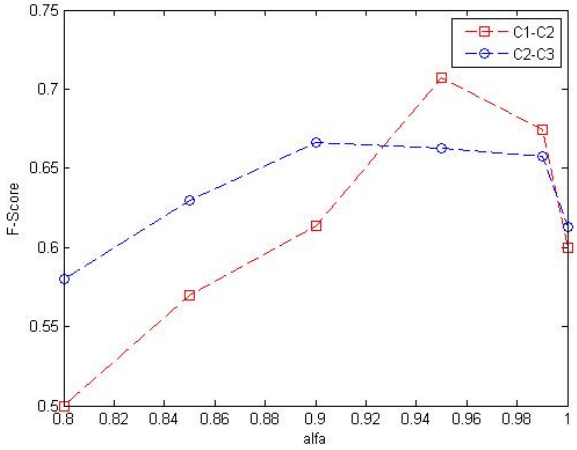

Fig. 11 F-score measure on the data of tables 3 & 4.

Fig. 12 Multicamera tracking results.

Table 3 Recall measure for different αs.

formula	α P(app)×P(st) + (1-α)P(N)					P(app)×p(st). [26]
cameras	α =0.80	α =0.85	α = 0.90	α = 0.95	α = 0.99	α =1
C_1-C_2	0.44	0.53	0.60	0.64	0.66	0.54
C_2-C_3	0.53	0.62	0.65	0.64	0.59	0.55

Table 4 Precision measure for different αs.

formula	α P(app)×P(st) + (1-α)P(N)					P(app)×p(st). [26]
cameras	α =0.80	α =0.85	α = 0.90	α = 0.95	α = 0.99	α =1
C_1-C_2	0.58	0.61	0.62	0.78	0.69	0.67
C_2-C_3	0.64	0.66	0.68	0.68	0.73	0.68

5 Conclusion

In this paper, an efficient method for tracking moving vehicles by using multicameras with disjoint views was presented. We introduced our solutions for tracking vehicles in a sequence of video frames captured by single cameras based on coding the shape of objects before and after occlusion. As the method does not require any hypothesis on the number of vehicles involved in occlusion it is suitable for traffic scenes due to its shape-based reasoning. Multicamera tracking was formulated using the appearance, space-time, and vehicle neighbors' relationships. The reasoning required estimating different probabilities for correspondence of observations. Matching was performed by a bipartite graph and the assignment problem. Experimental results showed the efficiency of proposed method.

Acknowledgement

We would like to express our gratitude for Tehran traffic control center for providing us with the Tehran Resalat tunnel dataset.

References

[1] Hu W., Wang T. and Maybank S., "A Survey on Visual Surveillance of Object Motion and Behaviors", *IEEE Transactions on Methods, Man, and Cybernetics*, Vol. 34, No. 3, pp. 334-352, Aug. 2004.

[2] Moeslund T., Hilton A. and Kruger V., "A Survey of Advances in Vision-Based Human Motion Capture and Analysis", *Journal of Computer Vision and Image Understanding*, *Elsevier Science Inc.*, New York, NY, USA, pp. 90- 126, 2006.

[3] Valera M. and Velastin S., "Intelligent Distributed Surveillance Methods: A Review", *IEEE Proceedings on Vision, Image, and Signal Processing*, Vol. 152, No. 2, pp.192-204, April 2005.

[4] Kastrinaki V., Zervakis M. and Kalaitzakis K., "A Survey of Video Processing Techniques for Traffic Applications", *Image Vision Computing*, Vol. 21, No. 4, pp.359-381, 2003.

[5] Moeslund T., Hilton A. and Kruger V., "A Survey of Advances in Vision-Based Human Motion Capture and Morris, Analysis", *Journal of Computer Vision and Image Understanding*, Vol. 104, No. 2, pp. 90-126, Nov. 2006.

[6] Morris T. and Trivedi M., "A Survey of Vision-Based Trajectory Learning and Analysis for Surveillance", *IEEE Transactions on Circuits and Methods for Video Technology*, Vol. 18, No. 8, pp. 1114-1127, Aug. 2008.

[7] Yilmaz A., Javed O. and Shah M., "Object tracking: A survey", *ACM Computing Surveys (CSUR)*, Vol. 38, No. 4, pp.13-58. 2006.

[8] Salari V. and Sethi I. K., "Feature point correspondence in the presence of occlusion", *IEEE Transactions on Pattern Analysis Mach. Intell.*, Vol. 12, No. 1, pp. 87-91, 1990.

[9] Veenman C., Reinders M. and Backer E., "Resolving motion correspondence for densely moving points", *IEEE Transaction on Pattern Analysis Mach. Intell*., Vol. 23, No. 1, pp. 54-72, 2001.

[10] Streit R. L. and Luginbuhl T. E., "Maximum likelihood method for probabilistic multihypothesis tracking", *Proceedings of the International Society for Optical Engineering (SPIE)*, pp. 394-405, 1994.

[11] Comaniciu D., Ramesh V. and Andmeer P., "Kernel-based object tracking", *IEEE Transaction on Pattern Analysis Mach. Intell.*, Vol. 25, No. 5, pp. 564-575, 2003.

[12] Malik J. and Russell S., "A Machine Vision Based Surveillance Method for California Roads", *PATH project Mov-83 Final Report*, University of California, Berkeley, 1994.

[13] Senior A., Hampapur A., Tian L., Brown L., Pankanti S. and Bolle R., "Appearance models for occlusion handling", *Image and Vision Computing*, Vol. 24, No. 11, pp.1233-1243, 2006.

[14] Bertalmio M., Sapiro G. and Randall G., "Morphing active contours", *IEEE Transactions on Pattern Analysis Mach. Intell.*,Vol. 22, No. 7, pp. 733-737, 2000.

[15] Kang J., Cohen I. and Medioni G., "Object reacquisition using geometric invariant appearance model", *International Conference on Pattern Recongnition (ICPR)*, pp.759-762, 2004.

[16] Sato K. and Aggarwal J., "Temporal spatio-velocity transform and its application to tracking and interaction", *Comput. Vision Image Understand*, Vol. 96, No. 2, pp. 100-128, 2004.

[17] Shan Y., Sahwney H. S. and Kumar R., "Unsupervised learning of discriminative edge measures for vehicle matching between nonoverlapping cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, No. 4, pp. 700-711, April 2008.

[18] Madden C., Cheng E. D. and Piccardi M., "Tracking people across disjoint camera views by an illumination-tolerant appearance representation", *Machine Vision and Applications*, Vol. 18, No. 3, pp. 233-247, August 2007.

[19] Huang T. and Russell S., "Object identification in a bayesian context", *Proceedings of IJCAI*, pp. 1267-1283, 1997.

[20] Javed O., Shafique K., Rasheed Z. and Shah M., "Modeling inter-camera space–time and appearance relationships for tracking across nonoverlapping views", *Computer Vision and*

Image Understanding, Elsevier Science Inc., Vol. 109, No. 2, pp. 146-162, February 2008.

[21] Xiao M., Han C. Z. and Zhang L., "Moving Shadow Detection and Removal for Traffic Sequences", *International Journal of Automation and Computing*, Vol. 4, No. 1, pp. 38-46, January 2007.

[22] Hu W., Hu M., Zhou X., Tan T. et al., "Principal Axis-Based Correspondence between Multiple Cameras for People Tracking", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, Vol. 28, No. 4, pp. 663-671, 2006.

[23] Kuhn H. W., "The hungarian method for the assignment problem" , *Naval Research Logistics Quarterly*, pp. 83-97, 1955.

[24] Comaniciu D., Ramesh V. and Andmeer P., "Kernel-based object tracking", *IEEE Transactions on Pattern Analysis*, Mach. Intell, Vol. 25, No. 5, pp. 564-575, 2003.

[25] Chen T. P., Haussecker H., Bovyrin A. et al., "Computer Vision Workload Analysis: Case Study of Video Surveillance Methods", *Intel Technology Journal*, Vol. 9, No. 2, pp. 109-118, 2005.

[26] Shabaninia E. and Kasaei S., "A novel vehicle tracking method with occlusion handling using longest common substring of chain-codes", *International CSI conference (CSICC2009*), Tehran, Iran, Oct. 2009.

[27] Makris D., Ellis T. J. and Black J. K., "Bridging the gaps between cameras", *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 205-210, 2004.

[28] Shabaninia E. and Kasaei S., "Neighboring Vehicles Modeling for Tracking across Nonoverlapping Cameras", *Iranian Conference on Electrical Engineering (ICEE2010)*, Isfahan, Iran, May 2010.

[29] Chen K., Lai C., Hung Y. and Chen C., "An Adaptive Learning Method for Target Tracking across Multiple Cameras", *IEEE Conf. on Computer Vision and Pattern Recognition,* pp. 1-8*,* 2008.

Elham Shabaninia was born in Iran, in 1984. She received the B.S. and M.S. degrees both in Computer Engineering from Shahid Bahonar University of Kerman and Sharif University of Technology of Tehran in 2006 and 2009, respectively. She is currently working as a lecturer in the Department of Computer Engineering at Shahid Bahonar University of Kerman. Her research interests are object tracking, machine vision and video processing.

Shohreh Kasaei received the B.Sc. degree from the Department of Electronics, Faculty of Computer and Electrical Engineering, Isfahan University of Technology (IUT), Iran, in 1986. She worked as research assistance in Amirkabir University of Technology (AUT), for three years. She then received the M.Sc. degree from the Graduate School of Engineering, Department of Electrical and Electronic Engineering, University of the Ryukyus, Japan, in 1994, and the Ph.D. degree from Signal Processing Research Centre (SPRC), School of Electrical and Electronic Systems Engineering (EESE), Queensland University of Technology (QUT), Australia, in 1998. She was awarded as the best graduate student in engineering faculties of University of the Ryukyus, in 1994, the best Ph.D. student studied in overseas by the ministry of Science, Research, and Technology of Iran, in 1998, and as a distinguished researcher of Sharif University of Technology (SUT), in 2002 and 2010, where she is currently a professor. She is the director of Image Processing Lab (IPL) at Sharif University of Technology. Her research interests are in image processing with primary emphasis on multi-resolution texture analysis, 3D computer vision, 3D object tracking, 3D model building, scalable video coding, image retrieval, video indexing, face recognition, hyperspectral change detection, video restoration, fingerprint authentication, and watermarking.