Tracking and Segmentation of Highway Vehicles in Cluttered and Crowded Scenes

Goo Jun J. K. Aggarwal
Dept. of Electrical and Computer Engineering
The University of Texas at Austin
{gjun, aggarwaljk}@mail.utexas.edu

Muhittin Gökmen
Dept. of Computer Engineering
Istanbul Technical University
gokmen@itu.edu.tr

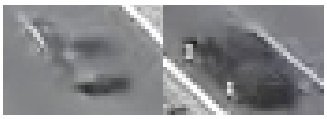
Abstract

Monitoring highway traffic is an important application of computer vision research. In this paper, we analyze congested highway situations where it is difficult to track individual vehicles in heavy traffic because vehicles either occlude each other or are connected together by shadow. Moreover, scenes from traffic monitoring videos are usually noisy due to weather conditions and/or video compression. We present a method that can separate occluded vehicles by tracking movements of feature points and assigning over-segmented image fragments to the motion vector that best represents the fragment's movement. Experiments were conducted on traffic videos taken from highways in Turkey, and the proposed method can successfully separate vehicles in overpopulated and cluttered scenes.

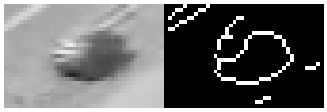
1. Introduction

Computer vision-based traffic monitoring systems are becoming popular because of their flexibility and easy installation as compared to ground-buried loop detectors. Vision-based system have many applications such as detecting illegally parked vehicles [6] or measuring traffic parameters [2]. The tracking and identification of a highway vehicle is a demanding application of the computer vision. The objective of this paper is to develop a vehicle tracking and segmentation algorithm that can work robustly with a low resolution highway monitoring video that is highly compressed.

Analyzing scenes from congested highways is difficult because there are many vehicles occluding each other. We need to segment occluded vehicles to count and identify each vehicle in the image. Model-based methods using the edge model or template matching have been widely studied to address this problem, but in general it is hard to apply models or templates when we have a scene with so many vehicles whose size is very small. It is even more difficult when the video is highly compressed or when weather conditions are bad because those conditions result in noisy images. In practice, highway traffic monitoring videos are usually highly compressed and are in low resolution, and matching models with vehicles in such a scene is difficult. Figure 1 illustrates problems with noisy images. In Figure 1(a), blurring and block artifacts make it hard to identify objects in the scene, and Figure 1(b) shows insufficiency of edge information to apply an edge-based method.



(a) Vehicle images degraded by compression



(b) Edge information is not reliable

Figure 1. Problems with small low-quality images

In this paper, we present a novel method to segment vehicles in a highway traffic monitoring video. The main idea is to divide occluded vehicles into many small fragments (or *patches*) and to group patches according to the clusters of motion vectors found by tracking feature points. The algorithm described in this paper requires no prior models, either for vehicles or shadows, and does not need any training. The proposed system is constructed in several steps:

- **Background subtraction** Using a background subtraction algorithm, foreground vehicles are separated from the background and form a foreground mask.
- **Occlusion detection** The system detects blobs that are suspected of having more than one vehicle in them by morphological analysis.
- **Motion analysis** By tracking feature points in the sus-

pected blobs, a set of motion vectors are obtained. The motion vectors obtained are grouped by a hierarchical clustering algorithm.

- **Vehicle segmentation** Suspected blobs are segmented into many small pieces by an over-segmentation algorithm. Each image patch is assigned to a cluster that best represents the patch's movement.

2. Related Work

There are notable early works in the area of vehicle monitoring and segmentation. Ferrier *et al* [3] proposed a real-time traffic monitoring system in 1994. Beymer *et al*'s 1997 paper [2] describes pioneering studies in real-time traffic monitoring applications using corner point tracking, and the idea of clustering feature points based on their motion is closely related to the segmentation algorithm presented in this paper. Kanhere *et al*'s study in 2005 [5] is also based on a similar idea and further extends Beymer *et al*'s approach. In contrast, Kamijo *et al*'s algorithm [4] is quite different, and they proposed a Markov random field (MRF) model to analyze the spatio-temporal sequence of vehicle images.

The most recent study on vehicle segmentation is by Song and Nevatia [9], in which a model-based approach was employed using 3-D box models, and a probabilistic hypothesis framework was proposed for the box models. The main differences between our work and Song and Nevatia's work [9] are that we are working on very small sized vehicles in the highway traffic videos, and our approach is purely based on the regions (blobs) and feature points; thus, no prior model is required.

3. Method

Figure 2 depicts an overview of proposed algorithm for vehicle segmentation. For a given image, a background subtraction algorithm is applied to obtain foreground masks, from which blobs are obtained by a connected component analysis. Occluded vehicles are detected based on shapes and orientations of blobs. For segmentation, first we compute motion vectors associated with the blob by detecting feature points from consecutive image frames. Obtained motion vectors are then grouped into two clusters, and over-segmented image fragments are assigned to one of the clusters that best represents the fragment's movement. Detailed procedure is described in following sections.

3.1. Background Subtraction

For background subtraction, an enhanced version of the adaptive mixture of Gaussian background model(MoG) [10] is employed. In the adaptive mixture of Gaussian model, each pixel is assumed to have a mixture of Gaussian distribution. For every frame, each pixel is classified

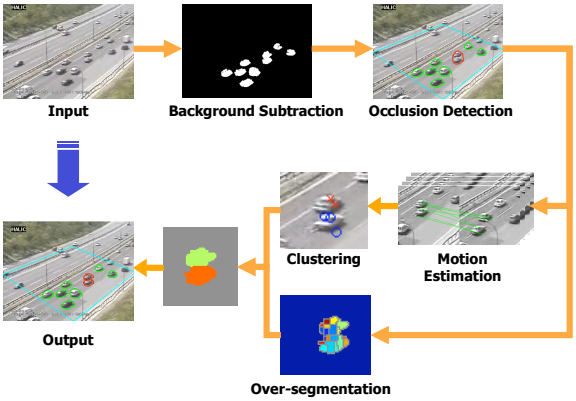


Figure 2. Overview of proposed system

to be a member of one of k Gaussian distributions, and the statistics of each distribution is updated in expectation maximization (E-M) manner. Between k distributions, first b distributions with highest weights are chosen as distributions of a background pixel, depending on the pre-defined threshold. When $b = 1$, the background distribution model becomes unimodal. The MoG background model is especially well suited for outdoor monitoring applications, since it makes the background model adaptable to gradual changes in weather and sunlight conditions.

The MoG algorithm is a pixel-by-pixel approach, and the decision is made only by the statistics of the background pixels. Foreground objects tend to split or have missing parts under noisy environments. Morphological operation can be applied to conceal the omissions; however, it often changes object shapes or connectivity properties as well. A highway traffic monitoring video usually contains large numbers of vehicles in the scene, and the size of each vehicle in the scene is relatively small. Having a few holes in or around a foreground object may make the classification problem even harder.

The background subtraction is a two-class classification problem; each pixel is classified either as a foreground or as a background pixel. Therefore, we can simply solve a two-class classification problem once we have estimated probabilistic distribution of foreground pixels as well as the distribution of background pixels. In the enhanced background subtraction algorithm, we first obtain initial foreground masks by the MoG algorithm and then apply a 3-dimensional connected component analysis on the sequence of foreground masks to construct 3-D blobs. 3-D blobs are then aligned to estimate mean and variance of each pixel in the foreground region. The results are shown in Figure 3 in comparison with the original scene and the foreground mask from the MoG algorithm. As can be seen in the figures, missing parts are successfully restored by the proposed algorithm.

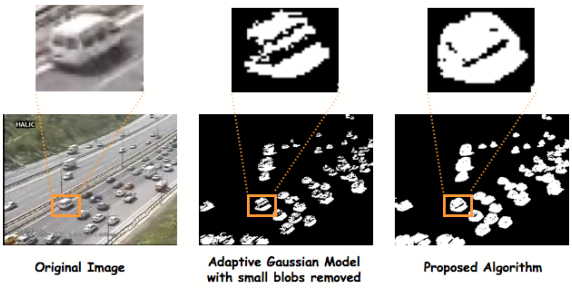


Figure 3. Enhanced background subtraction algorithm

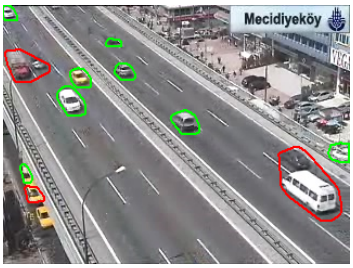


Figure 4. Occluded vehicles are detected in red

3.2. Detecting irregular blobs

As a result of background subtraction, we obtain a foreground mask, which is a binary image that indicates the existence of foreground objects. Blobs are isolated groups of bits that are connected to each other, and blobs are constructed using the connected component analysis. While some blobs contain single objects, some blobs contain more than one object. The objective of this section is to detect blobs composed of multiple vehicles. The detection algorithm is based on the morphological characteristics of blobs: solidity, eccentricity, and orientation. Color-based methods such as color clustering can be used, too; however, color-based methods fail if the colors of occluded vehicles are similar to each other.

In general, a vehicle is roughly a convex object, and the silhouette of a vehicle is very close to its convex hull. Here it is assumed that if there are two or more vehicles in one blob, the shape of that blob would be *less* convex. It is possible that two vehicles are so badly occlude that a blob containing both of them might look like a convex. In such a case the system cannot detect the occluded vehicle because there are too little information about occluded vehicles.

Solidity is defined as the area of a blob divided by the area of a convex hull of the blob. B_i denotes an i-th blob assuming there are total n blobs in the scene and C_i denotes the convex hull of the blob; then, the solidity of the i-th blob, S_i, is defined as $Area(B_i)/Area(C_i)$. Eccentricity and orientation can be obtained by modeling a blob as an ellipse. We look at the orientation of a blob only if the blob is highly eccentric, since the orientation of a circular shape is much more sensitive to noise. Let's denote the eccentricity of i-th blob as E_i, $E_i \in [0, 1]$, and the orientation of the blob as θ_i, $\theta_i \in [0, \pi]$. The orientation of a blob is compared to the angle of the lane, estimated by using line segments found from a Hough transform. The angle of the lane is denoted as θ_L. Using these regional characteristics of blobs, we suspect that B_i contains more than one vehicle in it if:

$$(S_i < T_S) \ \vee \ \{(E_i > T_E) \wedge (|\theta_i - \theta_L| > T_\theta)\} \qquad (1)$$

T_S, T_E, and T_θ are threshold values for solidity, eccentricity, and orientation, respectively. Threshold values are determined heuristically from empirical experiments.

Figure 4 shows the detected blobs with occlusions. The blobs that we suspect to have occlusions have contours in red, and green contours mean that those blobs are regarded as a single vehicle. The thresholds values are not very tight because we want to detect as many occluded vehicles as possible. False positives are preferred to false negatives since we can conclude that the suspected blob has only one moving object at later stages.

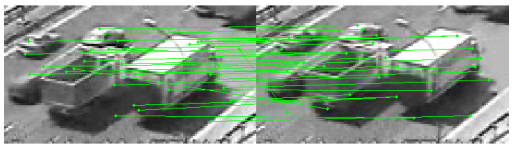


Figure 5. Finding motion vectors with feature points

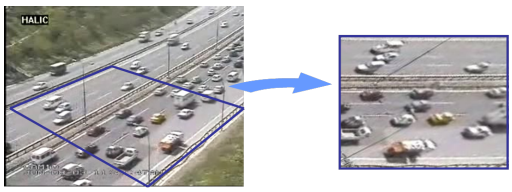


Figure 6. Projective transform for better estimation of vehicle movement

3.3. Clustering of feature points with motion vectors

In this section, we find motion vectors associated with the blob of our interest and cluster those motion vectors to find objects moving differently. To obtain motion vectors, we extract feature points detected repeatedly over a range of frames and track those feature points. By tracking feature points included in the blob, we obtain motion vectors associated with each feature point, and then group those motion vectors to construct clusters of feature points. The scale-invariant feature transform(SIFT) detector by Lowe [7] is used for feature point detection. The points detected by a SIFT detector are invariant under scale and orientation changes; thus, we can detect the same points repeatedly in different frames while the scale of the vehicle changes along the perspective. The more important advantage of the SIFT algorithm is that it provides not only feature point detectors, but also algorithms to obtain local descriptors around

(a) Clusters of motion vectors from two vehicles

(b) Recursive clustering from three vehicles

Figure 7. Examples of clustered feature points

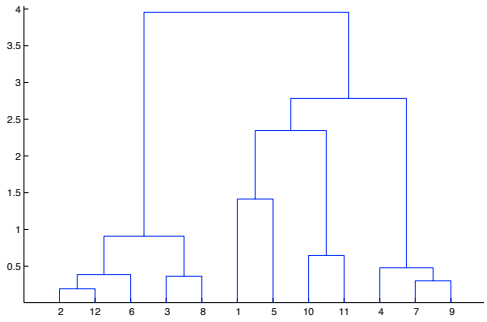


Figure 8. Dendrogram of clustered feature points

detected points. The results of detection and matching between different points are shown in Figure 5.

For better approximation of motion vectors, an off-line camera model was assumed to find a projective transformation that converts the given scene into a perspective-free plane. Video clips used in this paper were not taken from a top-view camera; therefore it is more desirable if we could transform given images to a plane where all motion vectors are approximately parallel to each other. Figure 6 shows original and transformed images. The reference points for the projective transform also serve as a region of interest (ROI) mask, and all blobs outside the ROI are ignored.

Let's assume that the index of the current frame is T; then, we search for SIFT points and descriptors from the $(T-N)$-th frame to the $(T+N)$-th frame, which makes the width of time window $2N+1$, including the current frame. To avoid unnecessary computations, we construct a 3-D blob using a stack of foreground masks and perform the detection only for the sub-images included in the bounding box of the 3-D blob of our interest. Detected points are filtered using the foreground masks to avoid points from the background being used. For easy indexing, assume that indices 1 to N refer to the N frames before the current frame, and indices $N+1$ to $2N$ refer to the N frames after the current frame. Assuming that M feature points are found in the current frame, let's denote the coordinates of detected feature points as $\alpha_{i0} = [p_{i0}\ q_{i0}]^T$, $1 \le i \le M$. If there exists a point whose descriptor matches to the $\mathbf{p}_i$ in frame j, let's call the matching point $\alpha_{ij} = [p_{ij},\ q_{ij}]^T$. Let's also indicate the existence of a matching pair using a $2N$-dimensional binary vector $\mathbf{z}_i$:

$$\mathbf{z}_i = [z_{i\,1}\ z_{i\,2}\ \ldots\ z_{i\,2N}]^T \tag{2}$$

$$z_{ij} = \begin{cases} 1 & \text{if } \exists \alpha_{ij} \text{ in frame } j \text{ that matches } \alpha_{i0} \\ 0 & \text{if } \nexists \alpha_{ij} \text{ in frame } j \text{ that matches } \alpha_{i0} \end{cases} \tag{3}$$

To estimate motion vectors, we perform the *projective transform* using the transform matrix H on the feature points to obtain $\alpha_{ij}^t = [p_{ij}^t,\ q_{ij}^t]^T$, where

$$\alpha_{ij}^t = \begin{bmatrix} p_{ij}^t & q_{ij}^t \end{bmatrix}^T = \begin{bmatrix} a/c & b/c \end{bmatrix}^T \tag{4}$$

$$\begin{bmatrix} a & b & c \end{bmatrix}^T = H \cdot \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^T \tag{5}$$

Then we construct a set of features, $W = \{(\mathbf{x}_i, \mathbf{y}_i) \mid i = 1, ..., M\}$, where $\mathbf{x}_i = [x_1 x_2 ... x_M]$ and $\mathbf{y}_i = [y_1 y_2 ... y_M]$ are N-dimensional vectors that represent horizontal and vertical movements.

$$x_{ij} = z_{ij} \cdot (p_{i0}^t - p_{ij}^t) \tag{6}$$

$$y_{ij} = z_{ij} \cdot (q_{i0}^t - q_{ij}^t) \tag{7}$$

Now we have M feature samples with many missing features. If there are feature samples with too few samples those features are removed, for it means that the corresponding feature point was not detected for many frames and the information is less reliable. To find distances between feature samples, first we normalize our features with the time index $\mathbf{k}$:

$$\bar{\mathbf{x}}_i = \mathbf{x}_i \circ \mathbf{k} \tag{8}$$

$$\bar{\mathbf{y}}_i = \mathbf{y}_i \circ \mathbf{k} \tag{9}$$

$$\mathbf{k} = \left[\frac{1}{N}\ \ \frac{1}{N-1} \ldots 2\ \ 1\ \ 1\ \ 2 \ldots \frac{1}{N-1}\ \ \frac{1}{N} \right]^T \tag{10}$$

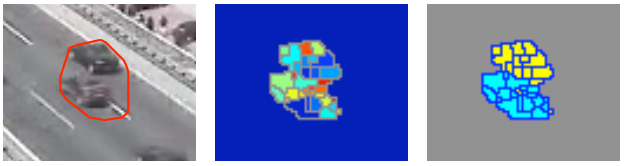
where $\circ$ denotes an element-wise product between two matrices while $\cdot$ indicates an inner product. Because we have many missing values, we cannot directly calculate the Euclidean distances between motion vectors. Therefore, here we define an alternative distance measure between two features:

$$d_x(m,n)^2 = \frac{[(\mathbf{z}_m \circ \mathbf{z}_n) \circ (\bar{\mathbf{x}}_m - \bar{\mathbf{x}}_n)]^T \cdot (\bar{\mathbf{x}}_m - \bar{\mathbf{x}}_n)}{\mathbf{z}_m \cdot \mathbf{z}_n} \tag{11}$$

$$d_y(m,n)^2 = \frac{[(\mathbf{z}_m \circ \mathbf{z}_n) \circ (\bar{\mathbf{y}}_m - \bar{\mathbf{y}}_n)]^T \cdot (\bar{\mathbf{y}}_m - \bar{\mathbf{y}}_n)}{\mathbf{z}_m \cdot \mathbf{z}_n} \tag{12}$$

$$d(m,n) = \sqrt{d_x(m,n)^2 + d_y(m,n)^2} \tag{13}$$

The meaning of $\mathbf{z}_i$ in the distance measure is that we are considering the distance between each element of feature vectors only for the overlapping elements. When there is no overlap for a certain pair of features, then we interpolate missing values using the linear regression analysis and then obtain the average distance between two vectors.



(a) Original (b) Watershed (c) Segmented

Figure 9. Vehicle segmentation using over-segmented patches from the watershed transform

Using the set of pairwise distances, $d(m, n)$, hierarchical agglomerative clustering is performed to construct at most two clusters: C_1 and C_2. The reason we construct at most two clusters is because we do not know how many vehicles are included in the blob, and we guess as small a number as possible. If there are more than two vehicles in the blob, then the segmented blobs are tested again for irregularity and recursively segmented into more segments. Clustered feature points are plotted on the image in Figure 7. Figure 7(a) shows the clustering results when there exist only two vehicles in the blob, and Figure 7(b) demonstrates the process of recursive segmentation. Figure 8 shows an example of the dendrogram resulting from the hierarchical agglomerative clustering. Sometimes only one cluster is constructed, and in that case we conclude that the blob contains only one moving object.

3.4. Vehicle segmentation

This section describes the actual segmentation of vehicles using the motion information obtained in the previous section. The segmentation of vehicles solely based on their appearance is not easy because of the variability of their shapes and colors. Image segmentation in general is still an unsolved problem; however, the problem becomes more tractable when we have a sequence of images with moving objects. Moving objects can be separated from the background and from each other more easily using their motion information. The main idea presented in this section is over-segmenting vehicles into smaller patches at first, finding clusters of motion vectors between frames, and then assigning each patch to one of the clusters.

Watershed transform is used for the over-segmentation [1]. We employed the watershed transform since other methods such as the superpixel algorithm [8] require too much computation. The watershed transform is a morphological transform that can be used to find natural boundaries in gray-scale images. Watershed transform can be computed in very little time and gives us tight boundaries between objects. One of the disadvantages of the watershed transform is that the algorithm produces pretty small patches, which makes it harder to decide where the patch belongs since the small patch does not contain much information. To resolve this issue, a simple post-processing is applied by consolidating small image fragments whose cluster labels are different from neighborhood into bigger patches.

For each patch, the average of absolute error between the patch and each frame is calculated using the motion vectors from each cluster, and the patch is assigned to the cluster which yields the smallest error. Let's assume that P_k^t denotes a projected version of the k-th patch, and the patch has $N(P_k)$ points in it. $I_j^t(x, y)$ denotes the image data of the j-th frame, $1 \leq j \leq 2N$. The assignment algorithm is as follows:

$$V_k(i,j) = \frac{1}{N(P_k)} \sum_{P_k^t} |I_j^t(x - x_{ij}, y - y_{ij}) - P_k^t| \quad (14)$$

$$S_k(b) = \frac{1}{|C_b|} \sum_{\mathbf{w}_i \in C_b} V_k(i,j) \quad (15)$$

- Assign P_k to clsuter 1, if $S_k(1) < S_k(2)$.
- Assign P_k to clsuter 2, if $S_k(1) > S_k(2)$.

$I_j^t(x - x_{ij}, y - y_{ij})$ is the shifted version of $I_j^t(x, y)$.

Figure 9 shows an example of segmented patches. Figure 9(a) is the original image with two vehicles connected in a blob, and Figure 9(b) shows the watershed transformed image. Figure 9(c) shows the final result of the proposed algorithm.

4. Experiments and Results

The traffic video clips were taken from highways in Turkey. The video clips have resolutions of 320 by 240 and 25 frames per second. The image sequences were resampled at 12.5 frames per second for processing. Experiments were performed on the videos taken from two different locations, Mecidiyekoy and Halic. Vedaldi's MATLAB/C version SIFT implementation [11] is used for feature point extraction and descriptor matching. The watershed transform and lane finding with Hough transform are done using MATLAB's image processing toolbox.

Table 1 shows the detection rate and segmentation results tested on the source video. Each video is 30-seconds long. The traffic in the Halic video is much heavier than in Mecidiyekoy's, and the result for the Halic video shows lower detection rates. Note that an average of 8 vehicles per frame exist in the Mecidiyekoy clip, while there are more than 10 vehicles per frame in the Halic clip, and the proportion of occluded vehicles in the Halic video is even worse. Although our video clips have lower screen resolutions than the typical resolutions in comparable studies (more than 640 by 480), we have even more vehicles in one scene than previous studies. The number of detected vehicles includes only correctly segmented vehicles (i.e. true positive), and overly segmented patches are not included in the count. The detection rate is good for such heavy traffic situations, and there are not many over-segmented fragments (i.e. false positives). Figure 10 shows the final output

Video	Number of frames	Detection			Segmentation		
		Number of vehicles	Detected vehicles	Accuracy	Connected vehicles	Correctly segmented	Over-segmentation
Mecidiyekoy	376	3034	2953	97.3%	365	284	45
Halic	376	4076	3505	86.0%	2108	1537	147

Table 1. Vehicle detection and segmentation results

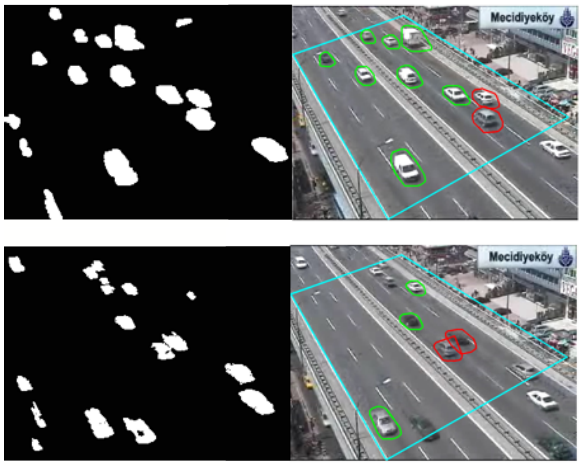


Figure 10. Segmentation results from the Mecidiyekoy video

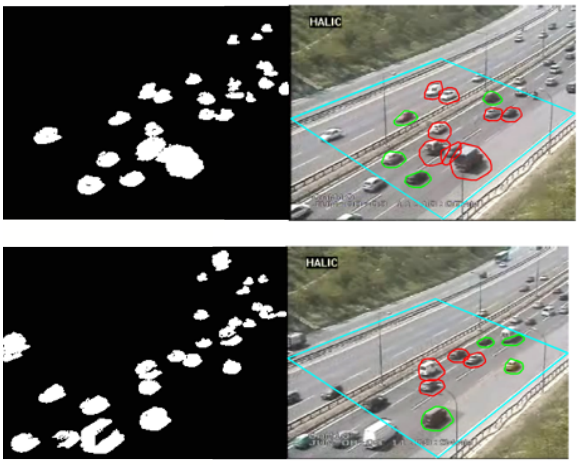


Figure 11. Segmentation results from the Halic video

of the proposed system for the Mecidiyekoy video, and figure 11 shows the results for the Halic video.

5. Conclusion

In this paper, we presented a novel algorithm that can track and separate highway vehicles in low quality highway monitoring videos to which methods based on shape or edge models are not easily applicable. Our segmentation method first finds motion vectors associated with the occluded vehicles and then assigns over-segmented image patches to the group of motion vectors that best represent the movement of the patch. The suggested algorithm shows excellent results for high traffic situations and also works well with highly compressed, low resolution videos.

Acknowledgements

We would like to thank Istanbul Metropol Municipility for providing the traffic videos.

References

[1] S. Beucher. The watershed transformation applied to image segmentation. *Conference on Signal and Image Processing in Microscopy and Microanalysis*, pages 299–314, 1991.

[2] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. *Computer Vision and Pattern Recognition*, 1997.

[3] N. J. Ferrier, S. M. Rowe, and A. Blake. Real-time traffic monitoring. *Applications of Computer Vision, Proceedings of the Second IEEE Workshop on*, pages 81–88, 1994.

[4] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi. Occlusion robust tracking utilizing spatio-temporal markov random field model. *Pattern Recognition, 15th International Conference on*, 2000.

[5] N. K. Kanhere, S. J. Pundlik, and S. T. Birchfield. Vehicle segmentation and tracking from a low-angle off-axis camera. *Computer Vision and Pattern Recognition, IEEE Conference on*, 2:1152–1157, 2005.

[6] J. T. Lee, M. Ryoo, M. Riley, and J. K. Aggarwal. Real-time detection of illegally parked vehicles using 1-d transformation. *Advanced Video and Signal Based Surveillance, IEEE International Conference on*, 2007.

[7] D. G. Lowe. Object recognition from local scale-invariant features. *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2:1150–1157, 1999.

[8] X. Ren and J. Malik. Learning a classification model for segmentation. *Proc. 9th Int'l. Conf. Computer Vision*, 1:10–17, 2003.

[9] X. Song and R. Nevatia. Detection and tracking of moving vehicles in crowded scenes. *Motion and Video Computing, IEEE Workshop on*, 2007.

[10] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition, IEEE Conference on*, 1999.

[11] A. Vedaldi. An open implementation of SIFT. http://vision.ucla.edu/~vedaldi/code/sift/sift.html.