EFFICIENT VEHICLE TRACKING AND CLASSIFICATION FOR AN AUTOMATED TRAFFIC SURVEILLANCE SYSTEM

Amol Ambardekar, Mircea Nicolescu, and George Bebis
Department of Computer Science and Engineering
University of Nevada, Reno
U.S.A.
ambardek@cse.unr.edu, mircea@cse.unr.edu, bebis@cse.unr.edu

ABSTRACT
As digital cameras and powerful computers have become wide-spread, the number of applications using vision techniques has increased significantly. One such application that has received significant attention from the computer vision community is traffic surveillance. We propose a new traffic surveillance system that works without prior, explicit camera calibration, and has the ability to perform surveillance tasks in real time. Camera intrinsic parameters and its position with respect to the ground plane were derived using geometric primitives common to any traffic scene. We use optical flow and knowledge of camera parameters to detect the pose of a vehicle in the 3D world. This information is used in a model-based vehicle detection and classification technique employed by our traffic surveillance application. The object (vehicle) classification uses two new techniques − color contour based matching and gradient based matching. Our experiments on several real traffic video sequences demonstrate good results for our foreground object detection, tracking, vehicle detection and vehicle speed estimation approaches.

KEY WORDS
Computer vision, object tracking, traffic surveillance, vehicle detection, vehicle tracking, and vehicle classification.

1. Introduction

The rapidly increasing capacity of digital storage, computation power and the recent innovations in video compression standards lead to a strong growth of available video content. Digital cameras, which were novelty items in the 80's, have become ubiquitous in the last two decades. This has led to cheaper and better video surveillance systems. The video data stored by these systems needs to be analyzed, which is generally done by humans on a need-to-know basis (e.g., as a forensic tool after a bank robbery). This undermines the ability of video surveillance as a real time observer. Visual traffic surveillance has attracted significant interest in computer vision, because of its tremendous application prospect. A traffic surveillance system needs to detect vehicles and classify them if possible. Generating vehicle trajectories from video data is also an important application and can be used in analyzing traffic flow parameters. Information such as gap, headway, stopped-vehicle detection, speeding vehicle, and wrong-way vehicle alarms can be useful for intelligent transportation systems. Efficient and robust localization of vehicles from an image sequence (video) can lead to semantic results, such as "Car No. 3 stopped," "Car No. 4 is moving faster than car No. 6." However, such information can not be retrieved from image sequences as easily as humans do.

The rest of this paper is organized as follows. Section 2 gives an overview of our approach. Section 3 gives implementation details of our approach. Experimental results of the proposed technique are presented in section 4. Section 5 discusses the conclusions and presents future directions of work.

2. Overview

Before presenting the details of the actual system, this section explains the different parts of the system and their relationship with each other. Fig. 1 shows the components of our traffic video surveillance system in the form of a block diagram.

Camera calibration. Camera calibration is an important part of many computer vision systems. Here we used an un-calibrated camera to capture the video sequence. Camera's intrinsic parameters (e.g. focal length) and its position in the world coordinate system are not known in advance. All these parameters are determined using geometric primitives commonly found in traffic scenes. Using these parameters, ground plane rectification can be done. If a pixel in the image appears on the ground plane, its 3D coordinates can be found in the world reference frame. Worrall *et al.* presented an interactive tool for calibrating a camera that is suitable for use in outdoor scenes [1]. They used this interactive tool to calibrate traffic scenes with acceptable accuracy. Masoud *et al.* presented a method that uses certain geometric primitives commonly found in traffic scenes in order to recover calibration parameters [2].

Background modeling and foreground object detection. We use static camera to capture the traffic video. However, inherent changes in the background

itself, such as wavering trees and flags, water surfaces, etc. the background of the video may not be completely stationary. These types of backgrounds are referred to as quasi-stationary backgrounds. Therefore, background modeling for traffic video surveillance needs to meet certain requirements. It needs to be able to handle quasi-stationary backgrounds and it needs to be fast to handle real-time requirements. This part of the system detects the moving objects (blobs) regardless of whether they present a vehicle or non-vehicle. The overall accuracy of the system depends on robust foreground object detection. Background modeling methods can be divided into two types: parametric [3] and non-parametric [4]. Even though some of these methods work well, they are generally unstable if right parameters are not chosen [3], and computationally intensive for real-time use [4]. We use a simple recursive learning method to model the background.

Vehicle pose estimation using optical flow. Optical flow algorithms estimate the motion of each pixel between two image frames. We use optical flow to estimate how different blobs are moving. Assuming that the vehicles move in the forward direction, optical flow gives a good estimate to how vehicles are oriented. This information is used by the reconstruction module to obtain a 3D model of the vehicle with the correct orientation.

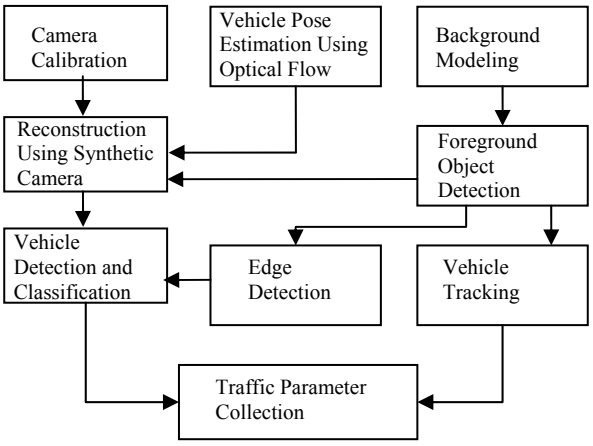


Fig. 1. Traffic video surveillance system overview.

Re-projection using synthetic camera. After the camera parameters are known, we use these parameters to construct a synthetic camera using OpenGL. 3D models are also created in OpenGL for the classes of vehicles for which we want to perform classification. Using the information from the vehicle pose estimation module and the foreground object detection module, we re-project the 3D wire-frame model back onto the image.

Edge Detection. We detect the edges using the Canny edge detector in the regions where the objects were found by the foreground object detection module. These edges are used in the vehicle detection and classification stage.

Vehicle detection and classification. The detection of vehicles has been receiving attention in the computer vision community because vehicles are such a significant part of our life [5, 6]. Vehicle classification is an inherently difficult problem. Gupte *et al.* [7] proposed a system for vehicle detection and classification. The classification is based on vehicle dimensions and is implemented at a very coarse granularity – it can only differentiate cars from non-cars. In order to achieve a finer-level classification of vehicles, we need to have a more sophisticated method that can detect the invariable characteristics for each vehicle category considered. Ma *et al.* developed a vehicle classification approach using modified SIFT descriptors [8]. However, application of such method to any arbitrary viewpoint and its real-time performance are still open questions. We use 3D wire-frame models to detect and classify vehicles. Two different methods were developed to match a 3D wire frame models with the detected edges. The first routine uses a simple color contour technique. The second routine uses a more complex Gaussian-based matching technique that also takes into consideration the gradient.

Vehicle Tracking. To collect any meaningful information from the sequence of images, it is important that we should be able to match the objects detected in consecutive frames. Over the years researchers in computer vision have proposed various solutions to the automated tracking problem. Reader may want to refer to [9] for details of different methods used for tracking. In our implementation, this part of the system tracks the blobs and tries to correct the errors from the foreground object detection module. It also keeps record of the tracks and their 3D world coordinates in each frame.

Traffic Parameter Collection. This module collects and displays information such as the number of active tracks (vehicles), instantaneous velocity of a vehicle, class of a vehicle, and average velocity of a vehicle during the entire time when it was in camera's field of view.

3. Description of our approach

3.1. Camera calibration and synthetic camera modeling

Camera calibration is a very important part of our system. Accuracy of the calibration dictates the accuracy of the overall system. It is equally important that the calibration process should be simple and do not require a special calibration object. Also, if one wants to process the video offline that was taken from an unknown camera at an unknown location, the camera parameters need to be determined from the video sequence itself. This requires the use of a self-calibration approach. Fortunately, traffic scenes generally provide enough geometric primitives to do this on-site.

If we know the vanishing points of the ground plane in perpendicular directions, we can estimate the intrinsic and extrinsic parameters of the camera up to scale. The scale (i.e., camera height) is determined using the point-to-point distances. After determining the camera calibration

parameters, a synthetic camera is constructed in OpenGL that can be used to re-project 3D models back onto the image plane.

3.2. Background Modeling and Foreground Object Detection

This is another important aspect of our video surveillance system. It is very important that this module detects the relevant details of the scene while excluding irrelevant clutter. It also needs to be fast for real-time processing of video sequences. We propose to use an adaptive background model for the entire region of awareness, and for segmenting the moving objects that appear in foreground. Our approach involves learning a statistical color model of the background, and process a new frame using the current distribution in order to segment foreground elements. The algorithm has three distinguishable stages: learning, classification and post-processing. In learning stage, we establish the background model using recursive learning. . We use all the channels (red, green, and blue) of a color image to increase the robustness. We assume that the pixel values tend to have Gaussian distribution and we try to estimate the mean and variance of the distribution using consecutive frames. In the classification stage, we classify the image pixels into foreground and background pixels based on background model. A single Gaussian distribution used in our implementation is not enough to model wavering tree branches which tend to fluctuate between two pixel values. Therefore we use the inherent information available to us to remove unwanted foreground. As we assume a fixed camera position, we can declare the region of interest (ROI) in the scene where vehicles will appear. One more advantage of using a ROI template is that it reduces the overall area to process for foreground object detection, hence speeding the algorithm. After the connected component analysis, we create a list of blobs (foreground objects).

3.3. Vehicle Pose Estimation Using Optical Flow

We need to estimate the pose of a vehicle for further processing (i.e., vehicle detection and classification). We use a pyramidal Lucas and Kanade optical flow technique [10]. Our algorithm has two stages: optical flow estimation and pose estimation. In the first stage, we calculate the pyramidal Lucas and Kanade optical flow for the detected foreground regions. We observed that without any loss of accuracy, we can estimate the optical flow after every T_{of} frames. This serves two purposes: it increases the speed of the algorithm as we don't have to calculate optical flow for every frame and the substantial relative motion between blobs results in more robust optical flow detection. The value of T_{of} depends on the frame rate (frames per second - fps) of the video sequence. We found that T_{of} = fps/10 works well for the video sequences we worked on.

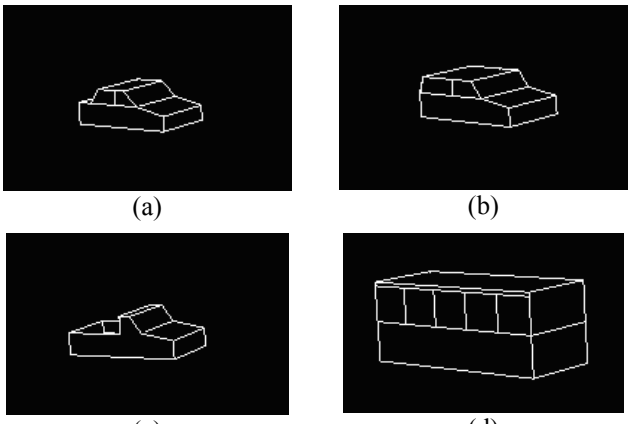
In the next stage, we find the average optical flow vector for every blob. The optical flow feature vectors corresponding to a blob are averaged to get the optical flow average vector v_{avg} that represents the orientation of the blob. If no vector corresponding to a blob is found, the blob is removed from the subsequent processing. Then, the angle α between the vector v_{avg} and the positive Y-axis (both in 3D world coordinate system) is calculated. This resolves the problem of finding the orientation of a blob. To tackle the problem of finding the location of a blob in the 3D world coordinate system, we assume that the center of a blob represents the center of an actual object and all blobs are on the ground plane. Fig. 2 shows an example of the average optical flow vectors (red arrows) found for three vehicles in the image.

Therefore, we have location and orientation of all the moving blobs (vehicles or non-vehicles) in the current frame.

Fig. 2. Average optical flow vectors (red arrows).

3.4. Reconstruction Using Synthetic Camera

We developed four vehicle models for the four classes that we consider here: car, SUV (Sports Utility Vehicle), pickup truck and bus. Fig. 3 shows the 3D wire-frame models. These models are rotated and translated using the output of the vehicle pose estimation module.



(a) (b) (c) (d)

Fig. 3. 3D wire-frame models.
(a) Car (b) SUV (c) Pickup truck (d) Bus

3.5. Vehicle Detection and Classification

We propose two novel methods for vehicle detection and classification. We have incorporated the detection problem as a part of the classification problem. When the matching score for any class of vehicle is lower than a

threshold then the object is classified as non-vehicle. The two classification algorithms proposed in this work are a color contour algorithm and a gradient based contour algorithm. Both algorithms try to match object edges with the 3D wire-frame models. Initially, the edge template is created by detecting the object edges. It is then matched with the 3D wire-frame models of the four vehicle classes. However, before doing this matching we need to rotate and translate the models such that they overlap the actual position of the vehicle in the image. After matching is done for all classes, the best match is assigned as the class of the vehicle under consideration. If the matching score for all classes is less than a threshold T_{match}, then we classify the object as non-vehicle.

Color contour matching algorithm. As discussed earlier, the inputs to this algorithm are the object edge template and the 3D wire-frame template. We create color contour templates using these two templates and then match them by *XOR*ing to get matching template and score. To create a color contour template for the object edge template, we start by drawing filled circles of blue color at all the edge pixels detected. Then, we reduce the radius of the circle gradually and repeat the procedure for green, red and finally for black color. While creating a color contour template for a model edge template, we use only black color. Fig. 4 shows an example of color templates and corresponding matching template. In Fig. 4(a), black contour represents the area closest to the actual edges; red contour represents area closer to the actual edges and so on.

The matching template shown in Fig. 4(c) gives an estimate of how close the 3D wire-frame model is to the edges of the object. Matching score is calculated by counting the number of different color pixels present in the matching template. Each black pixel in the matching template represents perfectly matched edge pixel and contributes the highest to the matching score. Each red pixel contributes more than green and blue pixel, but less than black pixel. This matching score is then normalized using the matching score obtained by *XOR*ing the object edge template with itself. The accuracy of the algorithm can be increased by changing the matching scores and/or changing the radii of the color contours. This method benefits from using graphics functions (drawing circles); as they are generally optimized. However, it lacks the ability to take into consideration the edge direction while doing template matching. Therefore, it gives false positives when a lot of edges are detected in the object edge template.

Gradient based matching algorithm. To address the problems encountered in the color contour matching algorithm, we propose a gradient based matching algorithm. We first calculate the gradient of the edges in both templates (object edge template and model edge template) using a 3×3 Prewitt mask. Then matching is done on the basis of gradient magnitude and direction.

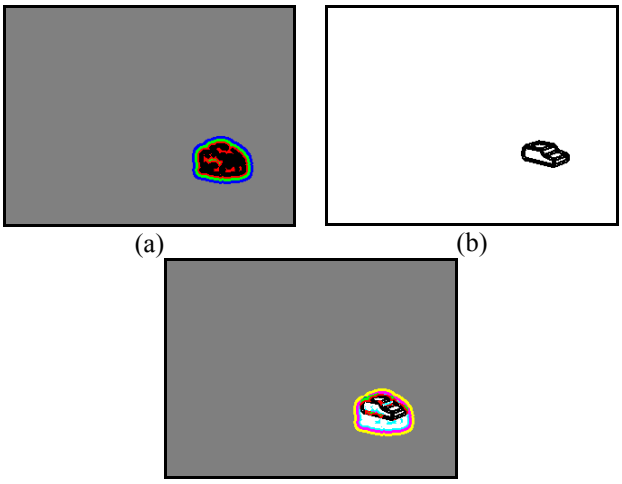


Fig. 4. Color contour templates and matching templates.
(a) Color contour template for object edge template
(b) Color contour template for 3D wire frame model
(c) Matching template derived after *XOR*ing (a) and (b)

We create two separate templates for each object edge template and model edge template. One of these templates contains the gradient magnitude values (magnitude template − *MT*) and other one contains edge direction information (direction template − *DT*). The values at location (i, j) in the magnitude and direction template are calculated using a Gaussian mask of size m×m (m=7 is used in our implementation). Therefore all edge points in the neighborhood of size m×m (centered at location (i, j)) contribute to the magnitude and direction values depending on their distance from pixel (i, j). Then, matching template *MAT* is derived using *MT* and *DT* of the blob edge template (*BET*) and model edge template (*MET*) using following equation:

$$MAT(i,j) = MT_{BET}(i,j) * MT_{MET}(i,j) * \cos(DT_{BET}(i,j) - DT_{MET}(i,j)).$$

The matching score (*MS*) is calculated by using matching template *MAT* using following equation:

$$MS = \begin{cases} \dfrac{\sum_{i,j} MAT(i,j)}{\sum_{i,j} MAT_{self}(i,j)}, & if \ \dfrac{|N(BET) - N(MET)|}{\min(N(BET) - N(MET))} < T_{match} \\ \dfrac{\sum_{i,j} MAT(i,j)}{\sum_{i,j} MAT_{self}(i,j)} \times scaling, & otherwise, \end{cases}$$

where $scaling = e^{-\left(\frac{(N(BET)-N(MET))}{\min(N(BET),N(MET))}\right)^2}$, MAT_{self} is the matching template obtained by matching *BET* with itself. *N(BET)* is the number of edge pixels in blob edge template, *N(MET)* is the number of edge pixels in model edge template, and T_{match} is the threshold that allows slack in difference between *N(BET)* & *N(MET)*.

The benefit of using a gradient based matching is that it takes into consideration the edge direction. While finding the matching score, we take into consideration the number of edge pixels available in both *BET* and *MET*. We do not scale the matching score down if the difference is less

than a threshold T_{match}, but it is scaled exponentially if the difference is more than T_{match}.

3.6. Vehicle Tracking and Traffic Parameter Collection

In this work we propose an algorithm based on blob tracking [7]. The main advantage of this algorithm is its speed. In terms of traffic parameter collection, we keep a record of how each track was classified in each frame, the number of active tracks (vehicles) present at any time, velocity of each vehicle at current time, average velocity of each vehicle during the entire time when it was visible in the camera's field of view. The velocity of the vehicle can be found by using the tracks' location information.

4. Experimental Results

We used our traffic surveillance system to process several video sequences. Results for two of these video sequences (S1 and S2) taken from two different locations are presented in this section.

Fig. 5 shows the correctness of camera calibration and pose estimation routines in video S1.

Fig. 5. Models overlapped onto actual vehicles (S1).

Fig. 6 shows an example of tracking. It shows how the vehicle (black SUV) labeled 094 was successfully tracked between frames 989 and 1093.

Table 1 and Table 2 show quantitative results for the video sequences S1 and S2 respectively. The classification results presented here use gradient based matching. The vehicle classes are car (0), SUV (1), pickup truck (2), bus (3) and non-vehicle (-1). For the patch of street under surveillance in the S1 video sequence, the posted speed limit was 25 mph, whereas it was 35 mph for the S2 video sequence. The average velocity found by the traffic surveillance system for different vehicles is in accord with the posted speed limits.

(a)

(b)

Fig. 6. Vehicle tracking (S1).
(a) Frame no. 989 of S1. (b) Frame no. 1093 of S1.

Table 1. Quantitative Results for the video sequence S1.

Vehicle No.	Actual class of vehicle	Maximally detected class of vehicle	Average Velocity (mph)
1	2	2	29.01
2	3	3	22.70
3	0	0	29.23
4	1	0	32.53
5	1	1	22.82
6	1	0	27.54
7	2	0	26.02
8	1	0	22.74
9	0	0	23.73
10	1	0	34.95
11	0	0	24.73
12	0	0	24.95
13	0	0	26.96
14	0	0	27.88
15	1	1	32.17
16	1	0	24.71
17	1	0	24.40
18	2	2	24.86
19	0	0	24.51
20	2	0	24.64
21	1	0	21.52

Table 2. Quantitative Results for the video sequence S2.

Vehicle No.	Actual class of vehicle	Maximally detected class of vehicle	Average Velocity (mph)
1	2	2	49.37
2	0	0	45.03
3	1	2	36.71
4	0	2	39.37
5	2	2	39.48
6	1	2	43.84
7	0	0	39.37
8	1	0	40.56
9	0	2	42.05
10	2	2	38.54
11	2	2	36.70

5. Conclusions

We presented a traffic surveillance system that identifies, classifies and tracks vehicles. The system is general enough to be capable of detecting, tracking and classifying vehicles while requiring only minimal scene-specific knowledge. We used a camera modeling technique that does not require prior, explicit calibration. The overall accuracy of the camera calibration system was good and it can be verified from the re-projected models that match the actual position of the vehicles in the image. The foreground object detection technique used is fast and found to be reliable. For the two video sequences, the tracking success rate was as high as 90%. We were also able to detect the average vehicle speeds using information recorded by tracking module. This information can also be used to find the number of vehicles present in the camera's field of view at particular time and to find the traffic flow in each direction.

We found that for the purpose of vehicle detection, the 3D wire-frame models used in this work are detailed enough to achieve high vehicle detection accuracy. We developed and used two 3D-model based matching techniques, namely color contour matching and gradient based matching. The benefit of using this technique is that it is fast (5 fps without optimization), compared to the different strategies suggested in the literature. Also, we did not restrict our view to lateral view with which many researchers were successful in getting high accuracy. Vehicle classification is the last step after foreground object detection, vehicle detection, and pose estimation. Therefore, errors in earlier steps affect vehicle classification. The classification accuracy can be improved by doing hierarchical classification where initial classification is done on the basis of length, width and area of blob, followed by further classification with detailed 3D class models.

Overall, the system works well as far as foreground object detection, tracking, vehicle detection and vehicle speed estimation are concerned. This paper also introduced two novel viewpoint independent vehicle classification approaches. They can be modified to do hierarchical classification where initial classification is done on the basis of length, width and area of blob, followed by further classification with detailed 3D class models.

References

[1] A. Worrall and G. Sullivan and K. Baker, A simple intuitive camera calibration tool for natural images, *Proc. of British Machine Vision Conference*, 1994, 781-790.

[2] O. Masoud, N. P. Papanikolopoulos, Using Geometric Primitives to Calibrate Traffic Scenes, *Proc. of International Conference on Intelligent Robots and Systems*, 2, 2004, 1878-1883.

[3] C. Wern, A. Azarbayejani, T. Darrel, and A. Petland, Pfinder: real-time tracking of human body, *IEEE Transactions on PAMI*, 19(7), 1997, 780–785.

[4] A. Tavakkoli, M. Nicolescu, G. Bebis, A Novelty Detection Approach for Foreground Region Detection in Videos with Quasi-stationary Backgrounds, *Proc. of the 2nd International Symposium on Visual Computing*, Lake Tahoe, Nevada, 2006, 40-49.

[5] C. Papageorgiou, and T. Poggio, A Trainable System for Object Detection, *International Journal of Computer Vision*, 38(1), 2000, 15-33.

[6] A. Rajagopalan, P. Burlina and R. Chellappa, Higher Order Statistical Learning for Vehicle Detection in Images, *Proc. of IEEE International Conference on Computer Vision*, 2, 1999, 1204-1209.

[7] S. Gupte, O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos, Detection and Classification of Vehicles, *IEEE Transactions on Intelligent Transportation Systems*, 3(1), 2002, 37-47.

[8] X. Ma, W. E. L. Grimson, Edge-based rich representation for vehicle classification, *Proc. of the International Conference on Computer Vision*, 2006, 1185-1192.

[9] N. K. Kanhere, S. T. Birchfield, and W. A. Sarasua, Vehicle Segmentation and Tracking in the Presence of Occlusions, *Transportation Research Board Annual Meeting*, 2006.

[10] B. D. Lucas, T. Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision, *Proc. of Imaging Understanding Workshop*, 1981, 121-130.