Detection and Classification of Vehicles

Surendra Gupte, Osama Masoud, Robert F. K. Martin, and Nikolaos P. Papanikolopoulos, *Member, IEEE*

***Abstract*—This paper presents algorithms for vision-based detection and classification of vehicles in monocular image sequences of traffic scenes recorded by a stationary camera. Processing is done at three levels: raw images, region level, and vehicle level. Vehicles are modeled as rectangular patches with certain dynamic behavior. The proposed method is based on the establishment of correspondences between regions and vehicles, as the vehicles move through the image sequence. Experimental results from highway scenes are provided which demonstrate the effectiveness of the method. We also briefly describe an interactive camera calibration tool that we have developed for recovering the camera parameters using features in the image selected by the user.**

***Index Terms*—Camera calibration, vehicle classification, vehicle detection, vehicle tracking.**

I. Introduction

TRAFFIC management and information systems rely on a suite of sensors for estimating traffic parameters. Currently, magnetic loop detectors are often used to count vehicles passing over them. Vision-based video monitoring systems offer a number of advantages. In addition to vehicle counts, a much larger set of traffic parameters such as vehicle classifications, lane changes, etc., can be measured. Besides, cameras are much less disruptive to install than loop detectors. Vehicle classification is important in the computation of the percentages of vehicle classes that use state-aid streets and highways. The current situation is described by outdated data and often, human operators manually count vehicles at a specific street. The use of an automated system can lead to accurate design of pavements (e.g., the decision about thickness) with obvious results in cost and quality. Even in large metropolitan areas, there is a need for data about vehicle classes that use a particular street. A classification system like the one proposed here can provide important data for a particular design scenario.

Our system uses a single camera mounted on a pole or other tall structure, looking down on the traffic scene. It can be used for detecting and classifying vehicles in multiple lanes and for any direction of traffic flow. The system requires only the camera calibration parameters and direction of traffic for initialization.

Manuscript received April 2001; revised February 25, 2002. This work was supported in part by the ITS Institute at the University of Minnesota, by the Minnesota Department of Transportation, and in part by the National Science Foundation under Grant CMS-0127893. The Guest Editor for this paper was P. A. Ioannou.

S. Gupte was with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA. He is now with Sun Microsystems, Palo Alto, CA 94303 USA (e-mail: gupte@cs.umn.edu).

O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: masoud@cs.umn.edu; martin@cs.umn.edu; npapas@cs.umn.edu).

Publisher Item Identifier S 1524-9050(02)04366-1.

The paper starts by describing an overview of related work, then a description of our approach is presented, a camera calibration tool developed by our group is described, experimental results are presented, and finally conclusions are drawn.

II. Related Work

Tracking moving vehicles in video streams has been an active area of research in computer vision. In [2], a real time system for measuring traffic parameters is described. It uses a feature-based method along with occlusion reasoning for tracking vehicles in congested traffic scenes. In order to handle occlusions, instead of tracking entire vehicles, vehicle subfeatures are tracked. This approach however is very computationally expensive. In [6], a moving object recognition method is described that uses an adaptive background subtraction technique to separate vehicles from the background. The background is modeled as a slow time-varying image sequence, which allows it to adapt to changes in lighting and weather conditions. In a related work described in [10], pedestrians are tracked and counted using a single camera. The images from the input image sequence are segmented using background subtraction. The resulting connected regions are then grouped together into pedestrians and tracked. Merging and splitting of regions is treated as a graph optimization problem. In [11], a system for detecting lane changes of vehicles in a traffic scene is introduced. The approach is similar to the one described in [10] with the addition that trajectories of the vehicles are determined to detect lane changes.

Despite the large amount of literature on vehicle detection and tracking, there has been relatively little work done in the field of vehicle classification. This is because vehicle classification is an inherently hard problem. Moreover, detection and tracking are simply preliminary steps in the task of vehicle classification. Given the wide variety of shapes and sizes of vehicles within a single category alone, it is difficult to categorize vehicles using simple parameters. This task is made even more difficult when multiple categories are desired. In real-world traffic scenes, occlusions, shadows, camera noise, changes in lighting and weather conditions, etc. are a fact of life. In addition, stereo cameras are rarely used for traffic monitoring. This makes the recovery of vehicle parameters—such as length, width, height etc., even more difficult given a single camera view. The inherent complexity of stereo algorithms and the need to solve the correspondence problem makes them unfeasible for real-time applications.

In [9], a vehicle tracking and classification system is described that can categorize moving objects as vehicles or humans. However, it does not further classify the vehicles into various classes. In [7], an object classification approach that uses parameterized 3-D models is described. The system uses a

3-D polyhedral model to classify vehicles in a traffic sequence. The system uses a generic vehicle model based on the shape of a typical sedan. The underlying assumption being that in typical traffic scenes, cars are more common than trucks or other types of vehicles. The University of Reading has done extensive work in three-dimensional tracking of vehicles and classification of the tracked vehicles using 3-D model matching methods. Baker and Sullivan [1] and Sullivan [13] utilized knowledge of the camera calibration and that vehicles move on a plane in their 3-D model-based tracking. Three-dimensional wireframe models of various types of vehicles (e.g., sedans, hatchbacks, wagons, etc.) were developed. Projections of these models were then compared to features in the image. In [15], this approach was extended so that the image features act as forces on the model. This reduced the number of iterations and improved performance. They also parameterized models as deformable templates and used principal component analysis to reduce the number of parameters. Sullivan *et al.* [14] developed a simplified version of the model-based tracking approach using orthographic approximations to attain real-time performance.

III. Overview

The system we propose consists of six stages.

1) Segmentation: in this stage, the vehicles are separated from the background in the scene.
2) Region Tracking: the result of the segmentation step is a collection of connected regions. This stage tracks regions over a sequence of images using a spatial matching method.
3) Recovery of Vehicle Parameters: to enable accurate classification of the vehicles, the vehicle parameters such as length, width, and height need to be recovered from the 2-D projections of the vehicles. This stage uses information about the camera's location and makes use of the fact that in a traffic scene, all motion is along the ground plane.
4) Vehicle Identification: our system assumes that a vehicle may be made up of multiple regions. This stage groups the tracked regions from the previous stage into vehicles.
5) Vehicle Tracking: due to occlusions, noise, etc., there is not necessarily a one-to-one correspondence between regions and vehicles, i.e., a vehicle may consist of multiple regions and a single region might correspond to multiple vehicles. To enable tracking of vehicles despite these difficulties, our system does tracking at two levels—region level and the vehicle level.
6) Vehicle Classification: after vehicles have been detected and tracked, they are classified.

The following sections describe each of these stages in more detail.

IV. Segmentation

The first step in detecting vehicles is segmenting the image to separate the vehicles from the background. There are various approaches to this, with varying degrees of effectiveness. To be useful, the segmentation method needs to accurately separate vehicles from the background, be fast enough to operate in real time, be insensitive to lighting and weather conditions, and require a minimal amount of initialization. In [6], a segmentation approach using adaptive background subtraction is described. Kalman filtering is used to predict the background image during the next update interval. The error between the prediction and the actual background image is used to update the Kalman filter-state variables. This method has the advantage that it automatically adapts to changes in lighting and weather conditions. However, it needs to be initialized with an image of the background without any vehicles present. In [5], a probabilistic approach to segmentation is described. They use the expectation maximization (EM) method to classify each pixel as moving object, shadow or background. Another approach to segmentation is time differencing, (used in [9]) which consists of subtracting successive frames (or frames a fixed interval apart). This method is also insensitive to lighting conditions and has the further advantage of not requiring initialization with a background image. However, this method produces many small regions that can be difficult to separate from noise.

We use a self-adaptive background subtraction method for segmentation. This is similar in principle to the method described in [5]. However we use a much simpler and more robust method for updating the background. In addition, our method automatically extracts the background from a video sequence and so no manual initialization is required.

Our segmentation technique consists of three tasks:

- Segmentation;
- Background update;
- Background extraction.

A. Segmentation

For each frame of the video sequence (referred to as *current image*), we take the difference between the current image and the current background giving the *difference image*. The difference image is thresholded to give a binary *object mask*. The object mask is a binary image such that all pixels that correspond to foreground objects have the value 1, and all the other pixels are set to 0.

B. Adaptive Background Update

The basic principle of our method is to modify the background image that is subtracted from the current image (called the *current background*) so that it looks similar to the background in the current video frame. We update the background by taking a weighted average of the current background and the current frame of the video sequence. However, the current image also contains foreground objects. Therefore, before we do the update we need to classify the pixels as foreground and background and then use only the background pixels from the current image to modify the current background. Otherwise, the background image would be polluted with the foreground objects. The binary object mask is used to distinguish the foreground pixels from the background pixels. The object mask is used as a gating function that decides which image to sample for updating the background. At those locations where the mask is 0 (corresponding to the background pixels), the current image is sampled. At those locations where the mask is 1—corresponding to

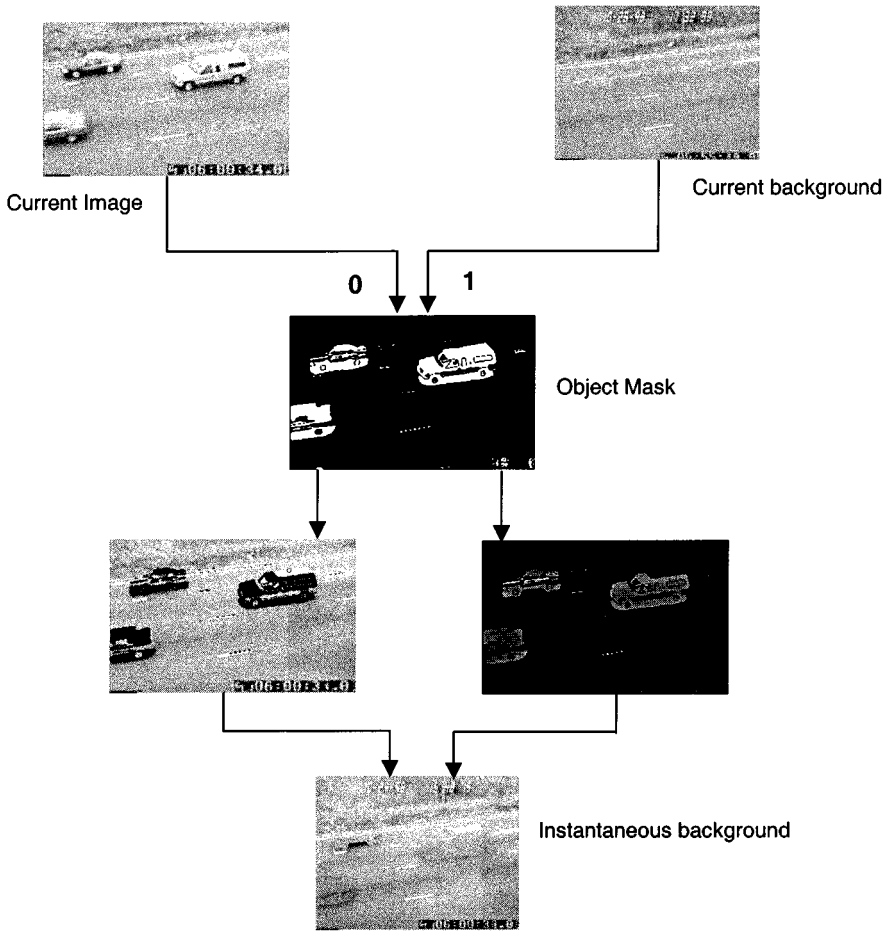


Fig. 1. Computation of the instantaneous background.

foreground pixels—the current background is sampled. The result of this is what we call the *instantaneous background*. This operation described above can be shown in diagrammatic form as in Fig. 1.

The current background is set to be the weighted average of the instantaneous and the current background

$$\mathrm{CB} = \alpha \mathrm{IB} + (1 - \alpha)\mathrm{CB}. \tag{1}$$

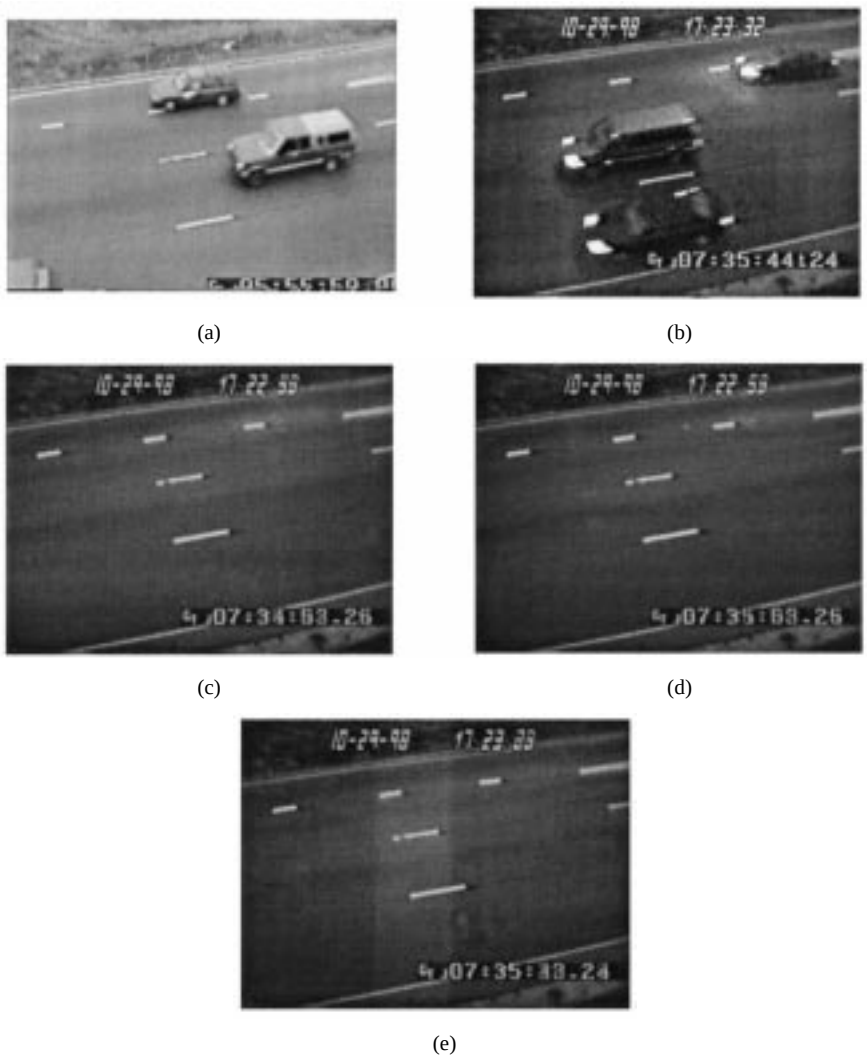
The weights assigned to the current and instantaneous background affect the update speed. We want the update speed to be fast enough so that changes in illumination are captured quickly, but slow enough so that momentary changes (due to, say the AGC of the camera being activated) do not persist for an unduly long amount of time. The weight has been empirically determined to be 0.1. We have found that this gives the best tradeoff in terms of update speed and insensitivity to momentary changes.

C. Dynamic Threshold Update

After subtracting the current image from the current background, the resultant difference image has to be thresholded to get the binary object mask. Since the background changes dynamically, a static threshold cannot be used to compute the object mask. Moreover, since the object mask itself is used in updating the current background, a poorly set threshold would result in poor segmentation. Therefore we need a way to update the threshold as the current background changes. The difference image is used to update the threshold. In our images, a major portion of the image consists of the background. Therefore the difference image would consist of a large number of pixels having low values, and a small number of pixels having high values. We use this observation in deciding the threshold. The histogram of the difference image will have high values for low pixel intensities and low values for the higher pixel intensities. To set the threshold, we need to look for a dip in the histogram that occurs to the right of the peak. Starting from the pixel value corresponding to the peak of the histogram, we search toward increasing pixel intensities for a location on the histogram that has a value significantly lower than the peak value (we use 10% of the peak value). The corresponding pixel value is used as the new threshold.

D. Automatic Background Extraction

In video sequences of highway traffic it might be impossible to acquire an image of the background. A method that can automatically extract the background from a sequence of video images would be very useful. Here we assume that the background is stationary and any object that has significant motion is considered part of the foreground. The method we propose works with video images and gradually builds up the background image over time.



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

Fig. 2. Background adaptation to changes in lighting conditions. (a) Initial background provided to the algorithm. (b) Image of the scene at dusk. (c) Current background after 4 s. (d) Current background after 6 s. (e) Current background after 8 s.

The background and threshold updating described above is done at periodic update intervals. To extract the background, we compute a binary *motion mask* by subtracting images from two successive update intervals. All pixels that have moved between these update intervals are considered part of the foreground. To compute the motion mask for frame i (MM_i), the binary object masks from update interval i (OM_i) and update interval $i-1$ (OM_{i-1}) are used. The motion mask is computed as

$$MM_i = \sim OM_{i-1} \,\&\, OM_i. \tag{2}$$

This motion mask is now used as the gating function to compute the instantaneous background as described above. Over a sequence of frames the current background looks similar to the background in the current image.

E. Self-Adaptive Background Subtraction Results

Fig. 2(a)–(e) shows some images that demonstrate the effectiveness of our self-adaptive background subtraction method. The image (a) was taken during the day. This was given as the initial background to the algorithm. The image (b) shows the same image at dusk. The images (c), (d), and (e) show how the background adaptation algorithm updates the background so that it closely matches the background of image (b). Fig. 3(a)–(e) demonstrates how the algorithm copes with changes in camera orientation.

V. Region Tracking

A vision-based traffic monitoring system needs to be able to track vehicles through the video sequence. Tracking eliminates multiple counts in vehicle counting applications. Moreover, the tracking information can also be used to derive other useful information like vehicle velocities. In applications like vehicle classification, the tracking information can also be used to refine the vehicle type and correct for errors caused due to occlusions.

The output of the segmentation step is a binary object mask. We perform region extraction on this mask. In the region tracking step, we want to associate regions in frame i with the regions in frame $i+1$. This allows us to compute the velocity of the region as it moves across the image and also helps in the vehicle tracking stage. There are certain problems that need to be handled for reliable and robust region tracking. When

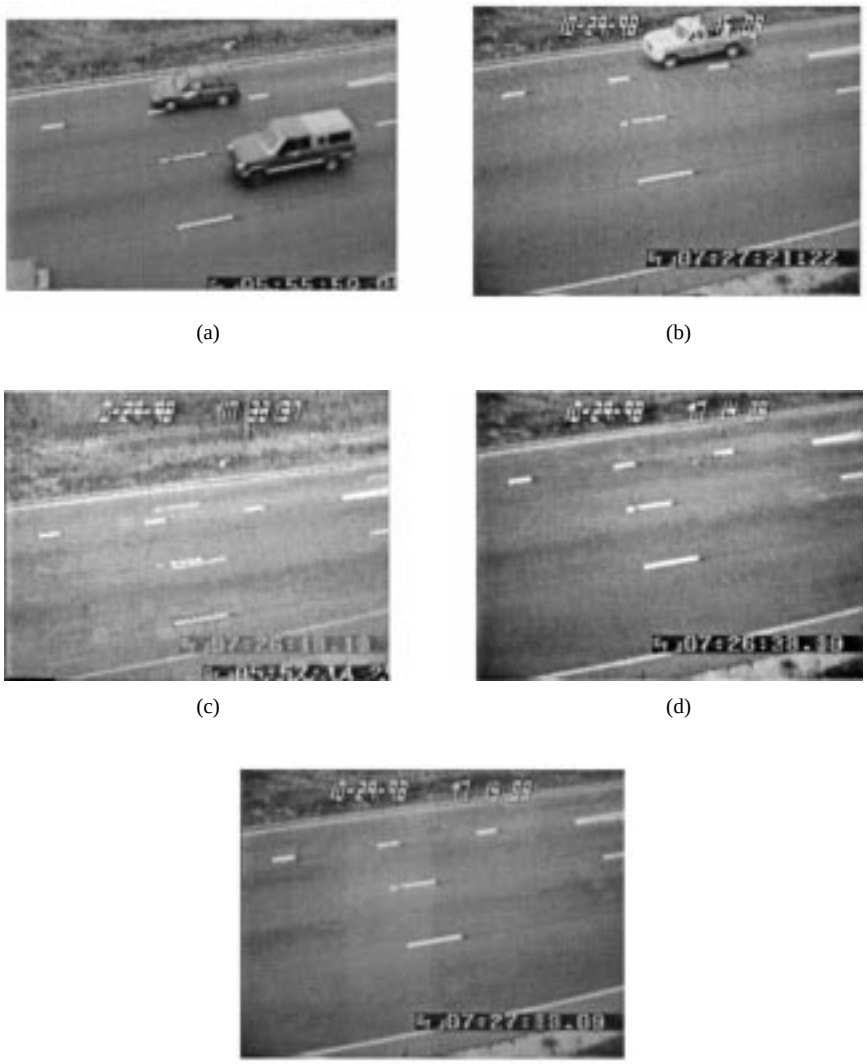


Fig. 3. Background adaptation to changes in camera orientation. (a) Initial background provided to the algorithm. (b) Image of the scene after camera orientation has changed. (c) Current background after 4 s. (d) Current background after 6 s. (e) Current background after 8 s.

considering the regions in frame i and frame $i+1$ the following problems might occur:

- A region might disappear. Some of the reasons why this may happen are as follows.
  - The vehicle that corresponded to this region is no longer visible in the image, and, hence, its region disappears.
  - Vehicles are shiny metallic objects. The pattern of reflection seen by the camera changes as the vehicles move across the scene. The segmentation process uses thresholding, which is prone to noise. At some point in the scene, the pattern of reflection from a vehicle might fall below the threshold and, hence, those pixels will not be considered as foreground. Therefore the region might disappear even though the vehicle is still visible.
  - A vehicle might become occluded by some part of the background or another vehicle.
- A new region might appear. Some possible reasons for this include the following.
  - A new vehicle enters the field of view of the camera so a new region corresponding to this vehicle appears.
  - For the same reason as that mentioned above, as the pattern of reflections from a vehicle changes, its intensity might now rise above the threshold used for segmentation, and the region corresponding to this vehicle is now detected.
  - A previously occluded vehicle might no longer be occluded.
- A single region in frame i might split into multiple regions in frame $i+1$ because of the following reasons.
  - Two or more vehicles might have been passing close enough to each other that they occlude (or are occluded) and, hence, are detected as one connected region. As these vehicles move apart and are not oc-

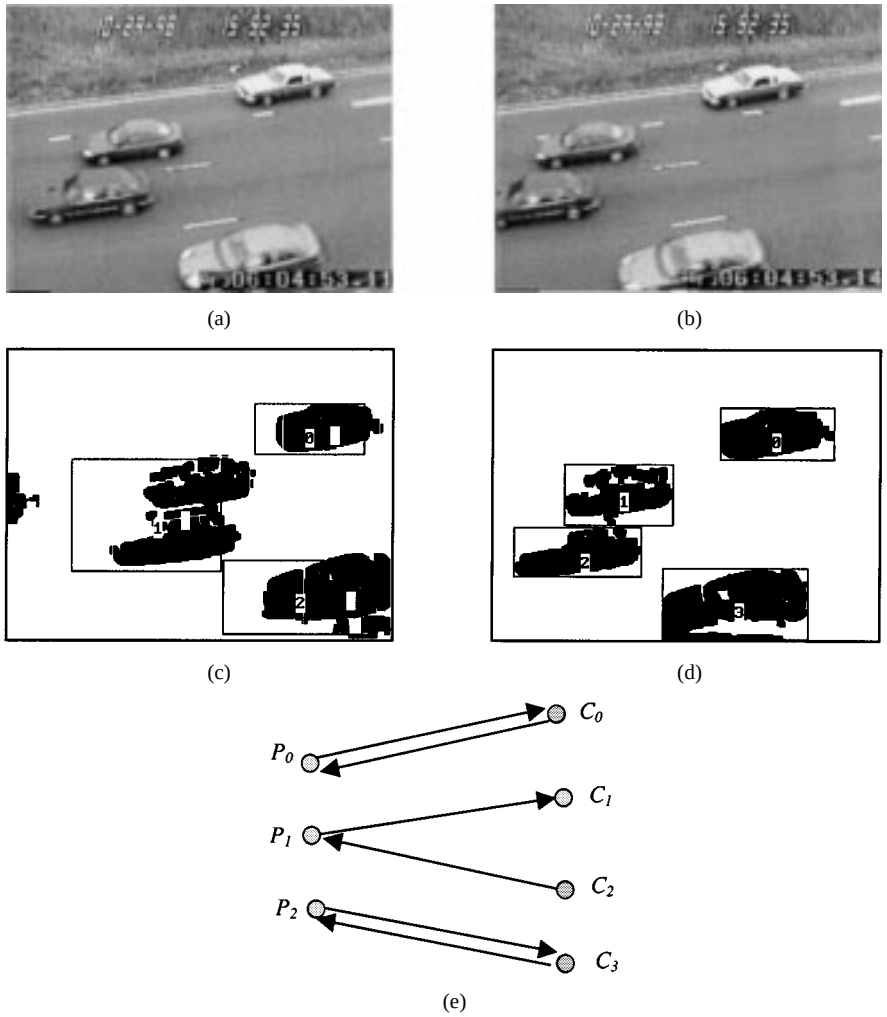


Fig. 4. Detected regions and their resultant association graph. (a) Frame i-1. (b) Frame i. (c) Previous regions P. (d) Current regions C. (e) The association graph shows that P_1 has split into C_1 and C_2.

cluded, the region corresponding to these vehicles might split up into multiple regions.
- Due to noise and errors during the thresholding process, a single vehicle that was detected as a single region might be detected as multiple regions as it moves across the image.

- Multiple regions may merge. Some reasons why this may occur include the following.
  - Multiple vehicles (each of which were detected as one or more regions) might occlude each other and during segmentation get detected as a single region.
  - Again, due to errors in thresholding, a vehicle that was detected as multiple regions might later be detected as a single region.

The region tracking method needs to be able to robustly handle these situations and work reliably even in the presence of these difficulties. We form an association graph between the regions from the previous frame and the regions in the current frame. We model the region tracking problem as a problem of finding the maximal weight graph. The association graph is a bipartite graph where each vertex corresponds to a region. All the vertices in one partition of this graph correspond to regions from the previous frame, P and all the vertices in the other partition correspond to regions in the current frame, C. An edge E_{ij} between vertices V_i and V_j indicates that the previous region P_i is associated with the current region C_j. A weight w is assigned to each edge E_{ij}. The weight of edge E_{ij} is calculated as

$$w(E_{ij}) = A(P_i \cap C_j) \tag{3}$$

i.e., the weight of edge E_{ij} is the area of overlap between region P_i and region C_j. The weight of the graph G is defined as

$$w(G) = \sum_{E_{ij} \in G} E_{ij}. \tag{4}$$

A. Building the Association Graph

The region extraction step is done for each frame resulting in new regions being detected. These become the current regions, C. The current regions from frame i become the previous regions P in frame $i+1$. To add the edges in this graph, a score is computed between each previous region P_i and each current region C_j. The score s is a pair of values $\langle s_{p \to c}, s_{c \to p} \rangle$. It is a measure of how closely a previous region P_i matches a current

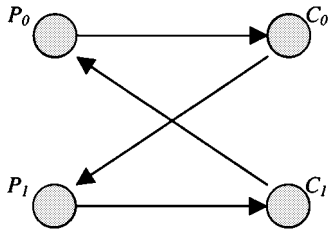


Fig. 5. A conflict component.

region C_j. The area of intersection between P_i and C_j is used in computing s

$$s_{p\to c} = \frac{A(P_i \cap C_j)}{A(P_i)}. \tag{5}$$

$$s_{c\to p} = \frac{A(P_i \cap C_j)}{A(C_j)} \tag{6}$$

This makes the score s independent of the actual area of both regions P_i and C_j.

B. Adding Edges

Each previous region P_i is compared with each current region C_j and the area of intersection between P_i and C_j is computed. The current region $C_{i\max}$ that has the maximum value for $s_{p\to c}$ with P_i is determined. An edge is added between P_i and $C_{i\max}$. Similarly, for each region C_j, the previous region $P_{j\max}$ that has the maximum value for $s_{c\to p}$ with C_j is determined. An edge is added between vertices $P_{j\max}$ and C_j.

The rationale for having a two-part score is that it allows us to handle region splits and merges correctly. Moreover, by always selecting the region $C_{i\max}(P_{j\max})$ that has the maximum value for $s_{p\to c}(s_{c\to p})$ we do not need to set any arbitrary thresholds to determine if an edge should be added between two regions. This also ensures that the resultant association graph generated is a maximal weight graph. An example is shown in Fig. 4.

C. Resolving Conflicts

When the edges are added to the association graph as described above, we might possibly get a graph of the form shown in Fig. 5. In this case, P_0 can be associated with C_0 or C_1, or both C_0 and C_1 (similarly, for P_1). To be able to use this graph for tracking we need to choose one assignment from among these. We enforce the following constraint on the association graph—*in every connected component of the graph only one vertex may have degree greater than 1*. A graph that meets this constraint is considered a conflict-free graph. A connected component that does not meet this constraint is considered a conflict component. We need to remove edges from a conflict component so that it becomes conflict free. 2^n different conflict free components can be generated from a conflict component having n vertices. One possibility is to generate each of the 2^n connected components and select the one with the maximum weight. This is the method used in [10]. However, this can be computationally quite expensive. We use a different method for resolving conflicts. For each conflict component we add edges in increasing order of weight if and only if adding the edge does not violate the constraint mentioned above. If adding an edge E_{ij} will violate the constraint, we simply ignore the edge and select the next one. The resulting graph may be suboptimal (in terms of weight), however, this does not have an unduly large effect on the tracking and is good enough in most cases.

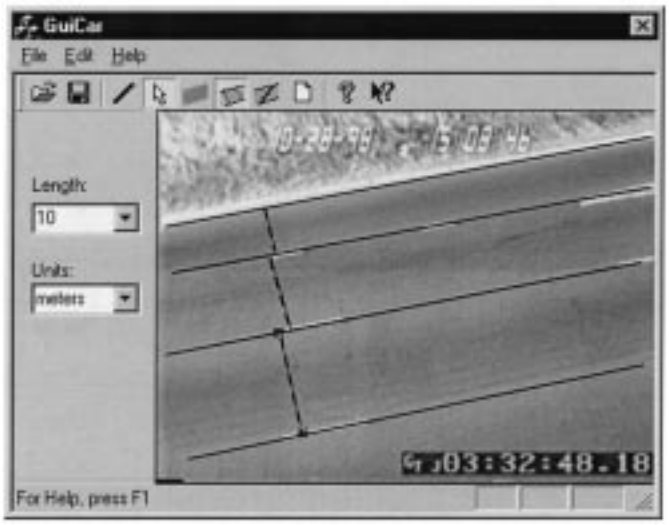
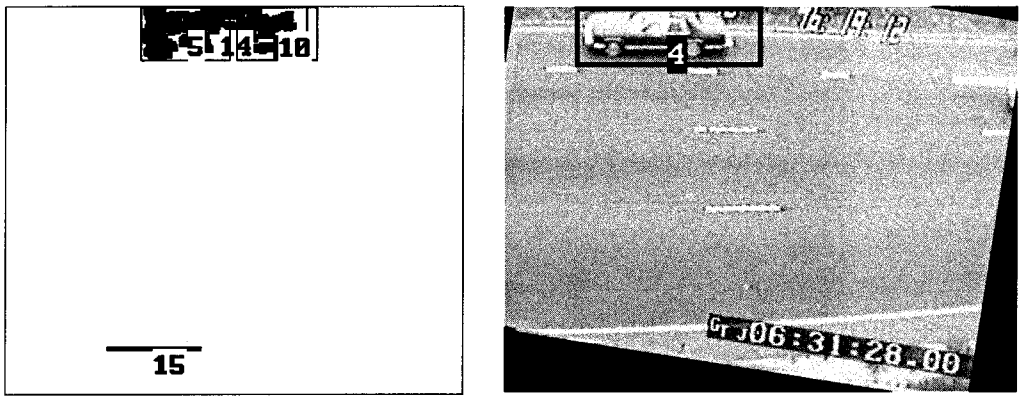


Fig. 6. Camera calibration tool.

VI. Recovery of Vehicle Parameters

To be able to detect and classify vehicles, the location, length, width and velocity of the regions (which are vehicle fragments) needs to be recovered from the image. Knowledge of camera calibration parameters is necessary in estimating these attributes. Accurate calibration can therefore significantly impact the computation of vehicle velocities and classification. Calibration parameters are usually difficult to obtain from the scene as they are rarely measured when the camera is installed. Moreover, since the cameras are installed approximately 20–30 feet above the ground, it is usually difficult to measure certain quantities such as pan and tilt that can help in computing the calibration parameters. Therefore, it becomes difficult to calibrate after the camera has been installed. One way to compute the camera parameters is to use known facts about the scene. For example, we know that the road, for the most part, is restricted to a plane. We also know that the lane markings are parallel and lengths of markings as well as distances between those markings are precisely specified. Once the camera parameters are computed, any point on the image can be back-projected onto the road. Therefore, we have a way of finding the distance between any two points on the road by knowing their image locations.

We have developed a camera calibration tool specifically for the kind of traffic scenes that we are frequently asked to analyze. This interactive tool has a user interface that allows the user to point to some locations on the image (e.g., the endpoints of a lane divider line whose length is known). The system can then compute the calibration parameters automatically. The proposed system is easy to use and intuitive to operate, using obvious landmarks, such as lane markings, and familiar tools, such as a line-drawing tool. The graphical user interface (GUI) allows the user to first open an image of the scene. The user is then able to draw different lines and optionally assign lengths to those lines. The user may first draw lines that represent lane separation (solid line, Fig. 6). They may then draw lines to designate the width of the lanes (dashed line, Fig. 6). The user may



(a) (b)

Fig. 7. (a) Previous regions 5 and 10 merge to form current region 14. (b) Final outcome where all these regions merge to create vehicle 4.

also designate known lengths in conjunction with the lane separation marks. An additional feature of the interface is that it allows the user to define traffic lanes in the video, and also the direction of traffic in these lanes. Also, special hot spots can be indicated on the image, such as the location where we want to compute vehicles' speeds. The actual computation of camera calibration from the information given by the GUI is fully outlined in [12]. This GUI proved to be much more intuitive than the methods we previously used. The only real difficulty arose with respect to accuracy in determining distances in the direction of the road. Some of these inaccuracies arise because the markings on the road themselves are not precise. Another part of the inaccuracy depends on the user's ability to mark endpoints in the image. In general, however, in spite of the inaccuracies discovered, this method of calibration proved to be much quicker than those previously used, more accurate, and more adaptable to generic scenes.

VII. Vehicle Identification

A vehicle is made up of (possibly multiple) regions. The vehicle identification stage groups regions together to form vehicles. New regions that do not belong to any vehicle are considered *orphan regions*. A vehicle is modeled as a rectangular patch whose dimensions depend on the dimensions of its constituent regions. Thresholds are set for the minimum and maximum sizes of vehicles based on typical dimensions of vehicles. A new vehicle is created when an orphan region of sufficient size is tracked over a sequence of a number of frames (three in our case).

VIII. Vehicle Tracking

Our vehicle model is based on the assumption that the scene has a flat ground. A vehicle is modeled as a rectangular patch whose dimensions depend on its location in the image. The dimensions are equal to the projection of the vehicle at the corresponding location in the scene.

A vehicle consists of one or more regions, and a region might be owned by zero or more vehicles. The region tracking stage produces a conflict-free association graph that describes the relations between regions from the previous frame and regions from the current frame. The vehicle tracking stage updates the location, velocity and dimensions of each vehicle based on this association graph. The location and dimensions of a vehicle are computed as the bounding box of all its constituent blobs. The velocity is computed as the weighted average of the velocities of its constituent blobs. The weight for a region $P_i \in$ vehicle v is calculated as

$$w_i = \frac{A(P_i \cap v)}{A(v)}. \tag{7}$$

$A(P_i \cap v)$ is the area of overlap between vehicle v and region P_i. The vehicle's velocity is used to predict its location in the next frame.

A region can be in one of five possible states. The vehicle tracker performs different actions depending on the state of each region that is owned by a vehicle. The states and corresponding actions performed by the tracker are:

1) Update: a previous region P_i matches exactly one current region C_j. The tracker simply updates the ownership relation so that the vehicle v that owned P_i now owns C_j;
2) Merge: regions $P_i \ldots P_k$ merge into a single region C_j. The area of overlap between each vehicle assigned to $P_i \ldots P_k$ is computed with C_j, if the overlap is above a minimum threshold, C_j is assigned to that vehicle;
3) Split: region P_i splits into regions $C_j \ldots C_k$. Again the area of overlap between each vehicle $v \in P_i$ is computed with $C_j \ldots C_k$. If it is greater than a minimum value, the region is assigned to v;
4) Disappear: a region $P_i \in P$ is not matched by any $C_j \in C$. The region is simply removed from all the vehicles that owned it. If a vehicle loses all its regions, it becomes a *phantom* vehicle. Sometimes a vehicle may become temporarily occluded and then later reappear. Phantoms prevent such a nonoccluded vehicle from being considered a new vehicle. A phantom is kept around for a few frames (3), and if it cannot be resurrected within this time, it is removed;
5) Appear: a region $C_j \in C$ does not match any $P_i \in P$. We check C_j with the phantom vehicles. If a phantom vehicle overlaps new region(s) of sufficient area, it is resurrected. If the region does not belong to a phantom vehicle and is of sufficient size, a new vehicle is created.

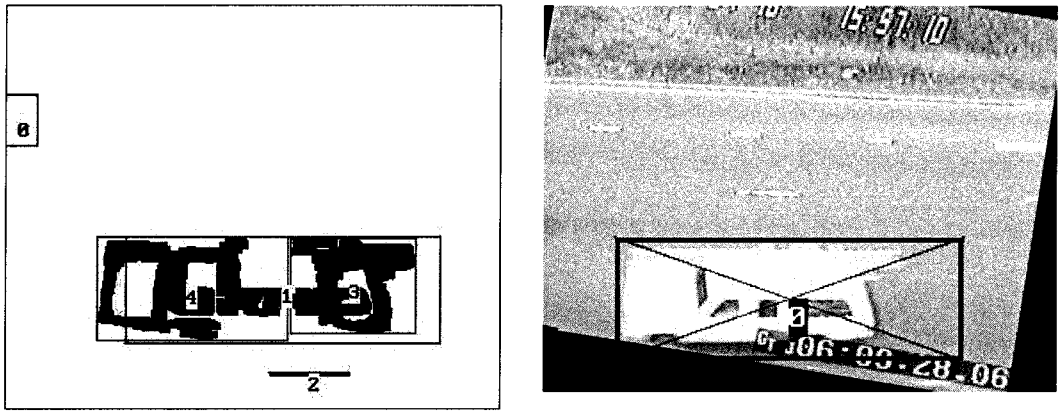


Fig. 8. (a) Previous region 1 splits into current regions 3 and 4. (b) The tracker correctly associates both the new regions with the same vehicle.

IX. Classification

To be useful, a vehicle classification system should categorize vehicles into a sufficiently large number of classes. However as the number of categories increases, the processing time required to do the classification also increases. Therefore, a hierarchical classification method is needed which can quickly categorize vehicles at a coarse granularity. Then depending on the application, further classification at the desired level of granularity can be done.

We use vehicle dimensions to classify vehicles into two categories: cars (which constitute the majority of vehicles) and noncars (vans, SUVs, pickup trucks, tractor-trailers, semis and buses). Separating, say SUVs from pickup trucks would require the use of more sophisticated, shape-based techniques. However, doing a coarse, dimension-based classification at the top-level significantly reduces the amount of work that needs to be done at a lower level. The final goal of our system is to be able to do a vehicle classification at multiple levels of granularity but currently, we classify vehicles into the aforementioned categories (based on the needs of the funding agency).

Since classification is done based on the dimensions of vehicles, we compute the actual length and height (Section VI) of the vehicles. Due to the camera orientation, the computed height is actually a combination of the vehicle's width and height. It is not possible to separate the two using merely vehicle boundaries in the image and camera parameters. The category of a vehicle is determined from its length and this combined value. We took a sample of 50 cars and 50 trucks and calculated the mean and variance of these samples. We used the combined width/height value for the vehicle height computed using the camera calibration parameters. From these samples, we were able to compute a discriminant function that can be used to classify vehicles.

The average dimensions of a truck are only slightly larger than the dimensions of the average car. In some cases, cars may actually be longer and wider than trucks (i.e., a Cadillac versus a small pickup). This ambiguity allows some error to enter the system when we define a decision boundary.

X. Results

The system was implemented on a dual Pentium 400 MHz PC equipped with a C80 Matrox Genesis vision board. We tested the system on image sequences of highway scenes. The system is able to track and classify most vehicles successfully. In a 20 minute sequence of freeway traffic, 90% of the vehicles were correctly detected and tracked. Of these correctly tracked vehicles, 70% of the vehicles were correctly classified. The processing was done at a frame rate of 15 frames/s. Figs. 7–9 show some results of our system. With more optimized algorithms, the processing time per frame could be reduced significantly. Errors in detection were most frequently due to occlusions and/or poor segmentation. Because of imperfections in updating the background, noise can be added or subtracted from the detected vehicles. At times, the noise is sufficient enough to cause the detected object to become too large or too small to be considered a vehicle. Also, when multiple vehicles occlude each other, they are often detected as a single vehicle. However, if the vehicles later move apart, the tracker is robust enough to correctly identify them as separate vehicles. Unfortunately, the two vehicles will continue to persist as a single vehicle if the relative motion between them is small. In such a case, the count of vehicles is incorrect. Another thing to note is that the images we used are grayscale. Since our segmentation approach is intensity based, vehicles whose intensity is similar to the road surface are sometimes missed, or detected as fragments that are too small to be reliably separated from noise. This too will cause an incorrect vehicle count. Classification errors were mostly due to the small separation between vehicle classes. Because we only used size as our metric, it is impossible to correctly classify all vehicles. To further improve our performance in classification rates, we would need to investigate additional cues.

Although our data consisted of traffic moving in only one direction, the tracking algorithm is general enough to work with multiple traffic directions. Also, our data was acquired on an overcast day thus removing the problem of shadows. As described, our system does not handle shadows although preliminary investigations into shadow handling have been positive. We intend to further test the system on more complex scenes and under a wider variety of illumination and weather conditions.

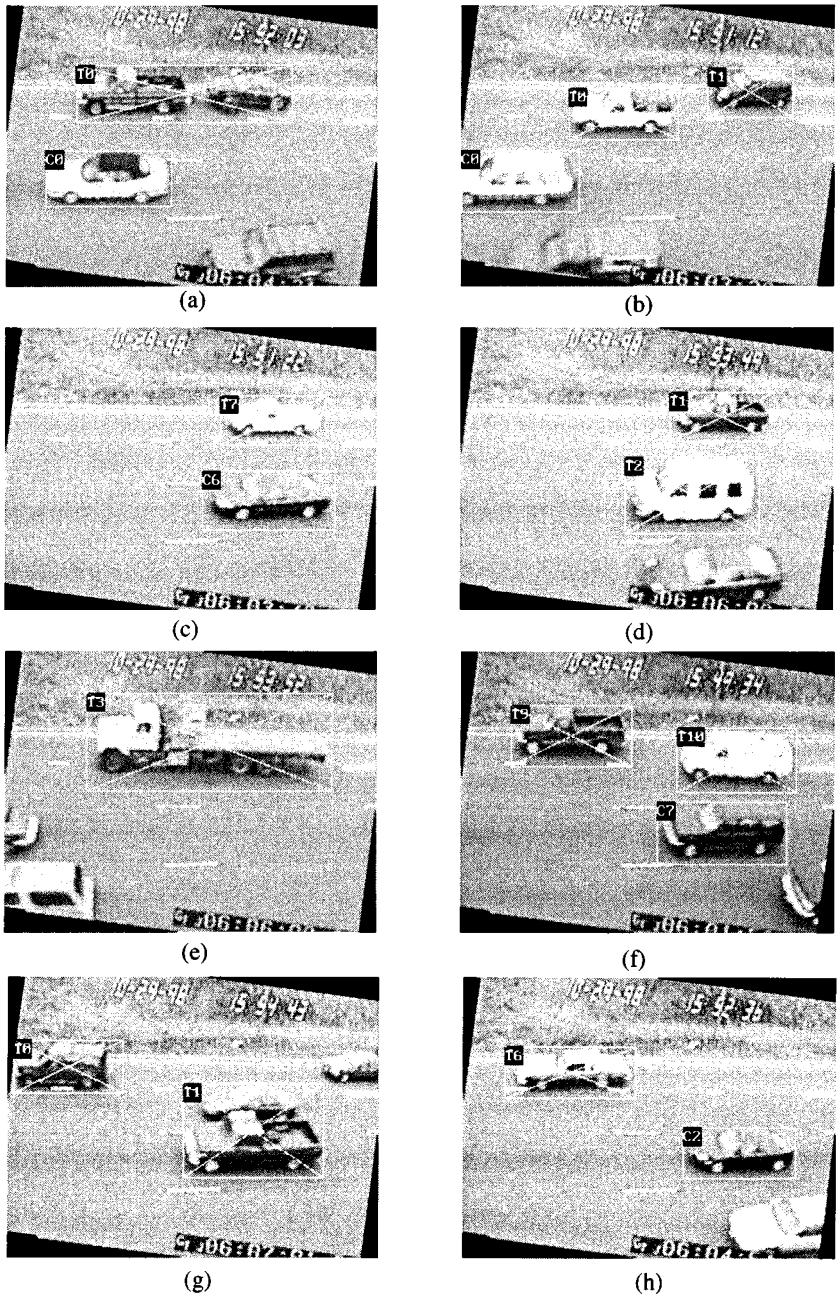


Fig. 9. Snapshots of different car/truck classification results. Cars are shown surrounded by their bounding box; noncars are marked by drawing the diagonals of the bounding box. Notice how in (g), the two vehicles were grouped and considered a truck because of large combined size. In (h), the car was misclassified as a truck due to large size.

We are also currently exploring various techniques to deal with the problems mentioned above.

XI. Conclusions and Future Work

We have presented a model-based vehicle tracking and classification system capable of working robustly under most circumstances. The system is general enough to be capable of detecting, tracking and classifying vehicles while requiring only minimal scene-specific knowledge. In addition to the vehicle category, the system provides location and velocity information for each vehicle as long as it is visible. Initial experimental results from highway scenes were presented.

To enable classification into a larger number of categories, we intend to use a nonrigid model-based approach to classify vehicles. Parameterized 3-D models of exemplars of each category will be used. Given the camera calibration a 2-D projection of the model will be formed from this viewpoint. This projection will be compared with the vehicles in the image to determine the class of the vehicle.

References

[1] K. D. Baker and G. D. Sullivan, "Performance assessment of model-based tracking," in *Proc. IEEE Workshop Applications of Computer Vision*, Palm Springs, CA, 1992, pp. 28–35.

[2] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A real-time computer vision system for measuring traffic parameters," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Puerto Rico, June 1997, pp. 496–501.
[3] M. Burden and M. Bell, "Vehicle classification using stereo vision," in *Proc. 6th Int. Conf. Image Processing and Its Applications*, vol. 2, 1997, pp. 881–887.
[4] A. De La Escalera, L. E. Moreno, M. A. Salichs, and J. M. Armingol, "Road traffic sign detection and classification," *IEEE Trans. Ind. Electron.*, vol. 44, pp. 848–859, Dec. 1997.
[5] N. Friedman and S. Russell, "Image segmentation in video sequences," in *Proc. 13th Conf. Uncertainty in Artificial Intelligence*, Providence, RI, 1997.
[6] K. P. Karmann and A. von Brandt, "Moving object recognition using an adaptive background memory," in *Proc. Time-Varying Image Processing and Moving Object Recognition*, vol. 2, V. Capellini, Ed., 1990.
[7] D. Koller, "Moving object recognition and classification based on recursive shape parameter estimation," in *Proc. 12th Israel Conf. Artificial Intelligence, Computer Vision*, Dec. 27–28, 1993.
[8] D. Koller, J. Weber, T. Huang, G. Osawara, B. Rao, and S. Russel, "Toward robust automatic traffic scene analysis in real-time," in *Proc. 12th Int. Conf. Pattern Recognition*, vol. 1, 1994, pp. 126–131.
[9] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving target classification and tracking from real-time video," in *Proc. IEEE Workshop Applications of Computer Vision*, 1998, pp. 8–14.
[10] O. Masoud and N. P. Papanikolopoulos, "Robust pedestrian tracking using a model-based approach," in *Proc. IEEE Conf. Intelligent Transportation Systems*, Nov. 1997, pp. 338–343.
[11] O. Masoud, N. P. Papanikolopoulos, and E. Kwon, "Vision-based monitoring of weaving sections," in *Proc. IEEE Conf. Intelligent Transportation Systems*, Oct. 1999, pp. 770–775.
[12] O. Masoud, S. Rogers, and N.P. Papanikolopoulos, "Monitoring Weaving Sections," ITS Institute, Univ. Minnesota, Minneapolis, CTS 01-06, Oct. 2001.
[13] G. D. Sullivan, "Model-based vision for traffic scenes using the ground-plane constraint," *Phil. Trans. Roy. Soc. (B)*, vol. 337, pp. 361–370, 1992.
[14] G. D. Sullivan, K. D. Baker, A. D. Worrall, C. I. Attwood, and P. M. Remagnino, "Model-based vehicle detection and classification using orthographic approximations," *Image Vis. Comput.*, vol. 15, no. 8, pp. 649–654, Aug. 1997.
[15] G. D. Sullivan, A. D. Worrall, and J. M. Ferryman, "Visual object recognition using deformable models of vehicles," in *Proc. Workshop on Context-Based Vision*, Cambridge, MA, June 1995, pp. 75–86.

Surendra Gupte was born in Bombay, India, in 1974. He received the Bachelor of Engineering degree from Poona University, Poona, India, in 1996 and the M.S. degree in computer science from the University of Minnesota, Minneapolis, in 2001.

He is currently working at Sun Microsystems, Palo Alto, CA. His research interests include computer vision, computer graphics, and artificial intelligence.

Osama Masoud was born in Riyadh, Saudi Arabia, in 1971. He received the B.S. and M.S. degrees in computer science from King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, in 1992 and 1994, respectively, and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, in 2000.

He is currently a Postdoctoral Associate in the Department of Computer Science and Engineering at the University of Minnesota. His research interests include computer vision, robotics, transportation applications, and computer graphics.

Dr. Masoud is the recipient of a Research Contribution Award from the University of Minnesota, the Rosemount Instrumentation Award from Rosemount Inc., Chanhassen, MN, and the Matt Huber Award for Excellence in Transportation Research.

Robert F. K. Martin was born in Appleton, WI, in 1972. He received the B.E.E. degree from the University of Minnesota-Twin Cities, Minneapolis, in 1995. He is currently working toward the Ph.D. degree at the same university.

From 1995 to 2001, he was a Principal Engineer with Lockheed Martin, Bethesda, MD. His research interests include computer vision, machine learning, and cognitive psychology.

Nikolaos P. Papanikolopoulos (S'88–M'93) was born in Piraeus, Greece, in 1964. He received the Diploma degree in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1987, the M.S.E.E. degree in electrical engineering, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, in 1988, and 1992, respectively.

Currently, he is a Professor in the Department of Computer Science and Engineering at the University of Minnesota, Minneapolis, MN. He has authored or coauthored more than 150 journal and conference papers and 35 refereed journal papers. His research interests include robotics, sensors for transportation applications, control, and computer vision.

Dr. Papanikolopoulos was the recipient of a Kritski Fellowship, in 1986 and 1987. He was the finalist for the Anton Philips Award for Best Student Paper at the IEEE Robotics and Automation Conference, in 1991. He was a Mc Knight Land-Grant Professor at the University of Minnesota, from 1995 to 1997, and has received the National Science Foundation's (NSFs) Research Initiation and Early Career Development Awards. Finally, he has received grants from the Defense Advanced Research Projects Agency (DARPA), Sandia National Laboratories, NSF, the United States Department of Transportation (USDOT), the Minnesota Department of Transportation (MN/DOT), Honeywell, and 3M.