Multiple Vehicle Detection and Tracking in Hard Real-Time *

Margrit Betke Esin Haritaoglu Larry S. Davis

University of Maryland
Institute for Advanced Computer Studies
College Park, MD 20742
Phone +1-301-405-6716, Fax +1-301-314-9658, E-mail:betke@umiacs.umd.edu

Abstract

A vision system is developed that recognizes and tracks multiple cars from sequences of gray-scale images taken from a moving car in hard real-time. The recognition method is based on feature detection, on-line deformable template matching, and temporal differencing. The vision system utilizes the hard real-time operating system Maruti which guarantees that the timing constraints on the various processes of the vision system are satisfied. The dynamic creation and termination of tracking processes optimizes the amount of computation resources spent and allows fast real-time recognition and tracking of multiple cars as demonstrated over a large number of image frames.

1 Introduction

We have developed a vision system that recognizes and tracks cars in hard real-time from sequences of gray-scale images taken from a moving car. We are interested in applications that improve traffic safety, for example, for camera-assisted or vision-guided vehicles. Such vehicles must react to dangerous situations immediately. This requires the supporting vision system not only to be extremely fast, but also to be guaranteed to react within a fixed time frame. Therefore, we use a hard real-time system that can predict in advance how long its computations take.

Recognizing and tracking objects in images taken by a *moving* camera (or fixed camera within a moving car) is much more challenging than real-time tracking with a *stationary* camera. Not only is there motion of the objects in the images, but also relative motion between the camera, the objects, and the environment. Our method uses the relative motion between the camera-assisted car and its environment to detect potential cars. To handle cases where there is little relative motion, our method searches for features that are typical for cars. Recognition of a car is verified if an objective function yields a high value. The objective function defines how likely it is that an object with certain parameters is a car. The objective function combines evaluating the history of tracking a potential car with the correlation of the potential car with a deformable template of a car created on-line using the method described in Ref. [1].

Various approaches for recognizing and/or tracking cars have been suggested in the literature, for example, detecting symmetry points [6], approximating optical flow [9, 2], exploiting binocular stereopsis [3, 4], matching templates [4], and training a neural net [5]. Related problems are autonomous convoy driving (e.g., [8]) and lane detection (e.g.,[3]). Unlike some methods described in the literature, our vision system can track more than one car at a time. In addition, it does not need any initialization by a human operator, but recognizes the cars it tracks automatically. Our method also does not rely on having to estimate road parameters (as does Ref. [3]).

Unlike other methods [2, 3, 4, 5, 6, 9], our vision system processes the video data in real-time without any specialized hardware. All we need is a consumer video camera and a low-cost PC. Simplicity is the key to the real-time performance of our method. We developed a system that is simple enough to be fast, but sophisticated enough to work robustly.

Most of the related research aims at "virtual real-time" performance, i.e., fast processing without timing guarantees. On the contrary, we utilize the advantages of Maruti, a hard real-time operating system developed at the University of Mary-

*The work was supported by ARPA, ONR, and the Philips Labs, grants N00014-95-1-0521, N6600195C8619, DASG-60-92-C-0055.

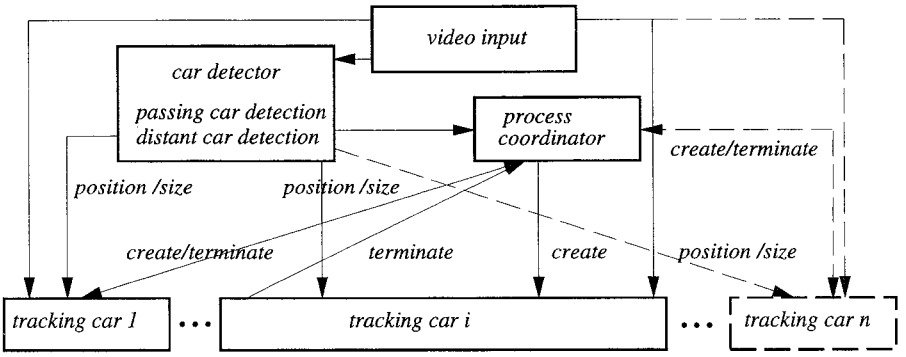

Figure 1: The hard real-time vision system for car detection and tracking.

land [7]. Maruti finds an efficient schedule for allocating resources to the various processes that search for, recognize, and track the cars. Maruti's scheduling guarantees that the required deadlines are met. It reserves the required resources for each task prior to execution.

This paper is organized as follows. An overview of the vision system is given in Section 2. Section 3 summarizes the features of the hard real-time operating system that are important to our work. The component of our vision system that detects cars is described in Section 4 and the component that tracks cars is described in Section 5. Section 6 reports our experimental results and Section 7 summarizes our conclusions.

2 Vision system overview

Given an input of a video sequence taken from a moving car, the vision system outputs an on-line description of the location and size of other cars in the environment.

The vision system contains three main components: the car detector, the process coordinator, and the tracker (see Figure 1). Once the car detector recognizes a potential car in an image, the process coordinator creates a tracking process for each potential car and provides the tracker with information about the size and location of the potential car. For each tracking process, the tracker analyzes the history of the tracked areas in the previous image frames and determines how likely it is that the area in the current image contains a car. If it contains a car with high probability, the tracker outputs the location and size of the hypothesized car in the image. If the tracked image area contains a car with very low probability, the process terminates. This dynamic creation and termination of tracking processes optimizes the amount of computation resources spent.

3 The hard real-time system

The ultimate goal of our vision system is to provide a car control system with a sufficient analysis of its changing environment, so that it can react to a dangerous situation immediately. Whenever a physical system like a car control system depends on complicated computations, such as are carried out by our vision system, the timing constraints on these computations become important. A "hard real-time system" guarantees – prior to any execution – that the system will react in a timely manner. In order to provide such a guarantee, a hard real-time system analyzes the timing and resource requirements of each computation task. The temporal correctness of the system is ensured if a feasible schedule can be constructed. The scheduler has explicit control over when a task is dispatched.

Our vision system utilizes the hard real-time system Maruti [7]. Maruti is a "dynamic hard real-time system" that can handle on-line requests. It either schedules and executes them if the resources needed are available, or rejects them. We implemented the processing of each image frame as a periodic task consisting of several subtasks (e.g., distant car detection, car tracking). Since Maruti is also a "reactive system," it allows switching tasks on-line. For example, our system could switch from the usual cyclic execution to a different operational mode. This supports our ultimate goal to improve traffic safety: The car control system could react within a guaranteed time frame to an on-line warning by the vision system which may have recognized a dangerous situation.

The Maruti Programming Language is a high-level language based on C with additional constructs for timing and resource specifications. Our programs are developed in the Maruti virtual runtime environment within UNIX. The hard real-time

platform of our vision system runs on a PC.

4 Car recognition

The input data of the vision system consists of image sequences taken from a camera-assisted moving car. The camera is mounted inside the car pointing towards the windshield. It takes pictures of the environment in front of the car, i.e., the road, other cars, trees next to the road, etc. Recognition of objects that suddenly enter the scene is difficult. For example, the continuously changing landscape along the road and the various lighting conditions that depend on the time of day and weather are not known in advance. Cars and trucks come into view with very different speeds, sizes, and appearances.

To facilitate robust and fast recognition of cars, we distinguish between recognizing cars that appear in the field of view after having passed the camera from behind, and cars that appear in the far distance in the front. Once a car is recognized, it is tracked in subsequent image frames until it moves out of sight.

4.1 Recognizing passing cars

When other cars pass the camera-assisted car, they are usually nearby and therefore cover a large portion of the image frames. They cause large brightness changes in such image portions over a small number of frames. We can exploit these facts to detect and recognize passing cars.

Large brightness changes over a small number of frames are detected by differencing the current image j frame from an earlier frame k and checking if the sum of the absolute brightness differences exceeds a threshold in an appropriate region of the image (see left of Fig. 2). Region R of image sequence $I(x, y)$ in frame j is hypothesized to contain a passing car if $\sum_{x,y \in R} |I_j(x, y) - I_{j-k}(x, y)| > \theta$ where θ is a fixed threshold.

The car motion from one image to the next is approximated by a shift of i/d pixels where d is the number of frames since the car has been detected, and i is a constant that depends on the frame rate. (For a car that passes the camera-assisted car from the left, for example, the shift is up and right, decreasing from frame to frame.)

If large brightness changes are detected in consecutive images, a grayscale template T of a size corresponding to the hypothesized size of the passing car is created from a model image M. It is correlated with the image region R that is hypothesized to contain a passing car. The normalized sample correlation coefficient r is used as a measure of how well region R and template image T correlate or match. A high correlation coefficient verifies that a passing car is detected. The model image M is chosen from a set of models that contains images of the rear of different kinds of vehicles. The average grayscale value in region R determines which model M is deformed into template T (see right of Fig.2). Sometimes the correlation coefficient is too low to be meaningful, even if a car is actually found (e.g., $r \leq 0.2$). In this case, we do not base a recognition decision on just one image, but instead use the results for subsequent images as described in the next sections.

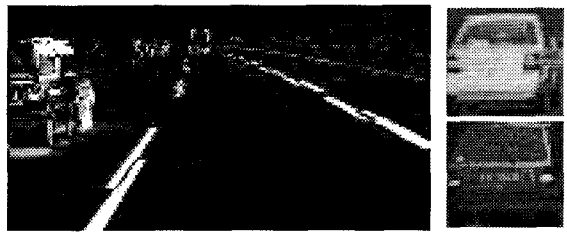

Figure 2: On the left, a passing car is detected in a difference image. On the right, two model images.

4.2 Recognizing distant cars

Cars that are being approached by the camera-assisted car usually appear in the far distance as rectangular objects. Generally, there is very little relative motion between such cars and the camera-assisted car. Therefore, any method based only on differencing image frames will fail to detect these cars. Therefore, we use a feature-based method to detect distant cars. We look for rectangular objects by evaluating horizontal and vertical edges in the images. The horizontal edge map $H(x, y, t)$ and the vertical edge map $V(x, y, t)$ are defined by a finite difference approximation of the brightness gradient and subsequent thresholding. Since the edges along the top and bottom of the rear of a car are more pronounced in our data, we use different thresholds for horizontal and vertical edges (the threshold ratio is 7:5).

Due to our real-time constraints, our recognition algorithm consists of two processes, a coarse and a refined search. The refined search is only employed for small regions of the edge map, while the coarse search is used over the whole image frame. The coarse search determines if the refined search is necessary. It divides the edge map into small horizontal regions that are processed separately. In each region a counting method checks if the edge map contains prominent, i.e., long, uninterrupted

edges. Whenever this is the case in some region, the refined search process is started in that region. Since the coarse search takes a substantial amount of time (because it processes the whole image frame), it is only called every 10th frame.

In the refined search, the vertical and horizontal projection vectors $\mathbf{v}$ and $\mathbf{w}$ of the horizontal and vertical edges H and V in the region are computed as follows (see also Fig. 3):

$\mathbf{v} = (\sum_{x=x_1}^{x_m} H(x, y_1, t), \ldots, \sum_{x=x_1}^{x_m} H(x, y_n, t), t)$,
$\mathbf{w} = (\sum_{y=y_1}^{y_n} V(x_1, y, t), \ldots, \sum_{y=y_1}^{y_n} V(x_m, y, t), t)$.

The threshold for selecting projection values is one half of the largest projection value in each direction; $\theta_{\mathbf{v}} = \frac{1}{2}\max\{v_i | 1 \leq i \leq m\}$, $\theta_{\mathbf{w}} = \frac{1}{2}\max\{w_j | 1 \leq j \leq n\}$. The projection vector of the vertical edges is searched from the left and also from the right until a vector entry is found that lies above the threshold in both cases. The positions of these entries determine the position of the left and right side of the potential object. Similarly, the projection vector of the horizonal edges is searched from the top and from the bottom until a vector entry is found that lies above the threshold in both cases. The positions of these entries determine the position of the top and bottom side of the potential object.

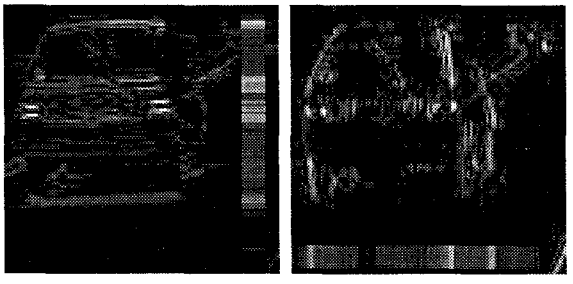

Figure 3: In the left image, the horizontal edge map H of a car is projected onto column $\mathbf{v}$ on the right. In the right image, the vertical edge map V is projected onto row $\mathbf{w}$ on the bottom.

To verify that the potential object is a car, an objective function is evaluated as follows. First, the aspect ratio of the horizontal and vertical sides of the potential object is computed to check if it is close enough to 1 to imply that a car is detected. Then the car template is correlated with the potential object marked by the four corner points in the image. If the correlation yields a high coefficient value, the object is recognized as a car. If the "history" of the potential object gives additional evidence for the existence of a car, our objective function yields a high value and outputs that a car is detected, where it is located, and what its size is (which can be used as a clue for its distance to the camera-assisted car).

5 Tracking cars

The process coordinator creates a separate tracking process for each recognized car. It uses the initial parameters for the position and size of the car that are determined by the car detector and ensures that no other process is tracking the same car. A "tracking window" is created that contains the car and is used to evaluate edge maps and templates in subsequent image frames. In each frame a refined search of the tracking window provides new estimates of the four sides of the rear of the car and uses them to determine the new window boundaries. In every 10th frame the window is correlated with a car template that is created on-line to the appropriate size to verify that the tracked object is still the car originally detected. In each frame, the objective function is evaluated. It checks the aspect ratio, size, and, if applicable, correlation for the current tracking window. It assigns credits or penalties depending on how likely it is that the object tracked is a car. We call a tracking process "stable" if the credit assignments have been high for several image frames. For stable processes, the history of the tracking window is used to update position and size parameters if the current parameters yield penalties. For example, this is the case if the rear window is mistaken to be the whole rear of the car. This can be determined easily, because the aspect ratio of the rear window is much larger than previous aspect ratios of the process. The process is adjusted by increasing the height of the window towards the bottom of the car.

The process coordinator ensures that two tracking processes do not track objects that are too close to each other in the image. This may happen if a car passes another car and eventually occludes it. In this case, the process coordinator terminates one of these processes. Processes that track potential cars for which the objective function yields low values terminate themselves. This ensures that oncoming traffic and objects that appear on the side of the road, such as traffic signs, are not mistakenly recognized to be cars.

6 Experimental results

Our data consists of almost 3000 images taken by a video camera from a moving car on an American and a German highway. The images are evaluated

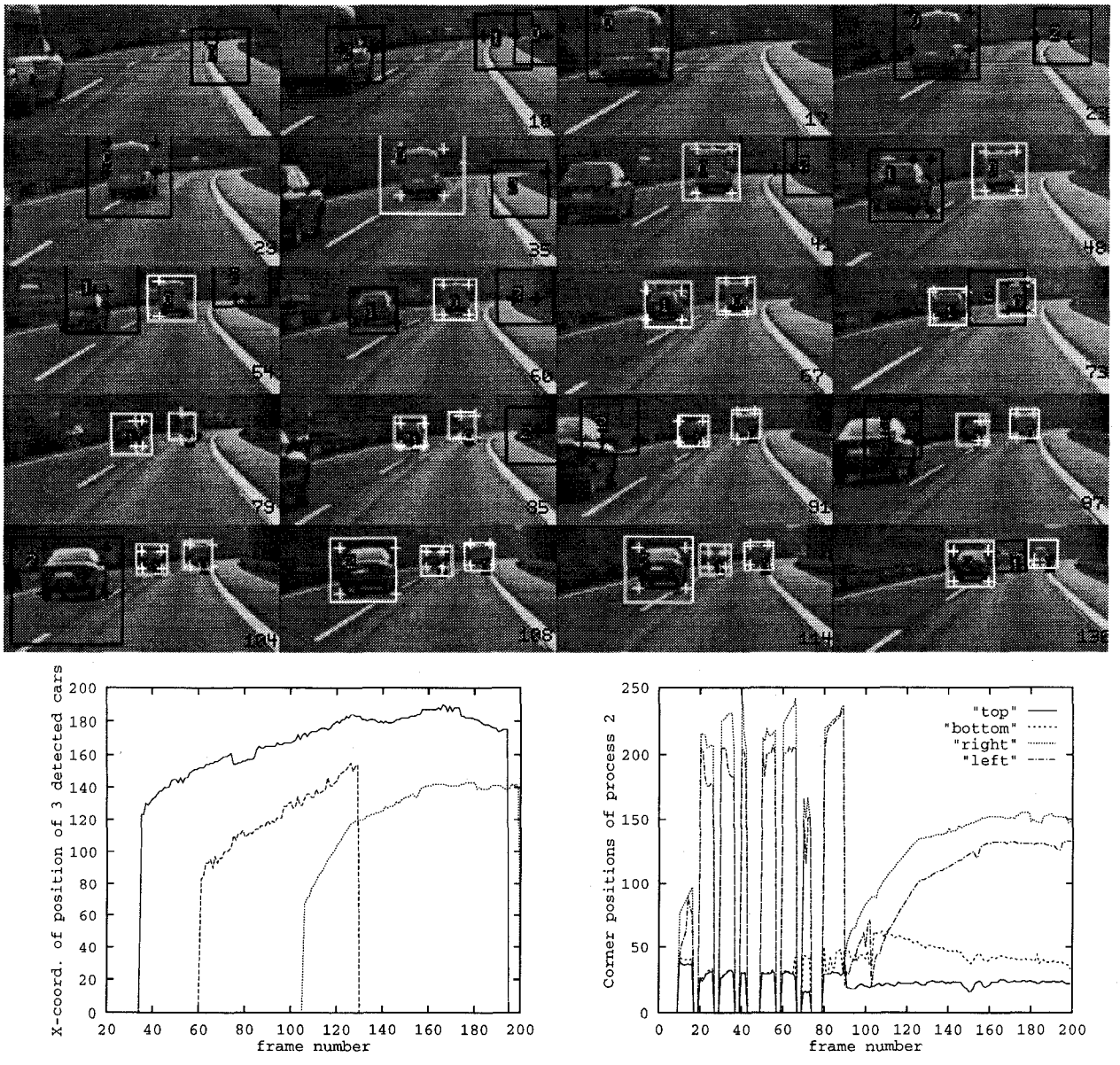

Figure 4: Example of an image sequence where three cars are recognized and tracked. On the top, images are shown with their frame number on the lower right corner. The black rectangle shows regions wherein moving objects are detected. The corner of these objects are shown as crosses. The rectangle and crosses turn white when the system recognizes these objects to be cars.
The graph at the bottom left illustrates the x-coordinate of the position of the three recognized cars. The graph at the bottom right illustrates the y-coordinates of the top and bottom and the x-coordinates of the left and right corners of the tracking window of process 2. Note that in the beginning process 2 is created several times to track a potential car, but is quickly terminated, because the tracked object is recognized not to be a car. At frame 108 process 2 starts tracking the third car.

Average Computation Time

Image area	Method	Time
complete image (every 10th frame)	edges	88 ms
	horizontal lines	35 ms
window	differencing	1.2 ms
	passing cars	19 ms
	template match	105 ms
	tracking cars	2-35 ms

in both hard and virtual real-time in the laboratory. Processing each image frame takes 98 ms on average; thus, we achieve a frame rate of approximately 10.2 frames per second. Note that the numbers are averages, because some images are processed quickly (if only a small car is being tracked), others take longer (e.g., if the complete image is searched). The average amount of time per process is summarized in the table above.

We obtain two sets of results for the German highway data by visual inspection (see also Fig. 4). In the first set every frame is processed, and in the second set (given in parentheses) every fourth frame is processed. A total of 21 (18) cars are detected and tracked successfully. The image size of the detected cars is between 10x10 and 80x80 pixels. On average, each detected car is tracked during 93 (27) frames. The car detector notices brightness changes immediately when a passing car appears in the image. The car is tracked by a window of the correct size after 14 (4) frames. It takes another 7 (3.75) frames on average until the car tracking has become stable and the detector verifies that it found a car. We encounter 3 (2) false alarms during a total of 58 (19) frames, i.e., scenarios where the detector outputs that a car is detected at a location, but the image does not contain a car at that location. We obtain similar results for the American data. However, since the driving speed on American highways is much slower than on German highways, there is less motion detectable from one image frame to the next.

Our system is robust unless it encounters uncooperative conditions, e.g., little brightness contrast between cars and environment, very bumpy roads, sharp tree shadows cast onto the highway, or congested traffic.

7 Conclusions

We have developed and implemented a hard real-time vision system that recognizes and tracks multiple cars from sequences of gray-scale images taken from a moving car.

To our knowledge, this is the first attempt to recognize and track multiple vehicles from a moving car in hard real-time that is described in the literature. The hard real-time system is essential for our ultimate goal to improve traffic safety. The control system of the camera-assisted car must be able to react to an on-line warning by the vision system which may have recognized a dangerous situation. Such a reaction must meet timing deadlines that can only be guaranteed by a hard real-time system.

Unlike several methods described in the literature, our vision system can track more than one car at a time, recognizes the cars automatically, and relies only on simple low-cost hardware. Extensions of our work will address driving situations with difficult lighting conditions and congested traffic.

Our hard real-time vision system could be used not only to track vehicles, but is also applicable to other motion analysis problems, for example, robot navigation or recognition of moving people for surveillance purposes.

Acknowledgment

We thank the German Department of Transportation for providing the European data and Prof. Ashok Agrawala and his group for support with the Maruti operating system.

References

[1] M. Betke and N. C. Makris. Fast object recognition in noisy images using simulated annealing. In *ICCV*, pages 523–530, Cambridge, MA, 1995. Also MIT-AI 1510.

[2] W. Krüger, W. Enkelmann, and S. Rössle. Real-time estimation and tracking of optical flow vectors for obstacle detection. In *Intelligent Vehicles*, pages 304–309, 1995.

[3] Q.-T.and J. Weber Luong, D. Koller, and J. Malik. An integrated stereo-based approach to automatic vehicle guidance. In *ICCV*, pages 52–57, Cambridge, MA, 1995.

[4] Y. Ninomiya, S. Matsuda, M. Ohta, Y. Harata, and T. Suzuki. A real-time vision for intelligent vehicles. In *Intelligent Vehicles*, pages 101–106, 1995.

[5] D. Noll, M. Werner, and W. von Seelen. Real-time vehicle tracking and classification. In *Intelligent Vehicles*, pages 101–106, 1995.

[6] U. Regensburger and V. Graefe. Visual recognition of obstacles on roads. In *IROS*, pages 982–987, 1994.

[7] M. C. Saksena, J. da Silva, and A. K. Agrawala. Design and implementation of Maruti-II. In *Principles of Real-Time Systems*. Prentice-Hall, 1994.

[8] H. Schneiderman, M. Nashman, A. J. Wavering, and R. Lumia. Vision-based robotic convoy driving. *Machine Vision and Applications*, 8(6):359–364, 1995.

[9] S. M. Smith. ASSET-2: Real-time motion segmentation and shape tracking. In *ICCV*, pages 237–244, Cambridge, MA, 1995.