Copyright
by
Birgi Tamersoy
2009

Vehicle Detection and Tracking in Highway Surveillance Videos

by

Birgi Tamersoy, B.S.

THESIS

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in Engineering

THE UNIVERSITY OF TEXAS AT AUSTIN
August 2009

Vehicle Detection and Tracking in Highway Surveillance Videos

APPROVED BY

SUPERVISING COMMITTEE:

J. K. Aggarwal, Supervisor

Kristen Grauman

Michael Ryoo

Acknowledgments

First of all, I would like to thank Dr. J. K. Aggarwal for his continuous support, guidance and motivation. It is certainly a privilege for me to learn directly from the master.

I would like to thank Dr. Kristen Grauman for her valuable comments and inspiring lectures. It is for sure that these lectures shaped my vision.

I would like to thank Dr. Michael Ryoo for his help in this thesis, and the rest of the Computer and Vision Research Center members for creating a friendly and sincere research environment.

I would like to thank Dr. Alper Sen and Dr. Ramesh Yerraballi for supporting me with TA positions during my research.

I would like to thank Dr. Muhittin Gokmen for providing us with the surveillance videos used in this work.

Finally, I am grateful to my family for mentally and physically supporting me all the time.

Vehicle Detection and Tracking in Highway Surveillance Videos

Birgi Tamersoy, M.S.E.
The University of Texas at Austin, 2009

Supervisor: J. K. Aggarwal

We present a novel approach for vehicle detection and tracking in highway surveillance videos. This method incorporates well-studied computer vision and machine learning techniques to form an unsupervised system, where vehicles are automatically "learned" from video sequences. First an enhanced adaptive background mixture model is used to identify positive and negative examples. Then a video-specific classifier is trained with these examples. Both the background model and the trained classifier are used in conjunction to detect vehicles in a frame. Tracking is achieved by a simplified multi-hypotheses approach. An over-complete set of tracks is created considering every observation within a time interval. As needed hypothesized detections are generated to force continuous tracks. Finally, a scoring function is used to separate the valid tracks in the over-complete set. The proposed detection and tracking algorithm is tested in a challenging application; vehicle counting. Our method achieved very accurate results in three traffic surveillance videos that are significantly different in terms of view-point, quality and clutter.

Table of Contents

Acknowledgments **iv**

Abstract **v**

List of Tables **viii**

List of Figures **ix**

Chapter 1. Introduction **1**
- 1.1 Motivation 1
- 1.2 Vision Based Monitoring Systems 2
- 1.3 Challenges 3
- 1.4 Approach 4
- 1.5 Roadmap 5

Chapter 2. Related Work **6**
- 2.1 Vehicle Detection and Tracking 6
  - 2.1.1 Vehicle Detection in Ground-level Videos 8
- 2.2 Object Tracking in General 9
- 2.3 Contribution 9

Chapter 3. Vehicle Detection using Unsupervised Learning **11**
- 3.1 Symbols and Definitions 12
- 3.2 Identifying the ROI 12
- 3.3 Background Subtraction 13
- 3.4 Training Phase 16
  - 3.4.1 Forming the Training Sets 16
  - 3.4.2 Feature Extraction 19
- 3.5 Detection Phase 20
- 3.6 Experimental Results 22
- 3.7 Discussion 25

Chapter 4. Vehicle Tracking **30**
4.1 Symbols and Definitions 31
4.2 Creating an Over-complete Set of Tracks 31
4.3 Improving the Vehicle Detection Results 35
4.4 Merging Tracks 37
4.5 Backward and Forward Localization 38
4.6 Experimental Results 40
4.7 Discussion 43

Chapter 5. Calculating the Traffic Parameters **45**
5.1 Symbols and Definitions 46
5.2 Traffic Flow Theory and Important Traffic Parameters 46
5.3 Counting the Vehicles 47
5.4 Experimental Results 48
5.5 Discussion 50

Chapter 6. Conclusions and Future Work **51**

Bibliography **52**

Vita **55**

List of Tables

3.1 Parameters and symbols used in Chapter 3. 12
3.2 Training statistics. 22
3.3 Frame-by-frame detection results. 23
3.4 Frame-by-frame detection results for the baseline algorithm. 25

4.1 Parameters and symbols used in Chapter 4. 31
4.2 Frame-by-frame detection results before tracking. 42
4.3 Frame-by-frame detection results after tracking. 42

5.1 Parameters and symbols used in Chapter 5. 46
5.2 Vehicle counting results. 48
5.3 Performance comparison with a baseline. 50

List of Figures

1.1 The motivation. 1
1.2 An example of a low quality video. 3

3.1 Vehicle detection system overview. 12
3.2 Example traffic surveillance videos. 13
3.3 Determining the ROI in Halic dataset. 14
3.4 Determining the ROI in Elmali dataset. 14
3.5 Determining the ROI in Mecidiyekoy dataset. 15
3.6 Results of the enhanced adaptive background mixture model. 16
3.7 Blob areas in four datasets. 17
3.8 Relative positive and negative example centers. 18
3.9 Positive and negative training examples from Elmali and Halic datasets. 18
3.10 Processing a suspicious blob. 21
3.11 ROC curves for the classifiers. 24
3.12 Good detection results in Mecidiyekoy (T) dataset. 26
3.13 Good detection results in Mecidiyekoy (B) dataset. 27
3.14 Good detection results in Elmali dataset. 28
3.15 Good detection results in Halic dataset. 29
3.16 Bad detection results. 29

4.1 Vehicle tracking system overview. 32
4.2 Observation window illustration. 32
4.3 Creating an over-complete set of tracks. 33
4.4 Motivation behind the tracking algorithm. 36
4.5 Improving the frame-by-frame detection results using tracking. 37
4.6 One vehicle having two tracks. 37
4.7 Eliminating FP tracks using merging. 38
4.8 Backward and forward localization. 39
4.9 Benefits of backward and forward localization. 40
4.10 Manually selected ROIs for the Mecidiyekoy dataset. 40
4.11 Detection results before and after tracking. 44

Chapter 1

Introduction

1.1 Motivation

Figure 1.1: The motivation[1].

Transportation, and hence traffic, is becoming an inevitable part of our daily life. Especially in big cities, people are spending a substantial time on daily commutes and it is by now clearly evident that using more resources, such as building

---

[1] Thanks to Chia-Chih Chen for locating this photo at `http://ffffound.com/image/819a732d51fc9456a217a24877bbe6e1ad63830c`.

more highways or extending the existing ones, is not enough. We have to develop "intelligent" traffic monitoring systems and use them to utilize the existing resources as good as possible.

The ultimate goal is to have fully automated monitoring systems that can observe, reason and decide. Such a system would gather information from several highways, analyse this information to form efficient detours and finally re-direct the new coming drivers to these calculated detours, minimizing the traffic and maximizing the overall throughput.

In this work we focus on detecting and tracking vehicles in highway surveillance videos. Irrespective of the sensor being used (it may be an on-road detector, or a surveillance camera as in our case), this analysis is required in order to make decisions.

1.2 Vision Based Monitoring Systems

Vision based monitoring systems have always been of more interest than on-road detector based system. There are two primary reasons for this: cost and reliability. On-road detectors, such as the loop detectors, use a sensor loop to detect vehicles. As a vehicle moves over the sensor loop, its metallic mass creates an electric current in the sensor. These systems have high costs and safety risks associated with their installation and maintenance. Besides, most of the highways already have existing video surveillance infrastructures, whereas very few of them have loop detectors.

Reliability is another concern. On-road detectors are point detectors and can only provide a limited amount of information. Surveillance cameras, on the other hand, provide a more thorough information of the whole scene. One may compute vehicle trajectories from surveillance videos. With these trajectories, traffic parameters will be calculated more reliably. Moreover, vehicle trajectories can be used to recognize interesting activities such as accidents and illegal actions.

1.3 Challenges

Detecting and tracking vehicles in highway surveillance videos is a challenging task. This is mainly because, the hardware systems that are used to capture these videos differ significantly. In general, one may not make any assumptions about the position of the camera, other than the fact that it will be on a relatively elevated place. You may not expect to have high quality, high resolution videos either. Especially with old surveillance equipment, effects of the lossy data compression and the noise introduced by the system becomes dramatic (see Figure 1.2). Furthermore, the accuracy of the proposed algorithm should *not* depend on lighting, weather or traffic conditions.

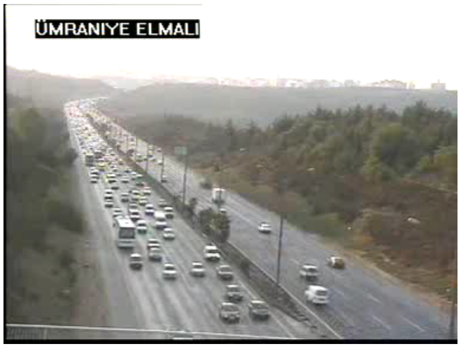


Figure 1.2: An example of a low quality video.

1.4 Approach

Researchers have approached this challenging problem in various ways. The proposed algorithms can be grouped in four categories: model based methods [24, 12, 22, 27], region based methods [16], active-contour based methods [13] and feature based methods [5, 10]. All of these are well-studied computer vision techniques for detecting and tracking objects in general. However, each of them has some pitfalls associated with it, and none of them is robust enough to work with all high/low quality, high/low resolution and heavy/light traffic surveillance videos.

We developed a new vehicle detection and tracking algorithm which is based on "learning" the vehicles in a surveillance video, *as they appear in the video.* The primary distinction of this method is that it does not make restrictive prior assumptions about the scene, the video or the vehicle appearances. This results in an extremely robust approach, which can handle surveillance videos that are significantly different in terms of view-point, quality and resolution.

Similar machine learning techniques have been widely used in other areas such as object classification and even in Intelligent Transportation Systems (ITS). However, most of the time this learning phase is not autonomous. We, on the other hand, incorporated well-known computer vision techniques with the machine learning techniques to derive an *unsupervised* system.

In our detection module, positive and negative vehicle examples are identified automatically. Then these examples are used to train a video-specific binary classifier. Vehicles in a video frame are detected using this binary classifier in conjunction with an adaptive background model.

For tracking, we derived a simplified multi-hypotheses algorithm instead of employing a straightforward linear dynamic model or a complicated probabilistic multi-hypotheses approach. In this method, an over-complete set of tracks is maintained using both observed and hypothesized detections. Then a scoring function is applied to determine the tracks corresponding to real vehicles.

1.5 Roadmap

In the following chapter we review previous research on vehicle detection and tracking in highway surveillance videos. In Chapter 3 we explain our novel vehicle detection approach. Chapter 4 presents our tracking algorithm. We explain a common application of vehicle detection and tracking in Chapter 5. Finally, we conclude and discuss possible future work in Chapter 6.

Chapter 2

Related Work

In this chapter we review previous work on vehicle detection and tracking. Even though our focus is on *detecting and tracking vehicles in highway surveillance videos*, we also mention previous research from related application areas.

2.1 Vehicle Detection and Tracking

Notable work in this area may be traced back to the early 1990s, where Sullivan [24] and Koller et al. [12] proposed model-based vehicle detection approaches. They detected the vehicles by matching the dominant lines in a frame to the edges of a projected 3D vehicle model. Even though these methods are capable of providing very detailed information about the vehicles, they cannot be used once the vehicles are partially occluded.

Michalopoulos [17] presented a background subtraction based algorithm. In his system, background is estimated using a simple adaptive model and vehicles are detected using background suppression. This detection process is performed in multiple user-defined spots along the highway. Then a “spatio-temporal state tracker” coalesces these individual detections to form a high confidence decision about the vehicle. Other vehicle detection algorithms based on various background models have been proposed [20, 9, 18]. The primary shortcoming of these background subtraction only methods is that they cannot handle the situations where the vehicles are partially occluded.

Koller et al. [13] proposed a deformable contours based algorithm to cope with partial occlusions. In their “occlusion reasoning” step the ground plane constraint is used to depth-order the vehicles. They handled the tracking using two linear Kalman Filters, where one is used to estimate the motion and the other one is used to estimate the shape. This tracking algorithm is effective in determining the occlusions that eventually occur, but in the case of crowded scenes where the vehicles enter the scene partially occluded, it will have significant errors.

In order to address the partial occlusion problem, Coifman and Beymer [5] proposed a feature based tracking algorithm. They detected the “corner” features and then grouped them according to a “common motion constraint”. Similarly, Jun et al. [10] used a feature based approach to resolve occlusions. They over-segmented the “irregular blobs” and then cluster the pieces according to the common motion constraint of the extracted features. However, both algorithms depend solely on the accuracy of feature detection and matching, which makes them error prone in noisy, low resolution videos. Moreover, common motion constraint is not applicable in very crowded scenes, where the vehicles are forced to move at similar speeds.

A relatively different approach for handling occlusions in vehicle detection and tracking was proposed by Kamijo et al. [11]. They partitioned the input frame into 8 × 8 pixel blocks. Detection and tracking of vehicles is done in the block-level using background subtraction and block matching. Similar to the deformable contour based approach mentioned before, this part based tracking method may have errors in highway scenes where vehicles enter the scene partially occluded and forced to move in similar speeds.

More recently, Song and Nevatia [22] incorporated a 3D model-based detection approach with background subtraction. They created 2D templates from the 3D vehicle models and used them to generate multiple hypotheses for a given foreground mask. This approach makes use of the template contours, so like the feature-based approaches, its performance on noisy, low resolution and crowded scenes is uncertain.

Papageorgiou and Poggio [19] approached a more general problem, object detection, from a machine learning perspective. They presented a trainable system which may be used for detection of several different object classes, including vehicles in a highway surveillance video. Their training is viewpoint-dependent and requires different sets of "labeled" positive and negative examples for each particular viewpoint. Forming these training sets may be costly.

2.1.1 Vehicle Detection in Ground-level Videos

There has also been a significant research in ITS about vehicle detection in ground level videos (see Sun et al. [25] for a review). Betke et al. [2] used the relative motion as a cue to detect the passing vehicles. They detected the distant vehicles using a feature based approach where they evaluate the horizontal and vertical edges in a frame.

van Leuven et al. [27] followed a simplistic approach in vehicle detection. They created a small number of basic templates which describe the outline of a vehicle and then used Chamfer 3/4 distance transform to locate the matches in a frame. Like most of the other template-based ground-level detection approaches [25], these two algorithms assume no occlusions and fairly good quality, high resolution videos. Due to the limitations of template-based methods, more research is headed towards appearance methods, where vehicle detection is approached as a two-class pattern classification problem [25]. Like Papageorgiou and Poggio's [19] approach, these on-road vehicle detection methods require labelled training sets.

2.2 Object Tracking in General

In most of the previous work discussed in Section 2.1, the tracking is achieved using linear dynamic models or Kalman Filters. These first-order Markovian methods cannot recover from failure.

Leibe et al. [14] addressed this issue by employing a multi-hypotheses tracking framework. They generated an over-complete set of hypothetical models and determined the best subset using the minimum description length criterion. This method is used to track pedestrians *and* vehicles in a city scene from a moving vehicle.

Ryoo and Aggarwal [21] proposed a similar multi-hypotheses tracking approach. Their method "observes" a scene until enough information is obtained and then probabilistically generates the most likely "explanation".

Both of these multi-hypotheses methods perform very well in complex scenes, but the highway surveillance imposes some restrictions that can (and should) lead to simpler approaches.

2.3 Contribution

This work has two main contributions:

A novel method for vehicle detection in highway surveillance videos is presented. This method incorporates well studied computer vision and machine learning techniques to form an unsupervised system. The proposed approach addresses most of the aforementioned problems such as partial occlusions, low quality and noisy videos, and costly viewpoint dependent training.

Moreover, a simple, but effective vehicle tracking algorithm is introduced. This tracking approach considers both the near-past *and* the near-future observations in the process of creating vehicle tracks. This algorithm also generates hypothesized detections as needed, and hence behaves like a simplified multi-hypotheses approach.

Performance of the developed vehicle detection and tracking algorithm is tested on a challenging task; counting vehicles in traffic surveillance videos that are significantly different in terms of view-point, quality and clutter.

Chapter 3

Vehicle Detection using Unsupervised Learning

As mentioned in Chapters 1 and 2, vehicle detection in highway surveillance videos is a challenging problem. These challenges include; large view-point changes, low quality videos and crowded scenes. Hence, a good detection algorithm must be robust enough to cope with all these challenges.

Figure 3.1 presents an overview of our approach. This approach has two major phases: the training phase and the detection phase. In the training phase some positive and negative vehicle examples are identified from background subtraction. Then these two sets of image patches are used to train *video-specific* vehicle classifiers. In the detection phase, background subtraction is used to determine the blobs that are suspected to have more than one vehicle. These blobs are further processed using the vehicle classifier obtained in the training phase.

The rest of the chapter is organized as follows: Section 3.2 explains an optional pre-processing step. In this step a region-of-interest (ROI) is identified for the surveillance video on hand. Section 3.3 reviews the employed background subtraction algorithm. In Sections 3.4 and 3.5 training and detection phases are explained in detail. Section 3.6 presents the results of the derived detection algorithm on three surveillance videos that are significantly different in terms of view-point, quality and clutter. The chapter is concluded in Section 3.7.

---

[0] The work presented in this chapter is published in [26].

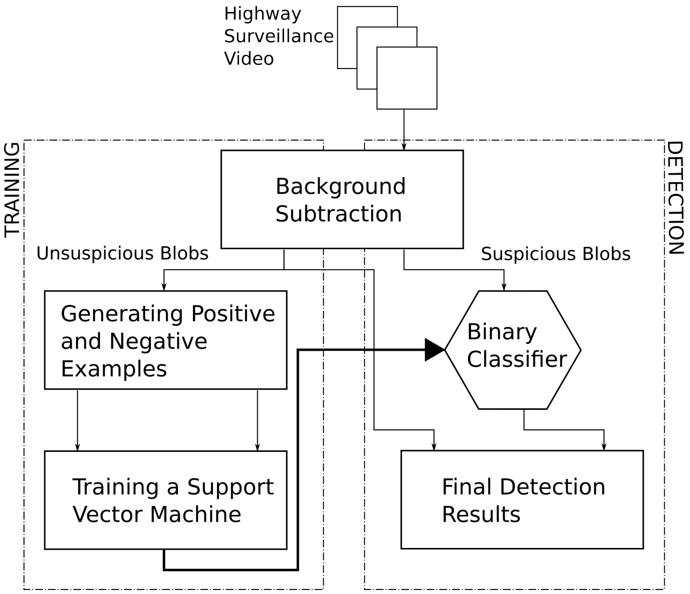


Figure 3.1: Vehicle detection system overview.

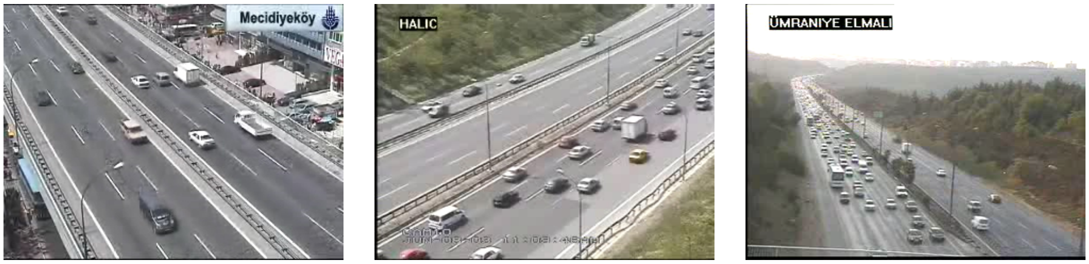
3.1 Symbols and Definitions

Symbol	Definition
τ_{area}	Blob area threshold.
w_s	Median width of the positive vehicle examples.
h_s	Median height of the positive vehicle examples.
τ_{dist}	Distance threshold between two detections.

Table 3.1: Parameters and symbols used in Chapter 3.

3.2 Identifying the ROI

Perspective effect in traffic surveillance videos is illustrated in Figure 3.2. In the region that is closer to the surveillance camera, vehicles *appear* larger and better separated. This region is called the “entrance region” (note that vehicles may either enter *or* exit from this region). Moreover, projection of the 3D scene on the 2D image plane introduces significant occlusions and the entrance region is the part of the scene which gets least affected from this projection. Hence, we define the entrance region to be the ROI of a surveillance video in this context.



(a) Mecidiyekoy dataset. (b) Halic dataset. (c) Elmali dataset.

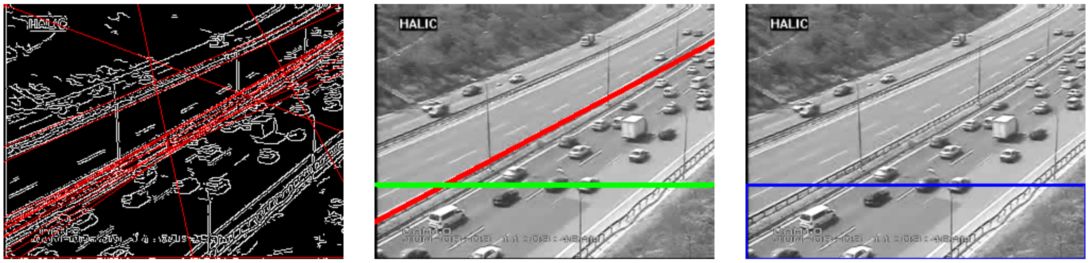
Figure 3.2: Example traffic surveillance videos.

The main road orientation in a highway surveillance video may be obtained by identifying the dominant lines in the video (see Figure 3.2). This is achieved in two steps: 1) Canny edge detection algorithm [3] is used to get the edge map of a frame in the video and 2) Hough line transform [7] is applied to the edge map to identify the dominant lines in the frame.

These dominant lines tend to lie on the sides and in the middle of the highway, since these two regions have long edges that are not occluded by vehicles. With this observation, the line with the median orientation is used in conjunction with a predefined upper limit to set the boundaries of the ROI. This process is illustrated in Figures 3.3, 3.4 and 3.5. This is a robust method for straight highways since the number of cars in the scene do not affect the final results.

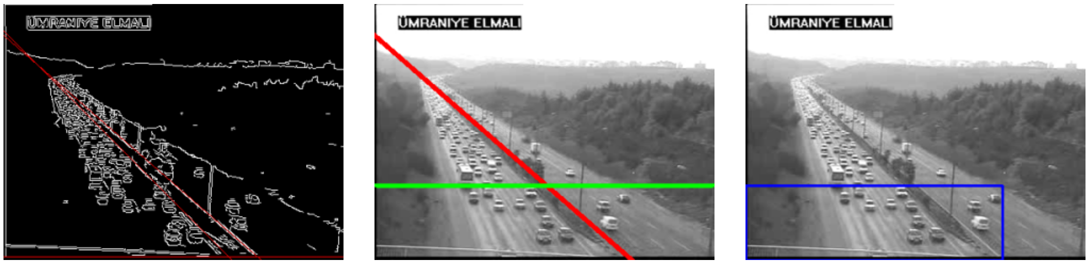
3.3 Background Subtraction

As Figure 3.1 shows, the quality of the background subtraction affects both the training and the detection phases in terms of computational costs and accuracy.



(a) Dominant lines (in red). (b) Road orientation (in red) and upper limit (in green). (c) ROI (in blue).

Figure 3.3: Determining the ROI in Halic dataset (Best viewed in color).



(a) Dominant lines (in red). (b) Road orientation (in red) and upper limit (in green). (c) ROI (in blue).

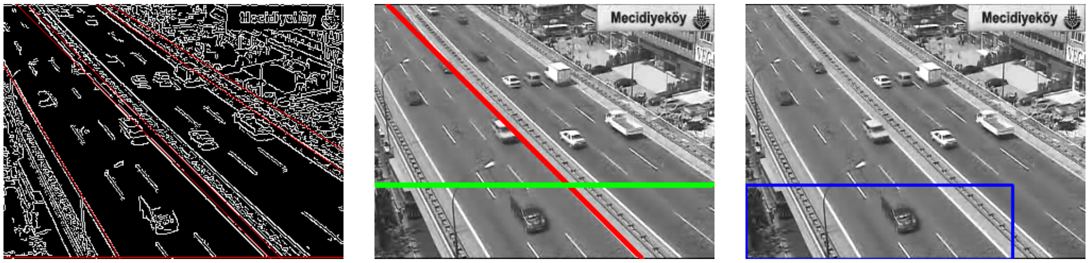
Figure 3.4: Determining the ROI in Elmali dataset (Best viewed in color).

In this work we employed an enhanced version [10] of the adaptive background mixture model [23].

In the adaptive background mixture model, the recent history of each pixel is modeled by a mixture of K Gaussian distributions, where history of a pixel is a time series of pixel values seen in that pixel (x_0, y_0) up to time t: $\{X_1, \ldots, X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\}$. In this model, the probability of observing the current pixel value becomes:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{3.1}$$

where K is the number of distributions and η is a Gaussian probability density



(a) Dominant lines (in red). (b) Road orientation (in red) and upper limit (in green). (c) ROI (in blue).

Figure 3.5: Determining the ROI in Mecidiyekoy dataset (Best viewed in color).

function. $\omega_{i,t}$, $\mu_{i,t}$ and $\Sigma_{i,t}$ are the estimated weight, mean and covariance matrix, respectively, of the i^{th} Gaussian at time t.

Each new pixel value X_t is tried to be matched to one of these K Gaussian distributions. If it cannot be matched to any of the existing distributions, the distribution with the least weight is replaced with a new one. Distribution means, covariance matrices and weights are updated accordingly.

For each pixel, B distributions with the most supporting evidence and least variance are chosen as the background model. If the new pixel value X_t is matched to one of these distributions, it is considered to be a background pixel. If not, the pixel is labeled as a foreground pixel. Then a two-pass connected component analysis is used to form the foreground blobs.

In the enhanced background subtraction method, Jun et al. [10] applies a 3D (time being the 3^{rd} dimension) connected component analysis on the foreground masks obtained from the original method of Stauffer and Grimson [23]. This 3D connected component analysis incorporates both spatial and temporal information in the background model, and hence results in extremely good foreground masks. Additionally, the background model is capable of "adapting" itself to the changing lighting and weather conditions, as well as the day/night changes. Figure 3.6 illustrates a sample input and the foreground mask obtained from that input.

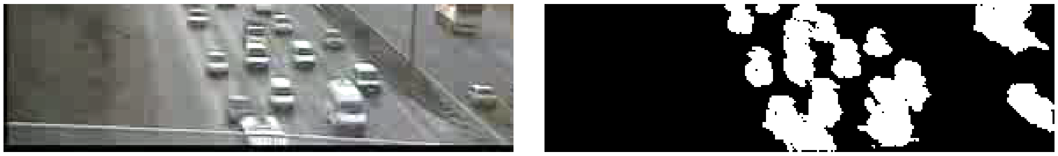

(a) Input frame.
(b) Foreground mask.

Figure 3.6: Results of the enhanced adaptive background mixture model.

3.4 Training Phase

Independent of the classifier one may use, every training algorithm has two major step: 1) forming positive and negative training sets and 2) feature extraction. In many applications the former step requires manual labeling which is a costly operation. In Section 3.4.1 we explain how to obtain these samples automatically from the video sequences *without* human supervision. The latter step of training also has a significant effect on the accuracy of the trained classifier. The employed feature extraction method is explained in Section 3.4.2.

3.4.1 Forming the Training Sets

Figure 3.6 illustrates that foreground masks have blobs of varying sizes. One key observation is that highway surveillance videos are usually captured from a distance, so vehicles in the same video have similar sizes (exceptions are long buses and trucks). Hence, the area of a blob which contains a single vehicle is significantly different from the area of a blob that contains two or more vehicles. Consequently a good area threshold can be used to separate these two sets of blobs.

Figure 3.7 shows the blob areas seen in four different datasets. As the figure illustrates, it is relatively easy to determine where the blob areas show a steep change. Therefore the area threshold (τ_{area}) can be selected from the range where this change is not significant. Additionally, since this is just the training phase, a "safe" threshold may be chosen, which would *not* capture all existing single vehicle blobs. Thresholds used for two different datasets is shown in the figure.

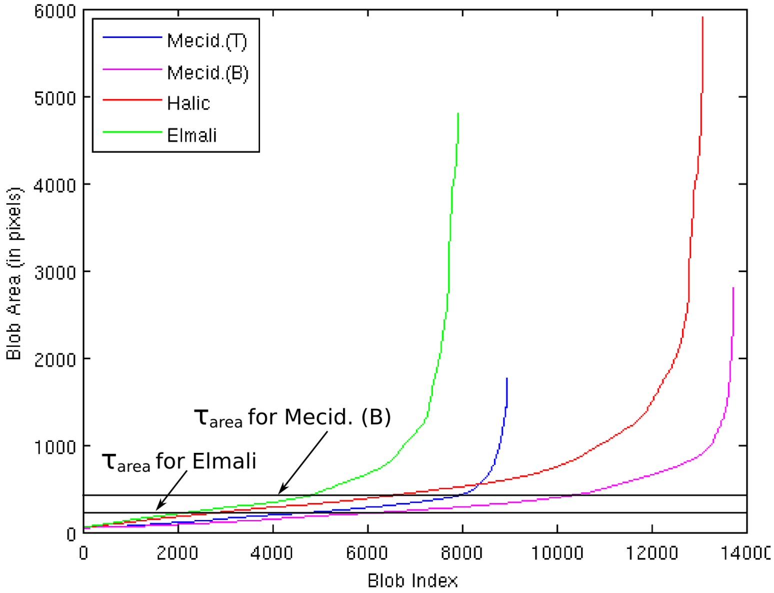


Figure 3.7: Blob areas in four datasets. Blobs seen in a dataset are sorted by area. The area thresholds used for Elmali and Mecidiyekoy (B) datasets are also shown.

With these observations, patches corresponding to "small" blobs are stored as positive examples. Figures 3.9(a) and 3.9(c) illustrate some automatically captured positive training examples from the Elmali and Halic datasets explained in Section 3.6. Experiments showed that almost 100% of all generated positive examples contain a single, centered vehicle, when safe area thresholds are used.

Negative examples of the training are generated using the locations of the positive examples. All patches centered at the corners and the edge midpoints of a positive example boundary are taken as negative examples as illustrated in Figure 3.8. There are two primary motivations behind this unusual definition of negative examples: 1) This way it is *guaranteed* that these are "real" negatives. However, if the negatives are selected randomly, a selected patch may end up having a single vehicle in the center. 2) The generated negative examples are "hard" negatives which is better for the training of the classifier (illustrated in Figures 3.9(b) and 3.9(d)). With these hard negatives the classifier is forced to learn structured clutter, rather than just plain background, which is beneficial for accurate localization of vehicles in blobs that are suspected to have more than one vehicle.

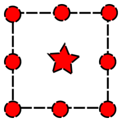
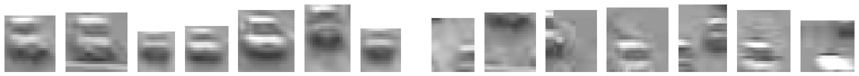
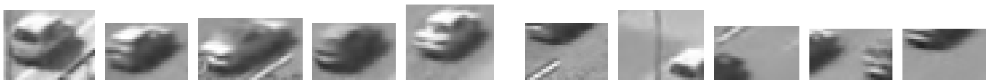


Figure 3.8: Relative positive and negative example centers. All patches centered at the corners and the edge midpoints of a positive example boundary are taken as negative examples.



(a) Elmali positive examples. (b) Elmali negative examples.



(c) Halic positive examples. (d) Halic negative examples.

Figure 3.9: Positive and negative training examples from Elmali and Halic datasets.

This strategy for automatically identifying the positive and negative examples works only if there are at least some number of well-separated vehicles in the video during the training phase. This is a reasonable assumption since in the ROI vehicles are relatively better separated and the training can be done when the highway is not extremely crowded.

Considering the number of existing traffic surveillance systems, and the fact that each system consists of a large network of cameras, this unsupervised method is crucial. On the other hand, if available, a human operator can still monitor the quality of the generated positive and negative examples and give feedback about the area threshold.

3.4.2 Feature Extraction

Color and texture based features are not discriminative enough to separate the positive examples from the negative ones (see Figure 3.9). On the other hand, positive and negative examples have significantly different line distributions and orientations. For this reason a histogram of oriented gradients (HOG) based method, similar to [15] and [6], is used to extract features from the patches.

First each patch is divided into four cells by a 2×2 grid. Then for each cell, an 8-bin HOG is computed. These histograms are normalized with respect to the number of pixels in each cell. And finally, these 8-bin histograms are concatenated and a support vector machine (SVM) is trained by the resulting 32-dimensional feature vectors.

This method has two advantages: 1) Since the training examples are automatically generated from the background subtraction, they have various sizes. This method does not require any resizing or alignment, which is important since these operations may not be accurate in this uncontrolled setting. 2) By using a 2×2 grid, partial spatial information is also captured. In this way the relative location of the edges is also used, as well as the edge orientations, to build a binary vehicle classifier.

3.5 Detection Phase

In the detection phase, blobs obtained by the background subtraction are partitioned into two sets using the area threshold, τ_{area}. As mentioned in Section 3.4.1, blobs with relatively small areas are assumed to contain only a single vehicle. Hence, these vehicles can be detected directly by using the foreground masks. Contrary to this, some blobs are suspected to have two or more vehicles, so they need to be further processed.

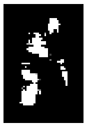
This is done with the help of the binary classifier generated in the training step (see Section 3.4). Given a patch descriptor, this classifier decides if the patch corresponds to a vehicle or not.

In order to detect the vehicles in these suspicious blobs, a sliding window approach is used. The window size is determined by the median width (w_s) and height (h_s) of the positive training examples of Section 3.4.1. As previously mentioned, highway surveillance videos are usually captured from a distance, so vehicles in the same video have similar sizes. This is why a fixed window size performs well on detecting a vast majority of vehicles (exceptions are long buses and trucks as explained in Section 3.6).

At each pixel location along the suspicious blob, a patch centered at that location is extracted. Then the 32-dimensional feature vector for that patch is computed and used in the binary classifier. This process is illustrated in Figure 3.10. Figure 3.10(a) shows a blob which is suspected to have more than one vehicle. The corresponding patch from the frame is shown in Figure 3.10(b), and Figure

3.10(c) illustrates the output of the pixel-wise sliding binary classifier. Each pixel in this binary output indicates whether there is a vehicle centered at that pixel or not.

(a) Suspicious blob. (b) Corresponding patch.



(c) Sliding window classifier output.

Figure 3.10: Processing a suspicious blob.

As Figure 3.10(c) suggests, results of the binary classifier should further be processed in order to eliminate some false positives (misleading detections that do not correspond to a vehicle). This is handled by the constraint that the distance between two detected vehicle centers cannot be less than a particular threshold (τ_{dist}). The system is relatively more sensitive to this threshold since it directly affects the number of false positives and false negatives (vehicles which are not detected). However, a good threshold can still be determined using the window size and the average single vehicle blob area (see Section 3.4.1). So, for every suspicious blob, results of the binary classifier are sorted in decreasing order of areas. Then each result center is compared with all other result centers with larger areas. If the Euclidean distance is less than the threshold τ_{dist}, this result is assumed to be a false positive and is eliminated.

3.6 Experimental Results

Experiments are performed on three different traffic surveillance videos taken in Istanbul, Turkey. The names Elmali, Halic and Mecidiyekoy correspond to the locations that the videos were taken. Elmali is a very low-quality video with 384×288 resolution. Halic and Mecidiyekoy are middle- and high-quality videos with 320×240 resolutions each. For Elmali and Halic, the ROIs are automatically determined. For Mecidiyekoy, the region of interest did not contain any suspicious blobs (due to viewpoint, quality and amount of clutter). Hence no region of interest is determined for Mecidiyekoy, and instead the frames are directly divided into two regions, top (T) and bottom (B). Experiments are done independently on these two regions.

For background subtraction, the MATLAB code provided by [10] is used. SVM related operations are handled by using the publicly available SVM library, LIBSVM [4]. The rest of the system is implemented in C++, using the open-source computer vision library, OpenCV [1].

Dataset	Training		Test		Accuracy
	+	-	+	-	
Mecid. (T)	1001	1002	837	993	91.20%
Mecid. (B)	200	1501	135	1180	95.13%
Halic	700	7700	317	436	98.008%
Elmali	1100	11500	403	924	87.18%

Table 3.2: Training statistics. "+" and "-" are the number of positive and negative examples, respectively.

A linear kernel is used for SVM training. Table 3.2 shows the number of positive and negative examples used in training and testing. All of these examples are generated automatically, and then partitioned randomly so that the training set contains relatively more elements. The classifiers are trained with the examples in the training set and then tested with the examples in the test test. In each experiment, the system extracted hundreds of positive and negative examples by processing less than a minute of the input video. For Mecidiyekoy, only 600 frames ($\sim$ 20 seconds) were enough to generate the required number of examples. Classification accuracies on the test tests ranged from 87% to an impressive 98% for the Halic dataset. The ROC curves of the four videos are illustrated in Figure 3.11. These curves are obtained by varying the discrimination threshold between positive and negative classes. Figure 3.11 shows that with less than 10% false positive rates, more than 90% detection accuracies can be achieved with these video-specific classifiers.

Dataset	Frames	Vehicles (GT)	Vehicles (D)	Acc.	FP	FN
Mecid. (T)	100	659	633	96.05%	24	26
Mecid. (B)	100	335	313	93.43%	18	22
Halic	100	487	442	90.75%	33	45
Elmali	100	1197	1106	92.39%	85	91

Table 3.3: Frame-by-frame detection results. “GT” stands for “Ground Truth”, “D” stands for “Detected”, “Acc.” stands for “Detection Accuracy”, “FP” stands for “False Positives” and “FN” stands for “False Negatives”.

Frame-by-frame detection results are presented in Table 3.3 and Figures 3.12, 3.13, 3.15 and 3.14. From each dataset, 100 frames (with 10 frame periods to avoid including very similar frames) are manually inspected. The total number of vehicles, detections, false positives and false negatives are counted. As Table 3.3 shows, the proposed system has very few false positives and slightly more false negatives. Hence, extremely good detection accuracies, ranging between 90% to 96%, could be achieved in these three significantly different and challenging highway surveillance videos.

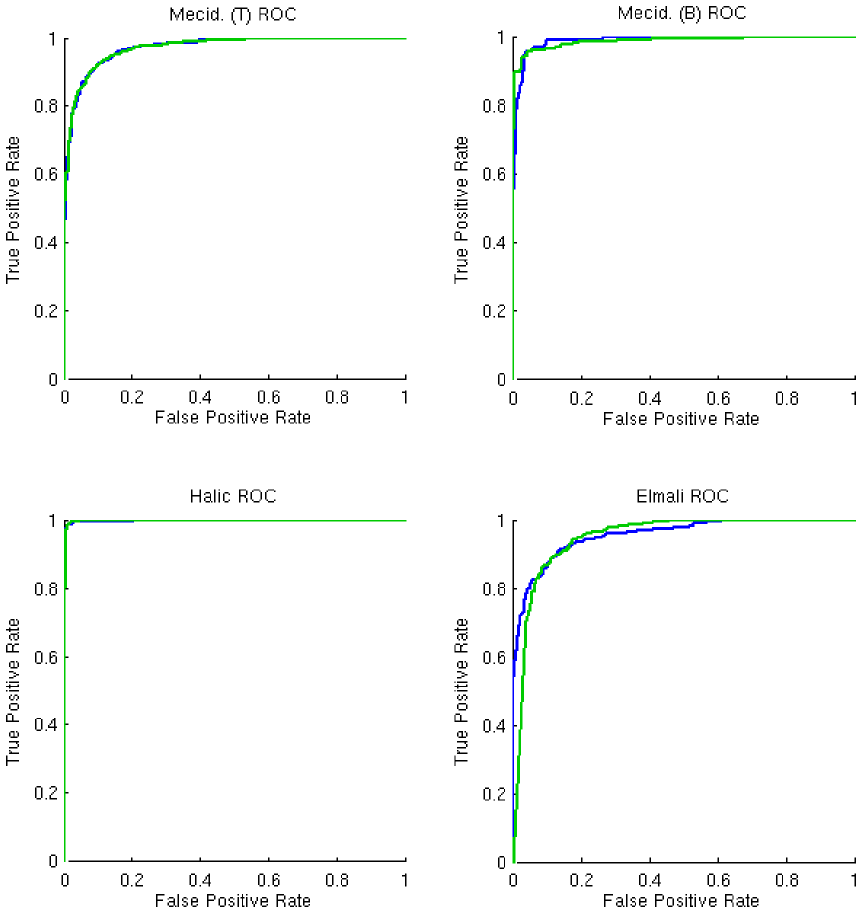


Figure 3.11: ROC curves for the classifiers.

For comparison, performance of a simple background subtraction only approach is presented in Table 3.4. In this baseline algorithm, every individual blob is considered to be a detected vehicle. Table 3.4 shows that as the amount of clutter in the video increases, the detection accuracies decrease dramatically with this baseline algorithm. On the other hand, our detection algorithm identifies the suspicious blobs and further processes them to achieve very good results. Note that different 100 frames are used in two sets of experiments.

It is worthwhile to emphasize the results of the Mecidiyekoy experiments. As

Dataset	Frames	Vehicles (GT)	Vehicles (D)	Acc.	FN
Mecid. (T)	100	409	280	68.45%	129
Mecid. (B)	100	234	204	87.17%	30
Halic	100	488	331	67.82%	157
Elmali	100	990	564	56.96%	426

Table 3.4: Frame-by-frame detection results for the baseline algorithm. "GT" stands for "Ground Truth", "D" stands for "Detected", "Acc." stands for "Detection Accuracy" and "FN" stands for "False Negatives".

Figures 3.12 and 3.13 show, when the video is divided into two regions, very accurate detection results could be achieved even though the vehicles in two regions look different. This is because we had trained one classifier for each region independent from the other region.

Figure 3.16 shows some shortcomings of our detection algorithm. The major shortcoming is the poor detection of large vehicles. This can be explained by the following; these vehicles look very different, training is done only with average-sized vehicles and the detection window is too small for these vehicles. Other minor shortcomings are shadow-related localization problems and some false positives which could not be eliminated by the method explained in Section 3.5.

3.7 Discussion

We presented a novel approach for vehicle detection in highway surveillance videos. The main contribution is the underlying unsupervised learning framework where the vehicles in a video are directly learned from the video without any prior knowledge or supervision.

With this framework, we achieved very good detection results in three videos that are significantly different in terms of quality, viewpoint and clutter.

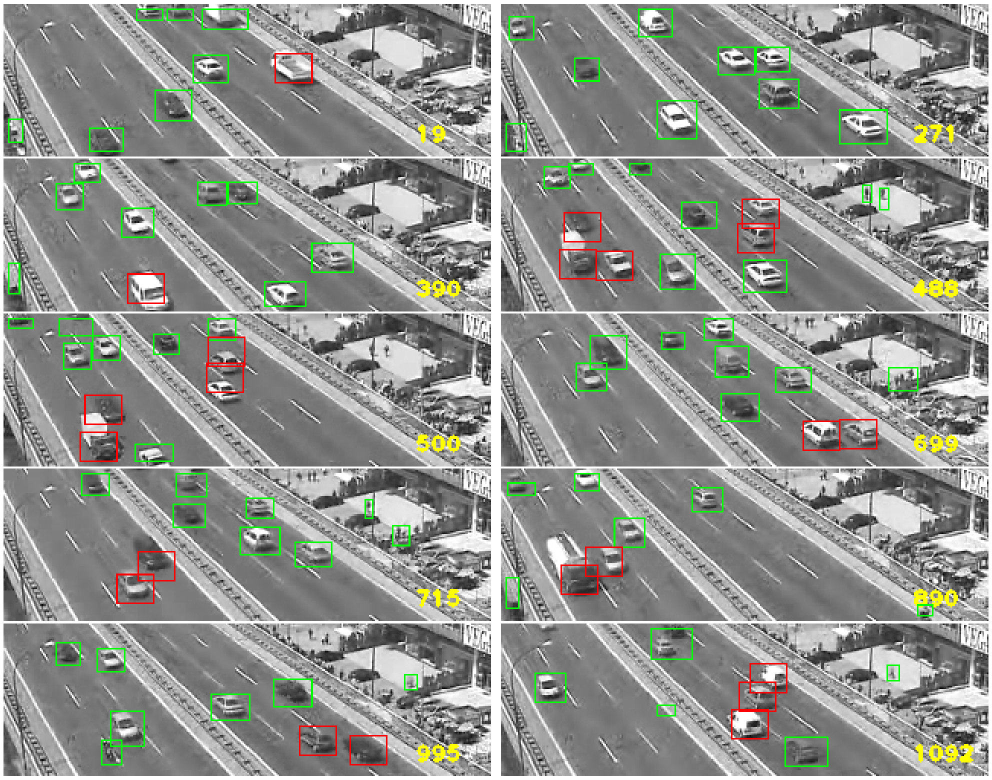


Figure 3.12: Good detection results in Mecidiyekoy (T) dataset. Colors represent; green: detection directly from background subtraction and red: detection by processing suspicious blobs (Best viewed in color).

By the Mecidiyekoy experiments (see Figures 3.12 and 3.13), we showed that this method can also be applied to complex highway surveillance videos, as long as the video can be partitioned into several regions. The only requirement is that the vehicles in a region should "appear" similar.

In Chapter 4 we explain a simple, but effective tracking algorithm that is used to link the detections in consecutive frames. It is shown that this tracking approach further improves the frame-by-frame results.

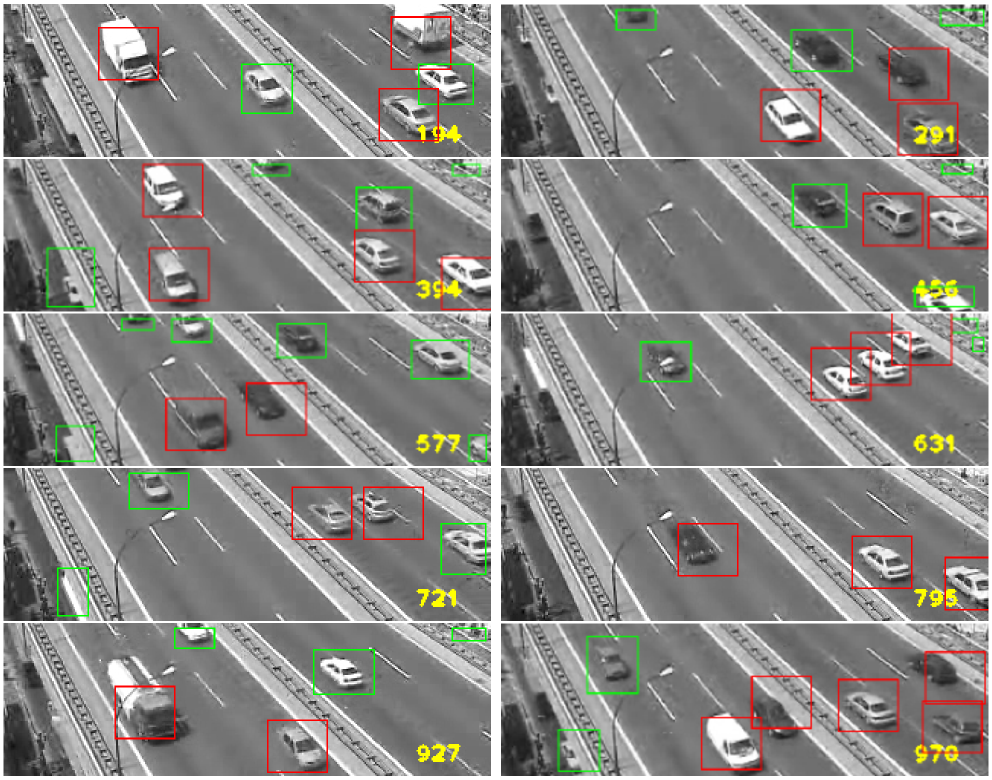


Figure 3.13: Good detection results in Mecidiyekoy (B) dataset. Colors represent; green: detection directly from background subtraction and red: detection by processing suspicious blobs (Best viewed in color).

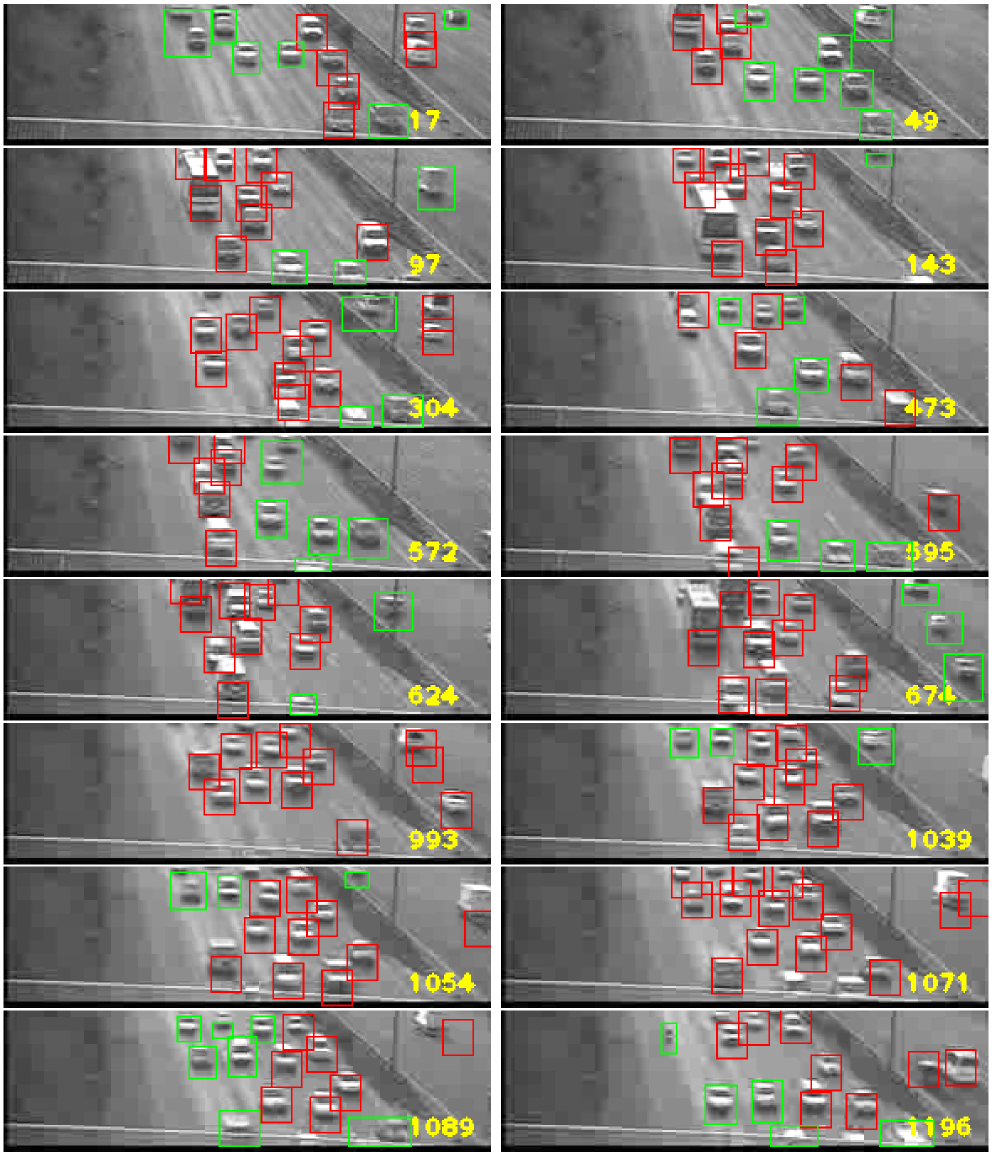


Figure 3.14: Good detection results in Elmali dataset. Colors represent; green: detection directly from background subtraction and red: detection by processing suspicious blobs (Best viewed in color).

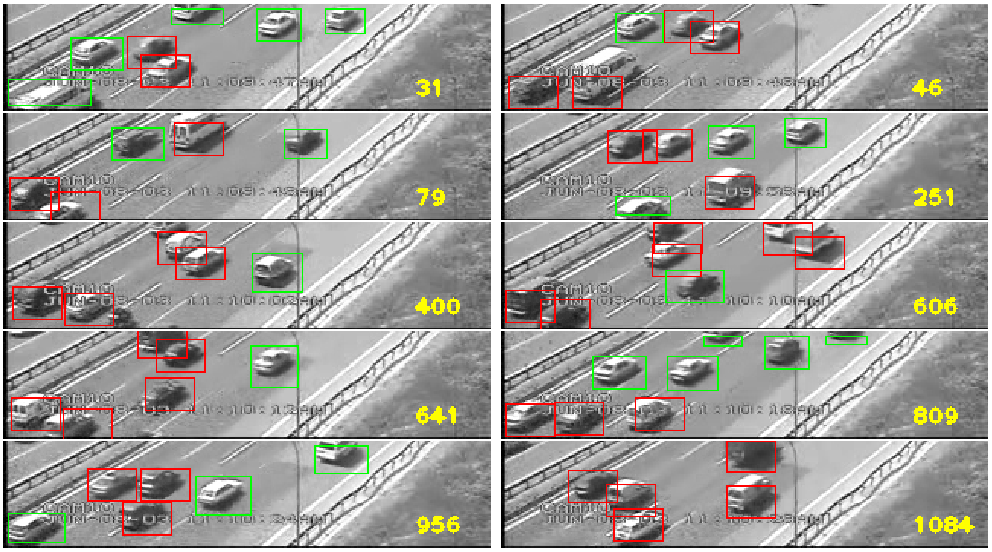


Figure 3.15: Good detection results in Halic dataset. Colors represent; green: detection directly from background subtraction and red: detection by processing suspicious blobs (Best viewed in color).

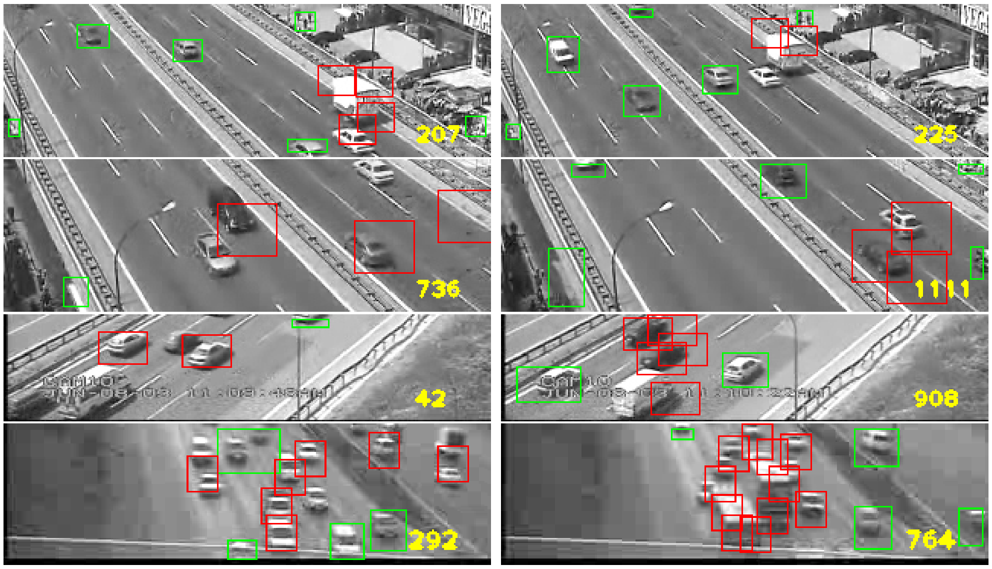


Figure 3.16: Bad detection results. The major shortcoming is the poor detection of large vehicles. This is mainly because the detection window is too small for these vehicles. Shadow-related localization problems is another minor shortcoming.

Chapter 4

Vehicle Tracking

As explained in Chapter 3 we achieved very good detection accuracies. However, these frame-by-frame results are not enough to reveal the important traffic parameters such as speed, flow and concentration. In order to compute these parameters, vehicles in the scene must be detected *and* tracked simultaneously. Moreover, tracking may also be used to improve the frame-by-frame detection results.

Various object tracking approaches have been discussed in Chapter 2. These approaches range from basic linear dynamic models [7] to more complex probabilistic multi-hypotheses methods [14][21].

Since highway scenes are extremely restricted environments, tracking in this environment does not require complicated methods that would be required in other tracking applications such as pedestrian tracking. On the other hand frame-by-frame detections are not perfect. So, a straightforward first-order Markovian approach would not be sufficient either.

In this work we developed an efficient and effective tracking algorithm based on having an over-complete set of tracks at any given time during the processing. In Section 4.2 the basics of this approach is explained. In Section 4.3 the benefits of the tracking on frame-by-frame detection results is discussed. Sections 4.4 and 4.5 explains merging and localization of tracks. In Section 4.6 the experimental results are presented and the chapter is concluded in Section 4.7.

4.1 Symbols and Definitions

Symbol	Definition
f_n	Video frame at time t_n.
r_n, i	One observed detection in frame f_n.
$\mathcal{T}$	Over-complete set of tracks.
$\mathcal{T}'$	Tracks in $\mathcal{T}$ after propagation.
T_t	A new track.
T_e	An existing track, $T_e \in \mathcal{T}$.
w_s	Median width of the positive vehicle examples.
h_s	Median height of the positive vehicle examples.
κ_1	Parameter for the observation window width.
f_c	Current frame, which is being processed.
f_{start}	The first frame in the observation window.
f_{end}	The last frame in the observation window.
τ_{match}	Distance threshold for matching a detection to a track.

Table 4.1: Parameters and symbols used in Chapter 4.

4.2 Creating an Over-complete Set of Tracks

Figure 4.1 presents an overview of our tracking algorithm. Before proceeding with the explanation, a few terms are defined:

observed detection - Every detection returned by the detection algorithm explained in Chapter 3 is considered to be an "observed detection (OD)". Note that any observed detection can actually be a true positive (TP) or a false positive (FP).

hypothesized detection - Any detection that is *generated* by the tracking algorithm is defined to be an "hypothesized detection (HD)".

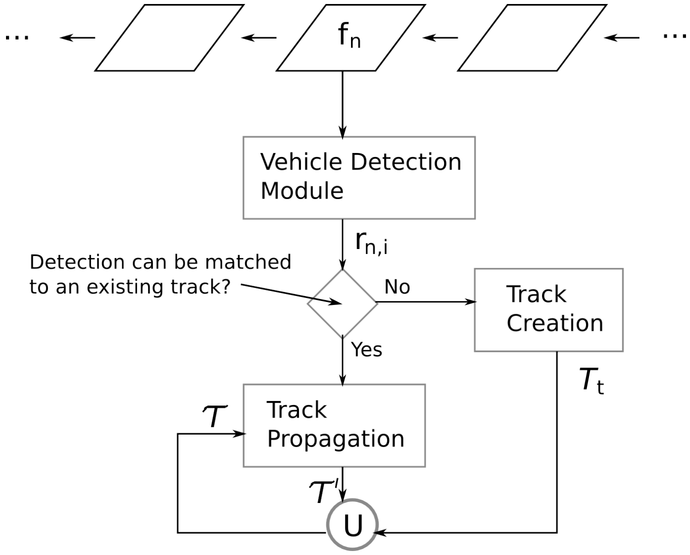


Figure 4.1: Vehicle tracking system overview.

track - A "track" is defined to be a list of observed and hypothesized detections. A track can either be valid or invalid depending on its score.

observation window - The "observation window" is the $2\kappa_1 + 1$ frames examined to create the over-complete set of tracks. f_{start} and f_{end} are the first and the last frames of this window.

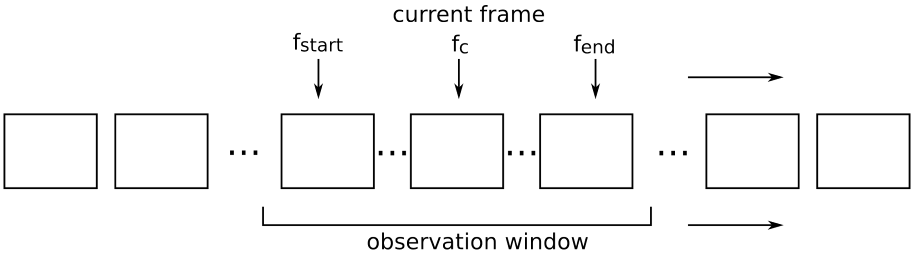
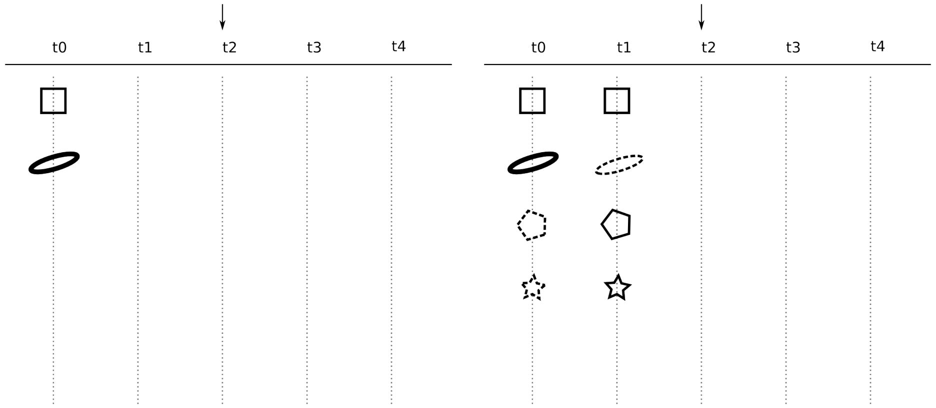
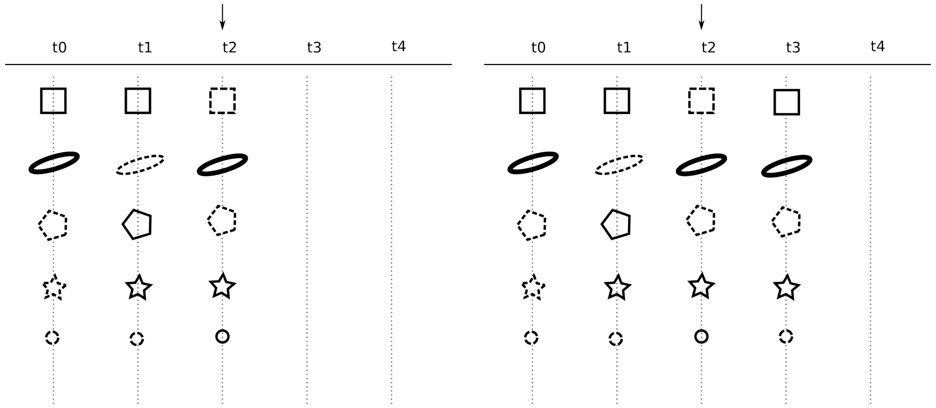


Figure 4.2: Observation window illustration.

The over-complete set of tracks is then created as follows:



(a) Tracks at time $t0$. (b) Tracks at time $t1$.



(c) Tracks at time $t2$. (d) Tracks at time $t3$.

Figure 4.3: Creating an over-complete set of tracks. Solid shapes represent observed detections, dotted shapes represent hypothesized detections. Columns correspond to frames and rows correspond to tracks. Tracks are initiated with the ODs of the first frame. Starting from the next frame, the ODs in all the following frames are either matched to the existing tracks or used to create new tracks. Whenever a track cannot be propagated by an OD, a HD is created and matched to the track.

Since there are no tracks to propagate at the beginning of the procedure, all ODs in frame f_{start} are used to initiate new tracks. Starting from the next frame, the ODs in all the following frames are either matched to the existing tracks or used to create new tracks.

For every existing track, the position in the next frame is predicted using a constant speed model. Then the OD in the next frame, which is closest to this predicted position, is identified. If the Euclidean distance between the detection and the prediction is less than a threshold (τ_{match}), the detection is added to the track. This threshold is determined using the median positive example dimensions w_s and h_s. If an OD is matched to a track, then this detection is removed from the list of ODs for that frame. This ensures that the same OD is *not* assigned to multiple tracks.

After the match making process, remaining tracks are propagated using HDs. These HDs are generated at the predicted positions. Similarly, unmatched ODs are used to create new tracks and these tracks are backward-propagated with HDs until frame f_{start}. Track propagation and creation is illustrated in Figure 4.3. In this figure, solid shapes on each column represent the ODs in the corresponding frame, dotted shapes represent the HDs, and rows represent the tracks.

After the above procedure is repeated for every frame between f_{start} and f_{end}, the system has an over-complete set of tracks, where every OD within the observation window is considered. Moreover, all of the tracks have the same cardinality, $|T_e| = 2\kappa_1 + 1$.

This over-complete set of tracks is maintained as the observation window proceeds. As new frames are processed, oldest track elements are removed from the existing tracks and new elements are added using new ODs or HDs. Also, new tracks are added to the set whenever an OD cannot be matched to an existing track. The tracks are removed from the set $\mathcal{T}$, only when all of their elements are outside the ROI.

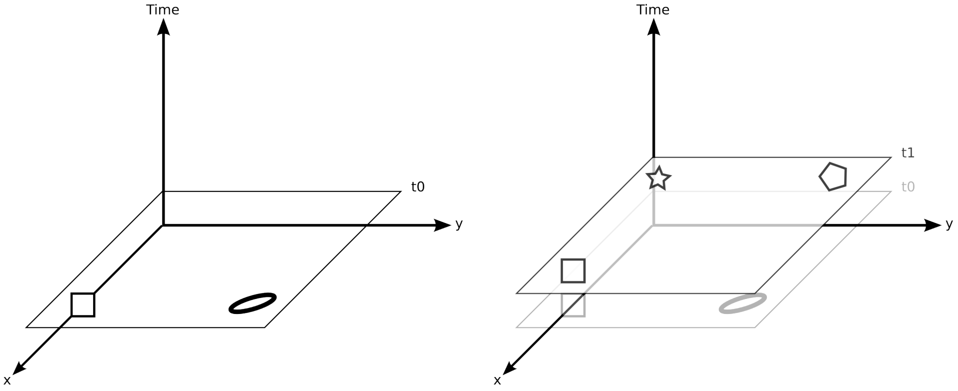
The motivation behind this tracking algorithm is illustrated in Figure 4.4. Frame-by-frame detection results may include FPs and FNs as well as the TPs (see Figures 4.4(a) through 4.4(d)). These FPs and FNs in a frame can only be identified when a set of frames within a time interval of this frame are examined collectively (see Figure 4.4(e)).

4.3 Improving the Vehicle Detection Results

Once the global set of tracks $\mathcal{T}$ is generated, each track is scored with respect to its elements. In our experiments a very simple scoring function is employed. In this function, each track starts with a score of 0. Then for every OD in the track, the score is increased by a value determined by the number of consecutive ODs up to this one. Every HD decreases the score by 1. Finally, only the tracks with positive scores are considered to be "valid" tracks *for the currently processed frame.*

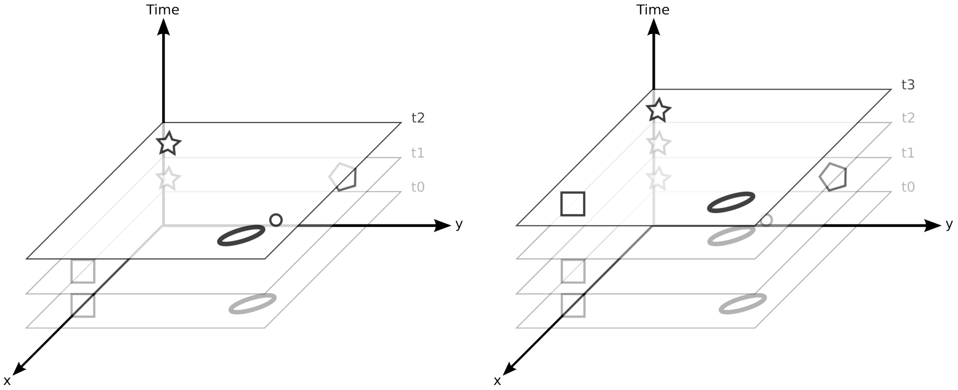
This method easily identifies the FPs and the FNs in a frame. In a track created due to a FP detection in the observation window, the majority of the elements are HDs. Hence, this track would have a negative score and be eliminated (see Figure 4.3(d), 3^{rd} and 5^{th} rows). Similarly, FNs are the HDs of tracks with high scores. Even though, the detection algorithm misses these detections, they can be re-generated by the tracking algorithm (see Figure 4.3(d), 1^{st}, 2^{nd} and 4^{th} rows).

The success of our tracking algorithm in identifying FPs and FNs is illustrated in Figure 4.5. Figure 4.5(a) shows the results obtained when only the detection algorithm explained in Chapter 3 is used. As the figure illustrates there is one FP and one FN in this frame. Both of these could be identified and fixed by the help of this tracking algorithm (see Figure 4.5(b).



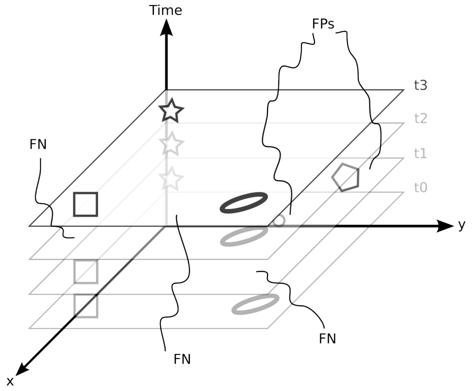
(a) Detections at time $t0$.

(b) Detections at time $t1$.



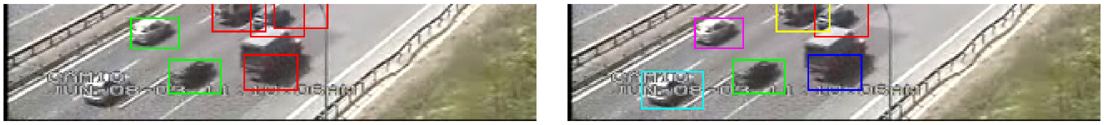
(c) Detections at time $t2$.

(d) Detections at time $t3$.



(e) Identified FPs and FNs.

Figure 4.4: Motivation behind the tracking algorithm. Shapes represent the ODs in the corresponding frames. FPs and FNs in a frame can only be identified when a set of frames within a time interval of this frame are examined collectively.



(a) Detection results before tracking.

(b) Detection results after tracking.

Figure 4.5: Improving the frame-by-frame detection results using tracking.

4.4 Merging Tracks

The tracking algorithm explained in Section 4.2 is based on creating "continuous" tracks. Whenever a suitable OD cannot be located, another one is hypothesized and the track is propagated.

In this tracking algorithm, occasionally, one vehicle may end up having two or more tracks as illustrated in Figure 4.6. This is because each HD may introduce a small distortion in the track. For each new frame, as long as the track orientation cannot be fixed by an OD, this distortion will continue to increase.

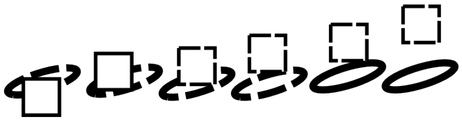


Figure 4.6: One vehicle having two tracks. Solid shapes represent real detections, dotted shapes represent imaginary detections.

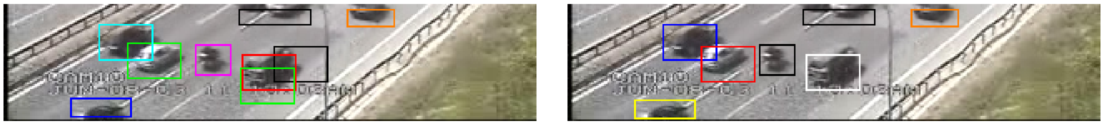
Two key observations help to locate these situations where two tracks need to be merged: 1) Even in extreme cases, the overall distortion cannot be dramatic. Hence, two tracks that need to be merged will have a significant frame-by-frame overlap. 2) If two tracks belong to the same vehicle, then they cannot have ODs for the same frame.

As the global set of tracks $\mathcal{T}$ is updated after each frame, track pairs satisfying the above properties are located and merged. Merging is done by replacing the

HDs of the first track with the corresponding ODs of the second track. If there are no corresponding ODs in the second track, the HDs of the first track are kept unchanged.

In the merging step it is important to mention some special cases. Two negative score tracks should not be merged because this merge may end up validating two FPs in the observation window. Another special case arises when two tracks have a significant overall overlapping, but at the same have a few conflicting ODs. In this case these tracks should be merged since these conflicts are most probably caused by some FPs.

Figure 4.7 illustrates the benefits of this merging step. In Figure 4.7(a), a single vehicle has three tracks associated with it. With the help of the merging step, these three tracks could be identified and merged together as illustrated in Figure 4.7(b).



(a) Detection results before merging.

(b) Detection results after merging.

Figure 4.7: Eliminating FP tracks using merging.

4.5 Backward and Forward Localization

The last step in the tracking algorithm is backward and forward localization. Once the tracks are computed, appropriately merged and scored, then we can further enhance the localization of the detections in a valid track.

In order to do so, vehicle specific average speeds are calculated using the frame-by-frame detections in a track. Then the first pair of consecutive ODs, with speed values within an acceptable range of the calculated average speeds, is located. Starting from this pair, the locations of all observed and hypothesized detections are fixed according to the average vehicle speed values. Note that all the speed values mentioned in this chapter are given in pixels/frame.

In the backward localization, if the previous detection is not close enough to the predicted position (τ_{match} is used as the threshold), then this detection is moved accordingly. Similarly in the forward localization the same procedure is applied to the next detection in the track. Figure 4.8 illustrates this procedure and Figure 4.9 shows its benefits. Note that this step also helps to eliminate some more FP tracks since it "fixes" the distortions in the track orientations and hence makes it easier for the merging step to identify these FP tracks.

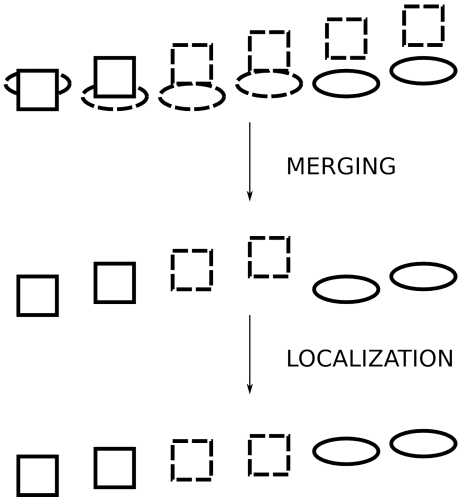
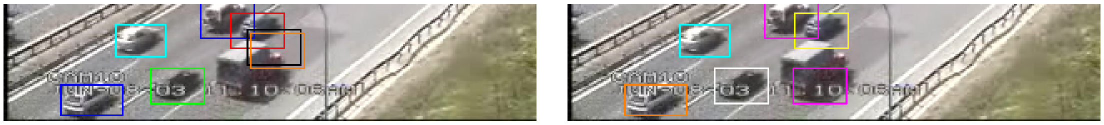


Figure 4.8: Backward and forward localization. After the tracks are created, merged and scored, the localization of the detections in a valid track can further be improved. This is achieved by computing vehicle-specific average speeds and fixing the displacements within the track that do not agree with these speeds.



(a) Detection results before localization. (b) Detection results after localization.

Figure 4.9: Benefits of backward and forward localization.

4.6 Experimental Results

Tracking experiments are performed on the same three traffic surveillance videos mentioned in Section 3.6. Similar to the frame-by-frame detection experiments of that section, for Elmali and Halic the ROIs are automatically determined. For Mecidiyekoy, two non over-lapping regions (one from the top half of the frame, and the other from the bottom half of the frame) are selected as the ROIs (see Figure 4.10). These two ROIs are slightly different than the ones used in Section 3.6. Experiments are done independently on these two regions.

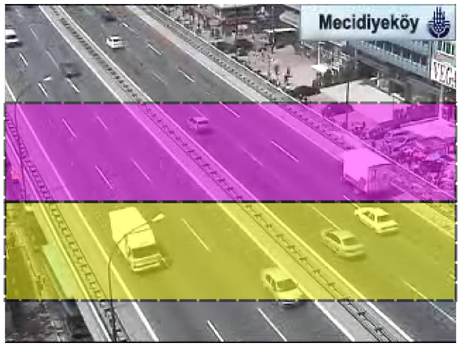


Figure 4.10: Manually selected ROIs for the Mecidiyekoy dataset.

The whole tracking algorithm is implemented in C++, using the open-source computer vision library, OpenCV [1].

In order to get the detection results for a frame, we first created (or updated if it already exists) the over-complete set of tracks (see Section 4.2). Then we merge the appropriate tracks (see Section 4.4) and then apply the backward and forward localization step (see Section 4.5). We achieved the best results when we apply *another* merging step after the localization. This second merge helped to eliminate a significant percentage of FPs.

In these experiments we used an observation window (see Section 4.2) of size $2\kappa_1 + 1 = 17$. Throughout the tracking, we maintained two average speeds, one for each main highway direction. We used these average speeds to predict the position of a track in the following frames.

In the merging step any two tracks with an overlapping area greater than

$$\frac{w_s * h_s * (2\kappa_1 + 1)}{3} \tag{4.1}$$

are merged even when they have conflicting ODs. This is because, it is very unlikely to have two separate vehicles in the ROI where one occludes, on average, 1/3 of the other for $2\kappa_1 + 1$ frames.

One particular detail in the localization step helped us to get very good tracks. In some cases, two consecutive detections in a track were very close, but the direction of the displacement between these two detections was conflicting with both of the two main highway directions. These detections are identified and irrespective of their closeness, the second detection is replaced with a properly placed one.

Frame-by-frame detection results obtained before and after tracking are presented in Tables 4.2 and 4.3. Note that *not* the same 100 frames are used in these two sets of experiments. Moreover, in Elmali and Halic videos, since the ROIs mostly contain only a single side of the highway, all the vehicles in the other side are ignored (contrary to the experiments of Section 3.6). In Mecidiyekoy (T) and Mecidiyekoy (B) experiments, vehicles on both sides are considered.

Dataset	Frames	Vehicles (GT)	Vehicles (D)	Acc.	FP	FN
Mecid. (T)	100	659	633	96.05%	24	26
Mecid. (B)	100	335	313	93.43%	18	22
Halic	100	487	442	90.75%	33	45
Elmali	100	1197	1106	92.39%	85	91

Table 4.2: Frame-by-frame detection results before tracking. "GT" stands for "Ground Truth", "D" stands for "Detected", "Acc." stands for "Detection Accuracy", "FP" stands for "False Positives" and "FN" stands for "False Negatives".

Dataset	Frames	Vehicles (GT)	Vehicles (D)	Acc.	FP	FN
Mecid. (T)	100	409	394	96.33%	25	15
Mecid. (B)	100	234	226	96.58%	16	8
Halic	100	488	460	94.26%	36	28
Elmali	100	990	920	92.92%	72	70

Table 4.3: Frame-by-frame detection results before tracking. "GT" stands for "Ground Truth", "D" stands for "Detected", "Acc." stands for "Detection Accuracy", "FP" stands for "False Positives" and "FN" stands for "False Negatives".

As Tables 4.2 and 4.3 show, with this tracking algorithm we could improve the detection results of Chapter 3. In two of these experiments the improvement was significant. Interestingly the FP rate of each experiment was approximately 7%. And even with these small FP rates extremely good detection accuracies could be achieved. Figure 4.11 compares some frame-by-frame detection results obtained before and after tracking.

Tables 4.2, 4.3 and Figure 4.11 focus on the impact of tracking on frame-by-frame detection results. The tracking quality is evaluated in Chapter 5.

4.7 Discussion

In this chapter we presented a simple, but effective approach for tracking vehicles in a highway surveillance video. This approach is designed to identify the FPs and FNs in frame-by-frame detection results as the tracking proceeds.

The simplicity of the current approach is a result of the restrictions imposed by the environment. However, a major advantage of this tracking framework is that it can easily be extended to meet the requirements of more complex tracking scenarios.

The criteria used in track propagation is an example. It is explained in Section 4.2 that tracks are propagated by using the Euclidean distance as the matching criteria. In an other application, such as indoor pedestrian tracking, the surveillance videos will have higher resolutions and better qualities. Then the algorithm can make use of the visual correlation as well as the Euclidean distance in the propagation process.

Similarly, depending on the application, a probabilistic track scoring function can be used instead of the simple method explained in Section 4.3.

In Chapter 5 we explain how to obtain some important traffic parameters using this vehicle tracking algorithm. This, in fact, is a major application of vehicle detection and tracking in highway surveillance videos.

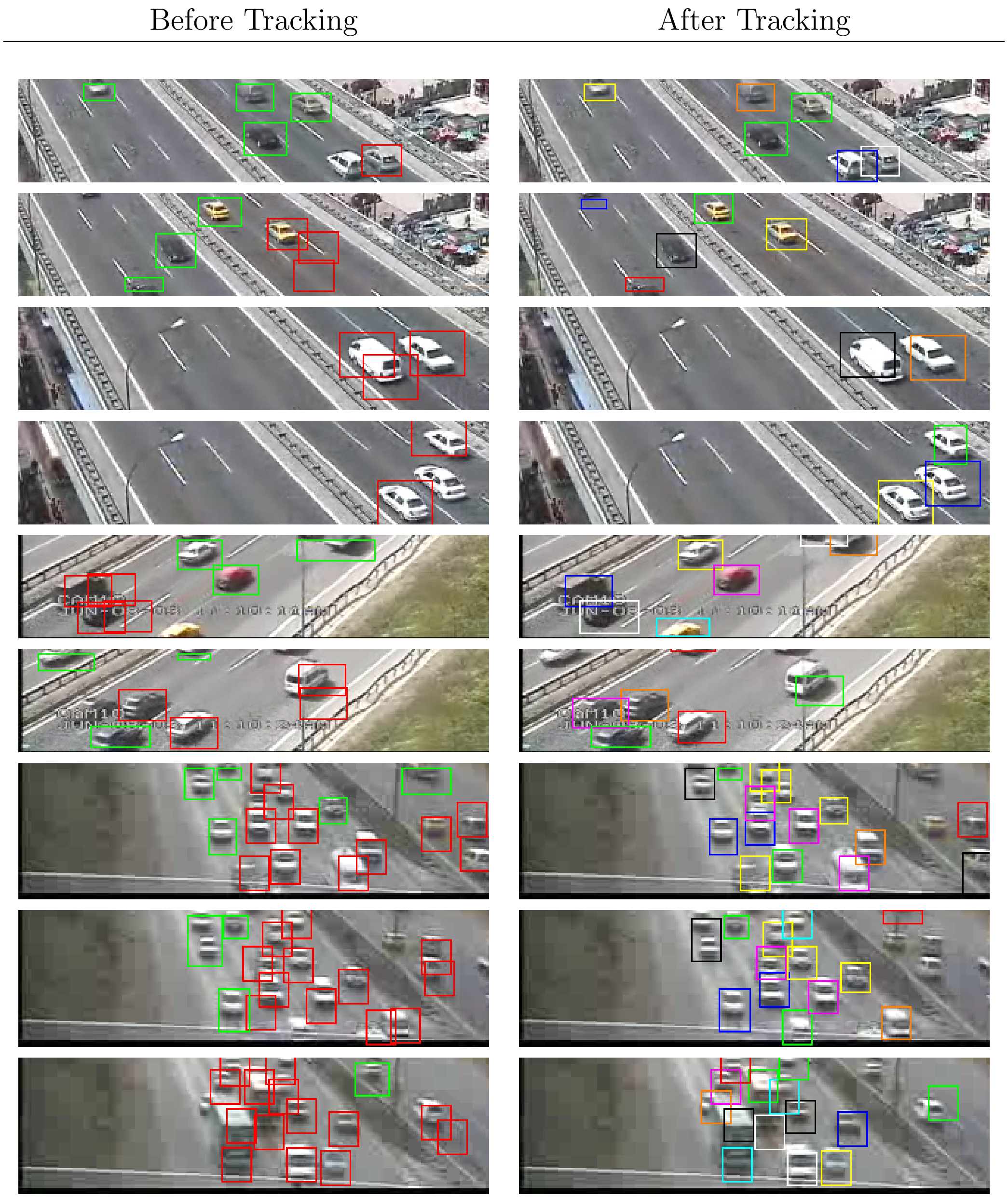


Figure 4.11: Detection results before and after tracking. The last row shows a major limitation of this tracking algorithm. Large vehicles cause many FP detections in consecutive frames. Hence these FPs cannot be identified.

Chapter 5

Calculating the Traffic Parameters

As mentioned in Chapter 1, the ultimate goal is to have fully automated traffic monitoring systems that can observe, reason and decide. In previous chapters "observing" a scene is explained. In this chapter the focus shifts to the "reasoning" step.

Calculating the traffic parameters and recognizing important activities are two of the most common applications of vehicle detection and tracking in highway surveillance videos. In the latter one, important activities such as illegal actions should be semantically defined, which is out of the scope of this work. Calculating the traffic parameters, on the other hand, can be achieved using the vehicle tracks obtained in Chapter 4.

The rest of the chapter is organized as follows: Section 5.2 briefly discusses the traffic flow theory and some important traffic parameters. Section 5.3 explains how to count the vehicles in a surveillance video using our tracking algorithm. Experimental results are presented in Section 5.4 and the chapter is concluded in Section 5.5.

5.1 Symbols and Definitions

Symbol	Definition
q	Flow.
N	Number of vehicles.
T	Duration of the observation.
$\bar{u}_t$	Time mean speed.
$\bar{u}_s$	Space mean speed.
$\tau_{vehicle}$	Threshold for identifying the tracks corresponding to vehicles.
C_{roi}	Approximate number of frames it takes for a vehicle to pass through the ROI.
κ_1	Parameter for the observation window width.
τ_{base}	Threshold used for counting in the baseline.

Table 5.1: Parameters and symbols used in Chapter 5.

5.2 Traffic Flow Theory and Important Traffic Parameters

Traffic flow theory is a tool that helps the researchers in many crucial ways. It makes it easier to understand complicated traffic behaviour by providing solid physical and mathematical explanations. Furthermore, it provides the necessary theoretic background "to design and operate streets and highways with the greatest possible efficiency" [8].

The three most important traffic parameters are flow, speed and concentration. These are defined as follows:

Flow - is the rate at which vehicles pass a given point in a highway:

$$q = \frac{N}{T} \tag{5.1}$$

where N is the number of vehicles, and T is the duration of the observation. Flow is usually expressed in vehicles/hour.

Speed - is defined in two ways:

$$\bar{u}_t = \frac{1}{N}\sum_{i=1}^{N} u_i \tag{5.2}$$

is the "time mean speed" with u_i being the speed of the i^{th} observed vehicle. This is simply the arithmetic mean of the vehicle speeds. Similarly, the "space mean speed" is the harmonic mean of the vehicle speeds and defined as:

$$\bar{u}_s = \frac{1}{\frac{1}{N}\sum_{i=1}^{N} u_i} \tag{5.3}$$

In traffic flow theory the latter one is used more often.

Concentration - is the number of vehicles per unit length.

Out of these three important parameters, calculating speed and concentration requires the real dimensions of the ROI. For the flow calculation only the number of vehicles is required. T in Equation 5.1 can easily be derived using the frame rate and the number of frames that are processed while the vehicles are being counted.

5.3 Counting the Vehicles

Our tracking algorithm maintains an over-complete set of tracks at any given time as explained in Section 4.2. The vehicle counting problem is then solved if the tracks that correspond to real vehicles can be identified within this over-complete set.

In Section 4.3, improving the frame-by-frame detection results using the over-complete set of tracks is explained. When identifying the detections in a frame, all tracks with positive scores are assumed to be valid tracks. This criteria is good enough for determining frame-by-frame detections, but it is not sufficient to identify the "real vehicle tracks" in the video.

Hence, for counting the vehicles, a track is considered to be a real vehicle only if it gets positive scores in a number of frames. A reasonable number for this threshold, $\tau_{vehicle}$, may be determined using the approximate number of frames it takes for a vehicle to pass through the ROI.

5.4 Experimental Results

Experiments are performed on the same three traffic surveillance videos mentioned in Sections 3.6 and 4.6. For the counting experiments only one region (B) from the Mecidiyekoy dataset is used.

A track that gets positive scores more than:

$$\tau_{vehicle} \cong \frac{C_{roi}}{2} \tag{5.4}$$

many times, where C_{roi} is the approximate number of frames it takes for a vehicle to pass through the ROI, is counted as a vehicle and all the other tracks are disregarded.

Dataset	NF	D	Ground Truth		Results		Error
			NV	Flow	NV	Flow	
Mecid. (B)	2250	90	256	10240	264	10560	3.12%
Halic	2250	90	235	9400	247	9980	5.10%
Elmali	2250	90	209	8360	220	8800	5.26%

Table 5.2: Vehicle counting results. “NF” stands for “Number of Frames”, “D” stands for “Duration” and “NV” stands for “Number of Vehicles”. Duration is given in seconds and flow is given in vehicles/hour.

Calculated number of vehicles and flow values are presented in Table 5.2. These numbers are obtained by running the tracking algorithm on a 90 second segment of each dataset. In this tracking algorithm, tracks are independently created, propagated and removed as the tracking proceeds. Hence, performance on a 90 second segment seems to be a good enough approximation of performance on the whole one hour of the surveillance video. Moreover, with these challenging videos, where the flow values are higher than 8300 vehicles/hour, there is a high cost associated with obtaining the ground truth of longer video segments.

These results may be interpreted as follows: 1) The speed-flow-concentration relationship is clearly seen. One may expect to get larger flow values for very crowded traffic scenes. However, concentration and speed are inversely proportional, so vehicles in a crowded scene will have relatively smaller speeds. In our experiments Elmali is the most crowded scene, but it has the smallest flow. 2) Error values less than 6% show that our vehicle detection and tracking algorithm is capable of producing very accurate results in significantly different and challenging videos.

The actual error in tracking may be slightly larger than the above error values since during the counting process some of the FN tracks may be cancelled by other FP tracks. However, the difference is expected to be very small since the tracks are formed using very accurate frame-by-frame detections as explained in Chapter 4.

We compare the performance of our tracking algorithm with a simple Kalman Filter based, first order Markovian approach in Table 5.3. In this baseline algorithm tracks are propagated using only ODs. Tracks with number of elements greater than a threshold (τ_{base}) are counted as vehicles. We tested the baseline with three different thresholds. Note that each dataset has a different value for C_{roi}.

Table 5.3 clearly shows that simple Kalman Filter based tracking algorithms cannot be employed in challenging applications such as vehicle counting. The primary reason is that these algorithms cannot recover from failure. A single FN is enough to divide a valid track into two invalid tracks. On the other hand, our approach can handle most of the FNs as well as the FPs.

Dataset	GT	Results							
		Our Method		$\tau_{base} = \kappa_1$		$\tau_{base} \cong C_{roi}/2$		$\tau_{base} \cong C_{roi}/4$	
		NV	Err	NV	Err	NV	Err	NV	Err
Mecid. (B)	256	264	3.12%	206	19.53%	198	22.65%	415	62.10%
Halic	235	247	5.10%	352	49.78%	163	30.63%	310	31.91%
Elmali	209	220	5.26%	699	234.44%	136	34.92%	393	88.03%

Table 5.3: Performance comparison with a baseline. “GT” stands for “Ground Truth”, “NV” stands for “Number of Vehicles” and “Err” stands for “Error”.

5.5 Discussion

In this chapter we showed a simple application of vehicle detection and tracking in highway surveillance videos. We counted the real vehicles in the over-complete set of tracks and then used this number to compute the traffic flow. If we had the required highway measurements, we would also be able to compute traffic speed and concentration values.

We achieved very good counting results. Error in the number of vehicles reported by our approach was less than 6% for all of the datasets.

Chapter 6

Conclusions and Future Work

We have developed a novel approach for vehicle detection and tracking in highway surveillance videos. In our detection module, we have shown how to build an unsupervised vehicle detection system that is robust against view-point, noise and low quality videos. We have also demonstrated that the restrictions imposed by the highways may be used to develop simplified multi-hypotheses tracking algorithms, which are very accurate and easily extendible.

We have tested our method in three traffic surveillance videos that are significantly different in terms of view-point, quality and clutter. Our algorithm has performed superbly in frame-by-frame vehicle detection, vehicle tracking and vehicle counting.

In the current approach binary classifiers used in vehicle detection are not adaptive. In future work we are planning to develop adaptive classifiers that will not require periodic training. Also, we will test the system with more videos over longer time periods. We are also interested in activity recognition in highway surveillance videos.

Bibliography

[1] *Open Source Computer Vision Library*, 2000. Software available at `http://sourceforge.net/projects/opencvlibrary/`.

[2] Margrit Betke, Esin Haritaoglu, and Larry S. Davis. Real-time multiple vehicle detection and tracking from a moving vehicle. *Machine Vision and Applications*, 12:69–83, 2000.

[3] J. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[4] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] Benjamin Coifman and David Beymer. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Res.: Part C*, 6(4):271–288, 1998.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. IEEE Conf. on Computer Vision and Pattern Recognition, 2005.

[7] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, 2003.

[8] Daniel L. Gerlough and Matthew J. Huber. *Traffic Flow Theory: A Monograph.* Transportation Research Board National Research Council, 1975.

[9] S. Gupte, O. Masoud, R. F. K. Martin, and N.P. Papanikolopoulos. Detection and classification of vehicles. *IEEE Trans. on Intelligent Transportation Systems*, 3:37–47, 2002.

[10] Goo Jun, J. K. Aggarwal, and Muhittin Gokmen. Tracking and segmentation of highway vehicles in cluttered and crowded scenes. IEEE Workshops on Applications of Computer Vision, 2008.

[11] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi. Traffic monitoring and accident detection at intersections. IEEE Conf. on Intelligent Transportation System, 1999.

[12] D. Koller, K. Daniilidis, and H. Nagel H. Model-based object tracking in monocular image sequences of road traffic scenes. *Int. J. of Computer Vision*, 10(3):257–281, 1993.

[13] Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. Proc. European Conf. on Computer Vision, 1994.

[14] Bastian Leibe, Konrad Schindler, Nico Cornelis, and Luc Van Gool. Coupled object detection and tracking from static cameras and moving vehicles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1683–1698, 2008.

[15] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. of Computer Vision*, 60:91–110, 2004.

[16] F. G. Meyer and P. Bouthemy. Region-based tracking using affine motion models in long image sequences. *CVGIP: Image Understanding*, 60(2):119–140, 1994.

[17] P. G. Michalopoulos. Vehicle detection video through image processing: the autoscopesystem. *IEEE Trans. on Vehicular Technology*, 40:21–29, 1991.

[18] B. Morris and M. Trivedi. Vector: Trajectory analysis for advanced highway monitoring. Presented at ITS America's Annual Meeting, 2009.

[19] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *Int. J. of Computer Vision*, 38:15–33, 2000.

[20] J. C. Rojas and J. D. Crisman. Vehicle detection in color images. IEEE Conf. on Intelligent Transportation System, 1997.

[21] Michael Ryoo and J. K. Aggarwal. Observe-and-explain: A new approach for multiple hypotheses tracking of humans and objects. IEEE Int. Conf. on Computer Vision and Pattern Recognition, 2008.

[22] Xuefeng Song and Ram Nevatia. A model-based vehicle segmentation method for tracking. IEEE Proceedings of the 10th Int. Conf. on Computer Vision, 2005.

[23] Chris Stauffer and W. E. .L. Grimson. Adaptive background mixture models for real-time tracking. IEEE Conf. on Computer Vision and Pattern Recognition, 1999.

[24] G. D. Sullivan. Model-based vision for traffic scenes using the ground-plane constraint. *in D. Terzopoulos and C. Brown (Eds): Real-time Computer Vision*, pages 93–115, 1994.

[25] Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection: A reviw. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28, 2006.

[26] Birgi Tamersoy and J. K. Aggarwal. Robust vehicle detection for tracking in highway surveillance videos using unsupervised learning. IEEE Int. Conf. on Advanced Video and Signal Based Surveillance, 2009.

[27] J. van Leuven, M. B. van Leeuwen, and F. C. A. Groen. Real-time vehicle tracking in image sequences. IEEE Instrumentation and Measurement Technology Conference, 2001.

Vita

Birgi Tamersoy was born in Izmir, Turkey on April 14, 1985, the only son of retired English teacher Çimen Tamersoy and retired computer engineer Mahmut Tamersoy. After graduating from American Collegiate Institute in June 2003, he entered the computer engineering program of Bilkent University, Ankara, where he was awarded with three merit scholarships in three consecutive years. He received his Bachelor of Science degree in May 2007. He started pursuing his graduate degree in August 2007, in the department of Electrical and Computer Engineering of the University of Texas at Austin. Since August 2008, he is a member of the Computer and Vision Research Center and working under the supervision of Dr. J. K. Aggarwal.

Permanent address: Armagan Sokak No:1 Sahilevleri Mahallesi
Narlidere Izmir, TURKEY 35320

This thesis was typeset with LaTeX[†] by the author.

[†] LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.