CAR-TR-834 N00014-95-1-0521
CS-TR-3667 DASG-60-92-C-0055
UMIACS-TR-96-52 July 1996

Multiple Vehicle Detection and Tracking in Hard Real Time

Margrit Betke, Esin Haritaoglu and Larry S. Davis.

Computer Vision Laboratory
Center for Automation Research and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742-3275

Abstract

A vision system has been developed that recognizes and tracks multiple vehicles from sequences of gray-scale images taken from a moving car in hard real time. Recognition is accomplished by combining the analysis of single image frames with the analysis of the motion information provided by multiple consecutive image frames. In single image frames, cars are recognized by matching deformable gray-scale templates, by detecting image features, such as corners, and by evaluating how these features relate to each other. Cars are also recognized by differencing consecutive image frames and by tracking motion parameters that are typical for cars.

The vision system utilizes the hard real-time operating system Maruti which guarantees that the timing constraints on the various vision processes are satisfied. The dynamic creation and termination of tracking processes optimizes the amount of computational resources spent and allows fast detection and tracking of multiple cars. Experimental results demonstrate robust, real-time recognition and tracking over thousands of image frames.

The support of ARPA, ONR, and Philips Laboratories under Contracts N00014-95-1-0521 and DASG-60-92-C-0055 is gratefully acknowledged.

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. Recognizing and tracking objects in images taken by a *moving* camera (or a fixed camera carried by 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 evaluation of the history of tracking a potential car with correlation of the potential car with a deformable template of a car created on-line using the method described in Ref. [3].

Various approaches for recognizing and/or tracking cars from a moving camera have been suggested in the literature – for example, detecting symmetry [33, 29, 4], approximating optical flow [32, 13, 15, 22], exploiting binocular or trinocular stereopsis [23, 4, 26, 18], matching templates [26], and training a neural net [27]. Related problems are autonomous convoy driving [31, 11], road detection and following [1, 2, 5–9, 12, 16, 19, 20, 23, 28], lane transition [17], and image stabilization [25]. (Since some of these areas have been researched extensively, we only refer to some of the more recent approaches.) There has also been work on tracking vehicles using stationary cameras (e.g., [21, 14, 10]). A collection of articles on vision-based vehicle guidance can be found in Ref. [24].

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. [23]). Unlike other methods [4, 13, 22, 23, 26, 27, 29, 32], our vision system processes the video data in real time without any specialized hardware. All we need is an ordinary 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.

We are interested in applications that improve traffic safety 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. Most of the related research aims at "virtual real-time" performance, i.e., fast processing without timing guarantees. In contrast, we utilize the advantages of Maruti, a hard real-time operating system developed at the University of Maryland [30]. 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 locations and sizes of other cars in the images. This description could be used to estimate the positions of the cars in the environment and their distances from the camera-assisted car.

The vision system contains three main components: the car detector, the process coordinator, and the tracker. The system is illustrated in 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 computational resources spent.

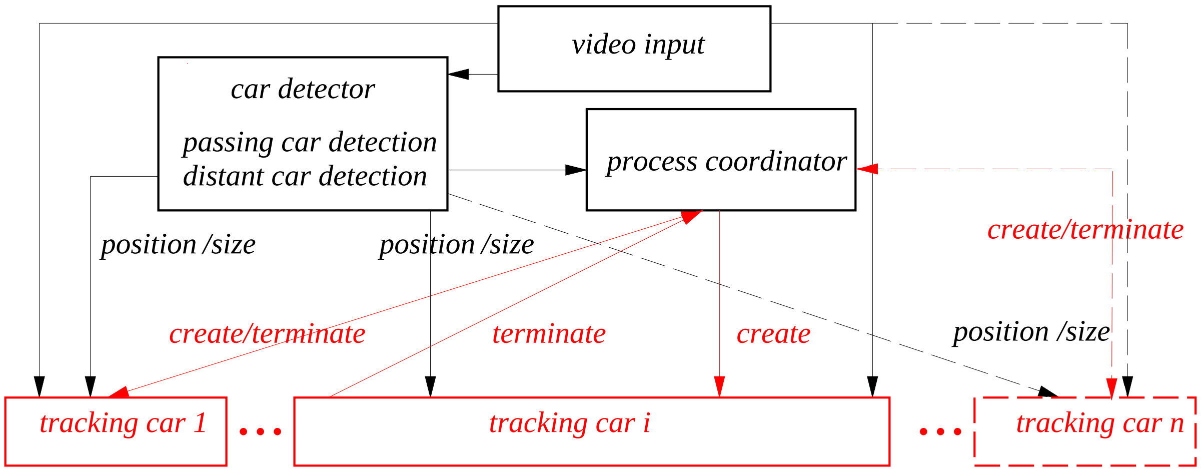


Figure 1: The hard real-time vision system for car detection and tracking.

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 [30]. 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 of improving 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.

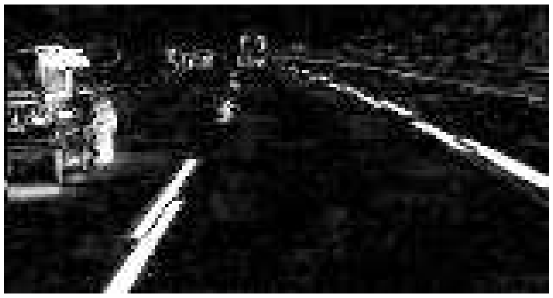
4 Car recognition

The input data of the vision system consists of image sequences taken from a moving car. The camera is mounted inside the car just behind the windshield. It takes images of the environment in front of the car, for example, the road, other cars, and trees next to the road. The task of the system is to distinguish the cars from other stationary and moving objects in the images and recognize them as cars. This is a challenging task, because 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. Recognition of objects that suddenly enter the scene is difficult. 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 front of the camera. 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 large portions of the image frames. They cause large brightness changes in such image portions over small numbers of frames. We can exploit these facts to detect and recognize passing cars. The image in Figure 2(a) illustrates the brightness difference caused by a passing car.


(a)

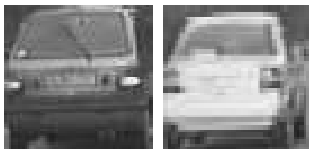

(b)

Figure 2: (a) A passing car is detected by image differencing. (b) Two model images.

Large brightness changes over small numbers 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. 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 gray-scale template T of a size corresponding to the hypothesized size of the passing car is created from a model image M using the method described in Ref. [3]. 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:

$$r = \frac{p_T \sum_{x,y} R(x,y)T(x,y) - \left(\sum_{x,y} R(x,y)\right)\left(\sum_{x,y} T(x,y)\right)}{\sqrt{p_T \sum_{x,y} R(x,y)^2 - \left(\sum_{x,y} R(x,y)\right)^2}\sqrt{p_T \sum_{x,y} T(x,y)^2 - \left(\sum_{x,y} T(x,y)\right)^2}}$$

where p_T is the number of pixels in the template image T that have nonzero brightness values. The normalized correlation coefficient is dimensionless, and $|r| \leq 1$. Region R and template T are perfectly correlated if $r = 1$. 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 gray-scale value in region R determines which model M is deformed into template T (see Fig. 2(b)). 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.

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. 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).

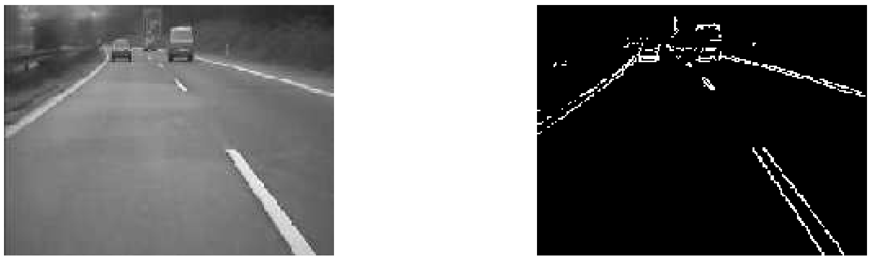


Figure 3: An example of an image with distant cars and its thresholded edge map.

Due to our real-time constraints, our recognition algorithm consists of two processes, a coarse and a refined search. The refined search is employed only 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 searches the thresholded edge maps for prominent (i.e., long, uninterrupted) edges. Whenever such edges are found in some image 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:

$$\mathbf{v} = (v_1, \ldots, v_m, t) = (\sum_{i=1}^{m} H(x_i, y_1, t), \ldots, \sum_{i=1}^{m} H(x_i, y_n, t), t)$$

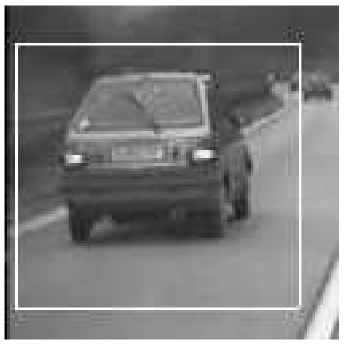
$$\mathbf{w} = (w_1, \ldots, w_n, t) = (\sum_{j=1}^{n} V(x_1, y_j, t), \ldots, \sum_{j=1}^{n} V(x_m, y_j, t), t).$$

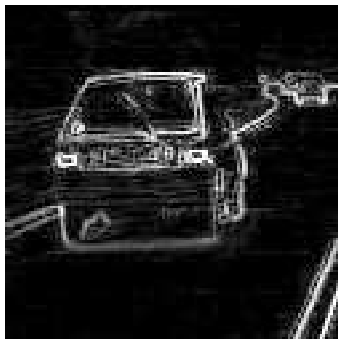
Figure 4 illustrates the horizontal and vertical edge maps H and V and their projection vectors $\mathbf{v}$ and $\mathbf{w}$. A large value for v_j indicates pronounced horizontal edges along $H(x, y_j, t)$. A large value for w_i indicates pronounced vertical edges along $V(x_i, y, t)$. The threshold for selecting large projection values is 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 starting 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 positions of the left and right sides of the potential object. Similarly, the projection vector of the horizonal edges is searched starting 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 positions of the top and bottom sides of the potential object.

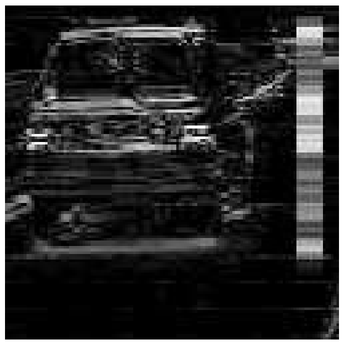
To verify that the potential object is a car, an objective function is evaluated as follows. First, the aspect ratio of the horizontal and vertical sides of the potential object is computed to check if it is close enough to 1 to imply that a car is detected. Then the car template is correlated with the potential object marked by the four corner points in the image. If the correlation yields a high value, the object is recognized as a car. The system outputs the fact that a car is detected, its location in the image, and its size.

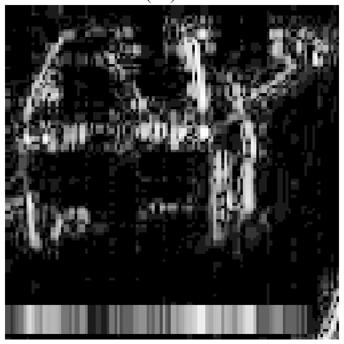
5 Car recognition based on tracking

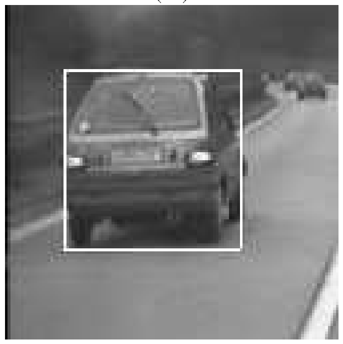
The previous sections described how vehicles can be recognized from a single image or from two consecutive images. However, immediate recognition from one or two images is very dif-


(a)


(b)


(c)


(d)


(e)

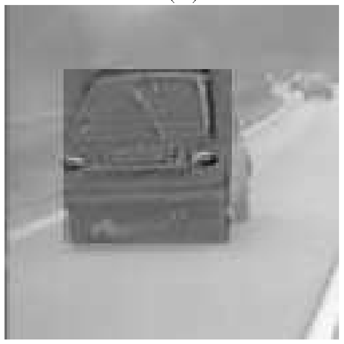

(f)

Figure 4: (a) An image with its marked search region. (b) The edge map of the image. (c) The horizontal edge map H of the image (enlarged). The column on the right displays the vertical projection values $\mathbf{v}$ of the horizontal edges computed for the search region marked in (a). (d) The vertical edge map V (enlarged). The row on the bottom displays the horizontal projection vector $\mathbf{w}$. (e) Thresholding the projection values yields the outline of the potential car. (f) A car template of the hypothesized size is overlaid on the image region shown in (e); the correlation coefficient is 0.74.

ficult and only works robustly under cooperative conditions (e.g., enough brightness contrast between vehicles and background). Therefore, if an object cannot be recognized immediately, our system evaluates several image frames and employs its tracking capabilities to recognize vehicles.

Figure 5 illustrates the tracking portion of our vision system. The process coordinator creates a separate tracking process for each potential car. It uses the initial parameters for the position and size of the potential car that are determined by the car detector and ensures that no other process is tracking the same image area. The tracker creates a "tracking window" that contains the potential car and is used to evaluate edge maps and templates in subsequent image frames. In each frame a refined search within the tracking window provides new estimates of the outline of the potential car and uses them to determine the new window boundaries. In every 10th frame a car template is created on-line and correlated with the object in the tracking window.

In each frame, the objective function evaluates how likely it is that the object tracked is a car. It checks the credit and penalty values associated with each process. Credit and penalty values are assigned depending on the aspect ratio, size, and, if applicable, correlation for the tracking window of the process. For example, if the normalized correlation coefficient is larger than 0.6 for a car template of $\geq 30 \times 30$ pixels, 10 credits are added to the accumulated credit of the process, and the accumulated penalty of the process is set to zero. If the normalized correlation coefficient for a process is negative, the penalty associated with this process is increased by 5 units. Aspect ratios between 0.7 and 1.4 are considered to be potential aspect ratios of cars; 1–3 credits are added to the accumulated credit of the process (the number increases with the template size). If the accumulated credit is above a threshold (10 units) and exceeds the penalty, it is decided that the tracked object is a car. These values for the credit and penalty assignments seem somewhat ad hoc, but their selection was driven by careful experimental analysis. Examples of accumulated credits and penalties are given in the next section in Figures 8 and 10.

Once a car is recognized, it is tracked robustly, even if the refined window search described in Section 4.2 does not find the correct outline of the car. This may happen, for example, if the rear window of the car is mistaken to be the whole rear of the car due to thresholding

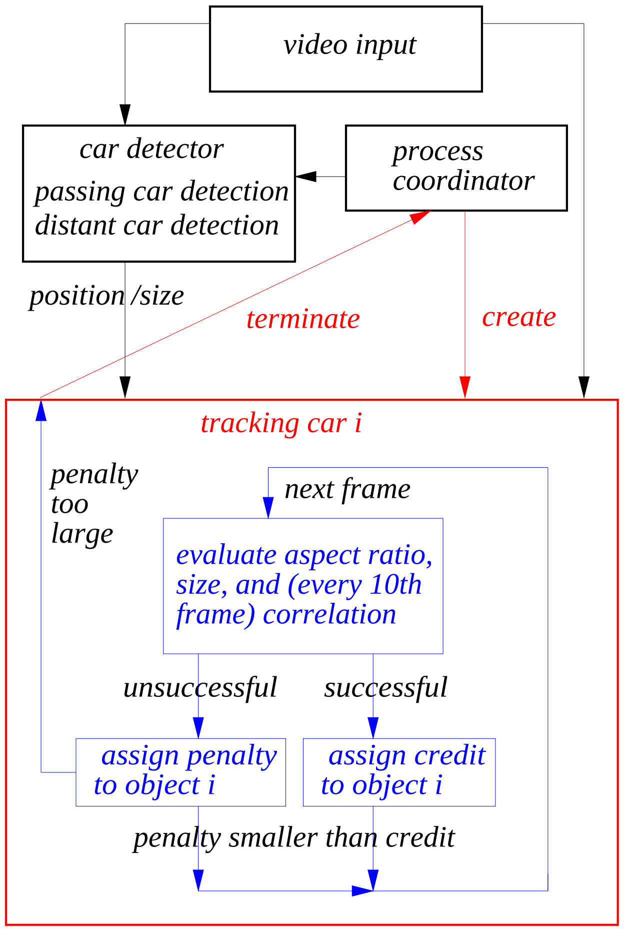


Figure 5: Tracking process i.

problems in the horizontal edge map. This can be determined easily, because the aspect ratio of the rear window is much larger than the aspect ratios of the car in preceding image frames. Thus, in the following image frame, the tracking window is adjusted by searching along the extended left and right side of the car for significant vertical edges. In particular, the pixel values in the vertical edge map that lie between the left bottom corner of the car and the left bottom border of the window, and the right bottom corner of the car and the right bottom border of the window, are summed and compared with the corresponding sum on the top. If the sum on the bottom is significantly larger than the sum on the top, the window is shifted towards the bottom (it still includes the top side of the car). Similarly, if the aspect ratio is too small, the correct positions of the car sides are found by searching along the extended top and bottom of the car for significant horizontal edges.

A window adjustment based on the aspect ratio is very useful for a number of reasons.

First, it captures the outline of a car, even if the refined search encounters thresholding problems due to low contrast between the car and the environment. Second, it supports recognition of passing cars that are not fully contained within the tracking window. Third, it compensates for the up and down motion of the camera due to uneven pavement. Finally, it ensures that the tracker does not lose a car even if the road curves. Figure 6 illustrates these cases and shows that, as a result of a shift of the tracking window, the cars are fully captured in the subsequent image frames.

A tracked car is no longer recognized if the penalty exceeds the credit by a certain amount (3 units). The tracking process then terminates. A process also terminates if not enough significant edges are found within the tracking window for several consecutive frames (although the credit may be much higher than the penalty as occurs in frame 200 in Figure 10). This ensures that a window that tracks a distant car does not "drift away" from the car and start tracking something else.

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. (For example, in Figure 9 process 1 is too close to process 2 in frame 41 and is terminated.)

Processes that track unrecognized objects terminate themselves, if the objective function yields low values for several consecutive image frames. This ensures that oncoming traffic or objects that appear on the side of the road, such as traffic signs, are not falsely recognized and tracked as cars (see, for example, process 0 in frame 4 in Figure 7).

6 Experimental results

Our data consists of more than 15,200 images taken by a video camera from a moving car on an American and a German highway. The images are evaluated in both hard and virtual real time in the laboratory. Figures 7 and 9 show how three cars are recognized and tracked in two image sequences. The graphs in Figures 8 and 10 illustrate how the tracking processes are analyzed. Table 1 provides some of the results of our vision system for data that was analyzed in hard real time. The system recognized and tracked a black van for 47 seconds.

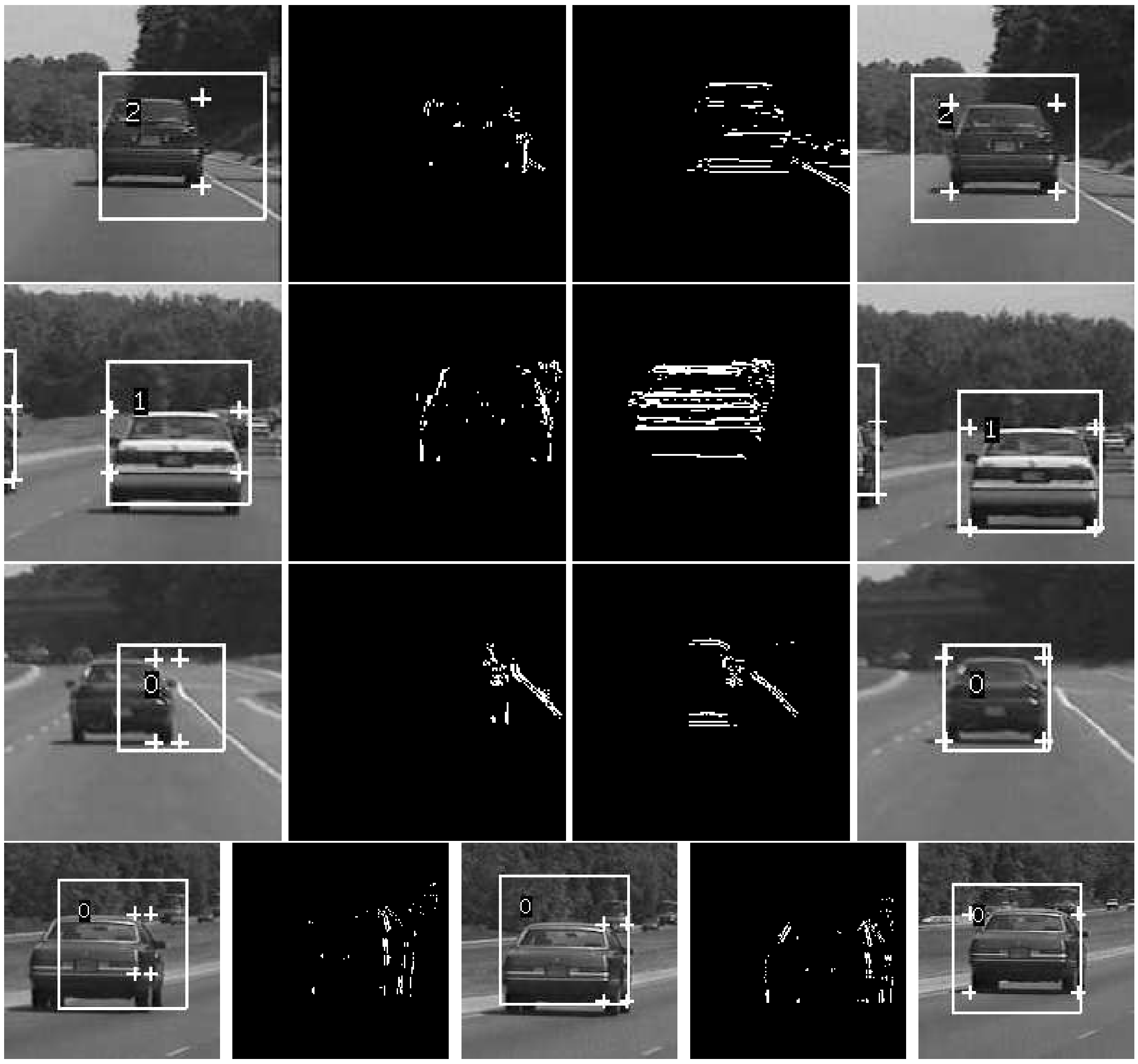


Figure 6: In the first row, the left and right corners of the car are found in the same position due to the low contrast between the sides of the car and background (first image). The vertical edge map resulting from this low contrast is shown in the second image. Significant horizontal edges (the horizontal edge map is shown in third image) are found to the left of the corners and the window is shifted to the left (fourth image).
In the second row, the window shift compensates for a ten-pixel downward motion of the camera due to uneven pavement.
In the third row, the car passed underneath a bridge, and the tracking process is slowly "recovering." The tracking window no longer contains the whole car and its vertical edge map is therefore incomplete (second image). However, significant horizontal edges (third image) are found to the left of the corners and the window is shifted to the left.
In the last row, a passing car is tracked incompletely. First its bottom corners are adjusted, then its left side.

The duration of tracking is shorter for the other sequences, because the tracked cars quickly disappear in the distance or are occluded by other cars. The hard real-time system does not provide control over exactly which image frame on the video tape is processed when our vision system is started. Therefore, we obtain slightly different recognition and tracking results each time our system is run. The timings in Table 1 are averages over 10 runs.

Table 1: Hard Real-Time Recognition

Vehicle	Duration of Recognition and Tracking
black van	47 s
white mid-size car 1	8 s
white compact car	13 s
station wagon	13 s
white mid-size car 2	17 s
white car	13 s

Processing each image frame takes 98 ms on average in hard real time; thus, we achieve a frame rate of approximately 10.2 frames per second (NTSC: processing on average every third frame; PAL: processing on average every second or third frame). Note that the numbers are averages, because some images are processed quickly (if only a small car is being tracked), while others take longer (e.g., if the complete image is searched). The average amount of processing time is summarized in Table 2.

Table 2: Average Computation Time for the German Data

Step	Image Area	Time
computing edge maps	complete image	88 ms
	(every 10th frame)	
computing horizontal lines		35 ms
differencing image frames	window	1.2 ms
detecting passing cars		19 ms
matching templates	(every 10th frame)	105 ms
tracking cars		2-35 ms

We obtain two sets of results for the German highway data by visual inspection using the virtual runtime environment (see also Table 3). In the first set every other frame is processed, and in the second set (given in parentheses) every eighth frame is processed. A total of 23 (20) out of 28 cars are detected and tracked successfully. The image size of the detected cars is between 10×10 and 80×80 pixels. On average, each detected car is tracked during 106 (31) frames. Then it usually is either occluded by other cars, or becomes too small. 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 (4.6) frames. It takes another 7 (3.7) 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 system detects a car, but the image does not contain a car at this location. One such false alarm is created by a traffic sign; it is shown in Figures 9 and 10.

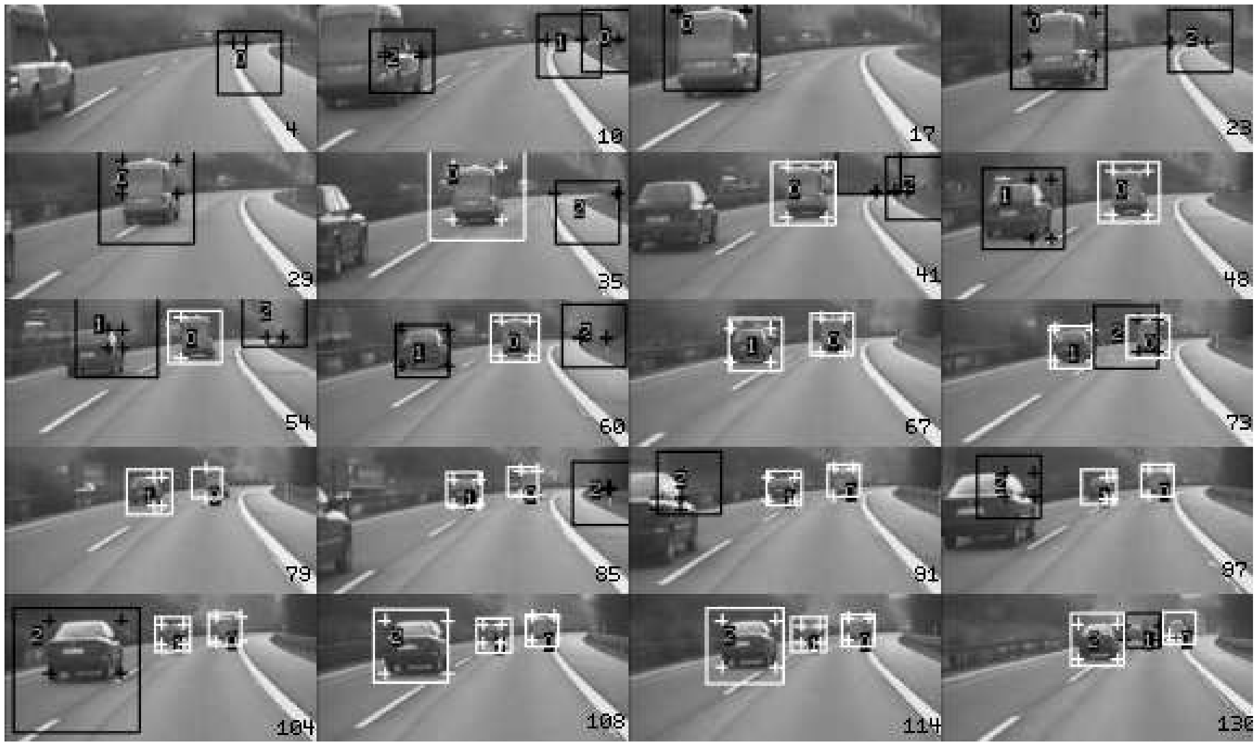
We obtain similar results for the American data. However, since the driving speed on American highways is much slower than on German highways, less motion is detectable from one image frame to the next. Therefore, we get good results from processing only every 10th frame. The results are summarized in Table 3.

Table 3: Recognition and Tracking Results

Data origin	German		American
total number (#) of frames	ca. 5572		9670
	2786×2	696×8	967×10
# of frames processed (in parentheses: results normalized for every frame)	every 2nd	every 8th	every 10th
detected and tracked cars	23	20	25
cars not tracked	5	8	3
size of detected cars (in pixels)	10×10–80×80	10×10–80×80	20×20–100×80
average # of frames during tracking	105.6 (211.2)	30.6 (245)	30.1 (301)
average # of frames until car tracked	14.4 (28.8)	4.6 (36.8)	4.5 (45)
average # of frames until stable detection	7.3 (14.6)	3.7 (29.8)	2.1 (21)
false alarms	3	2	3
# of frames during false alarm	58 (116)	19 (152)	22 (220)

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

Sequence 1



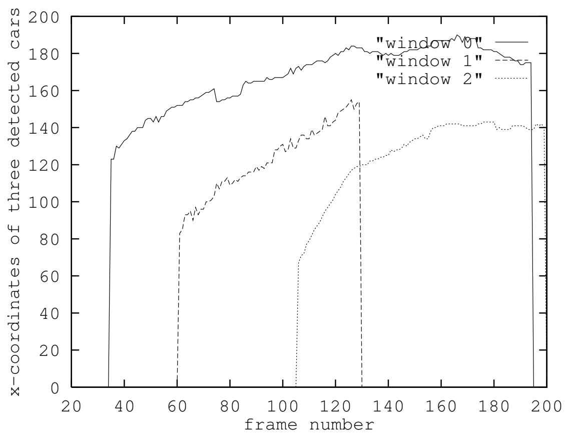


Figure 7: Example of an image sequence in which cars are recognized and tracked. On the top, images are shown with their frame numbers in their lower right corners. The black rectangles show regions within which moving objects are detected. The corners of these objects are shown as crosses. The rectangles and crosses turn white when the system recognizes these objects to be cars.
The graph on the bottom shows the x-coordinates of the positions of the three recognized cars.

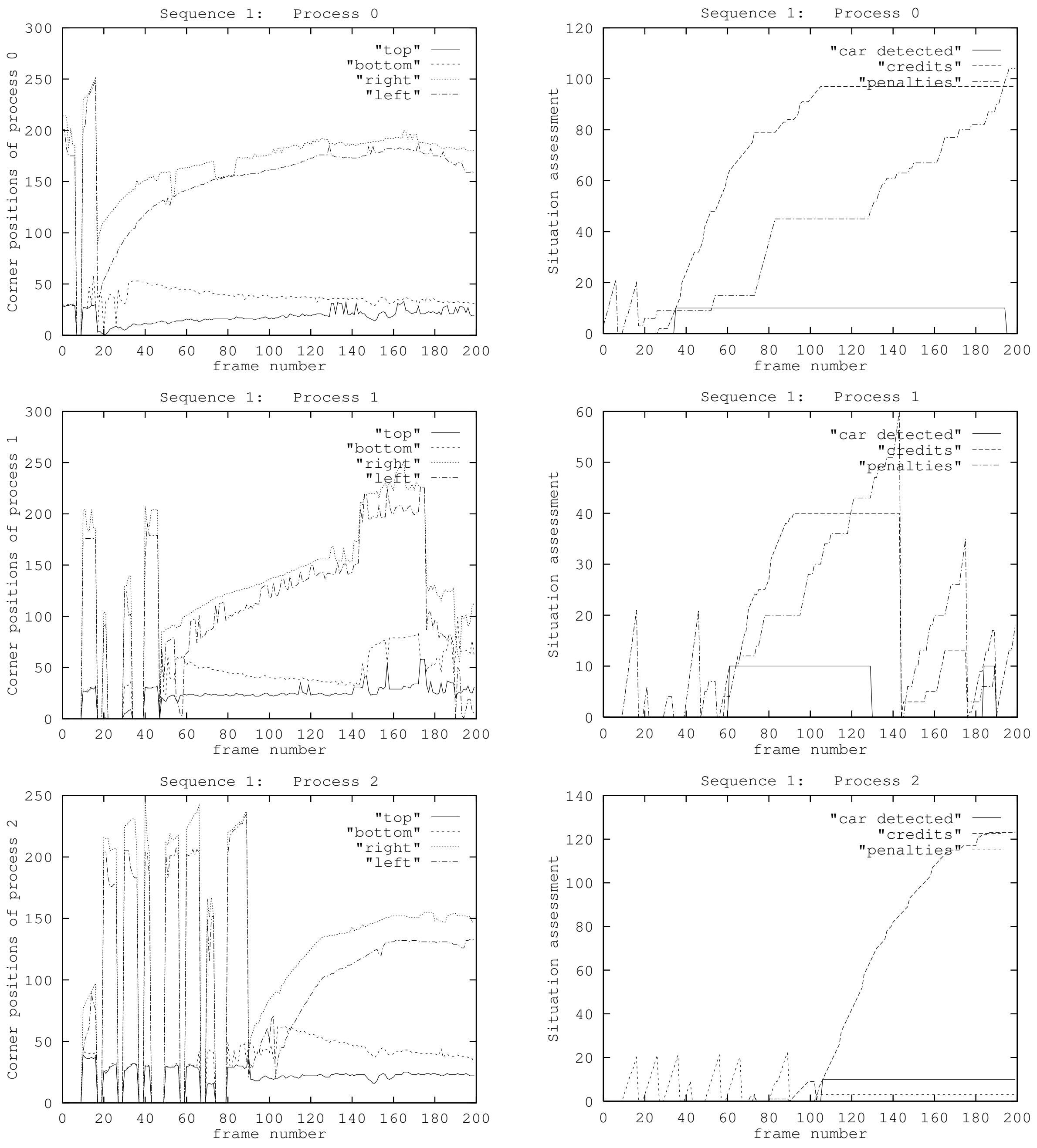
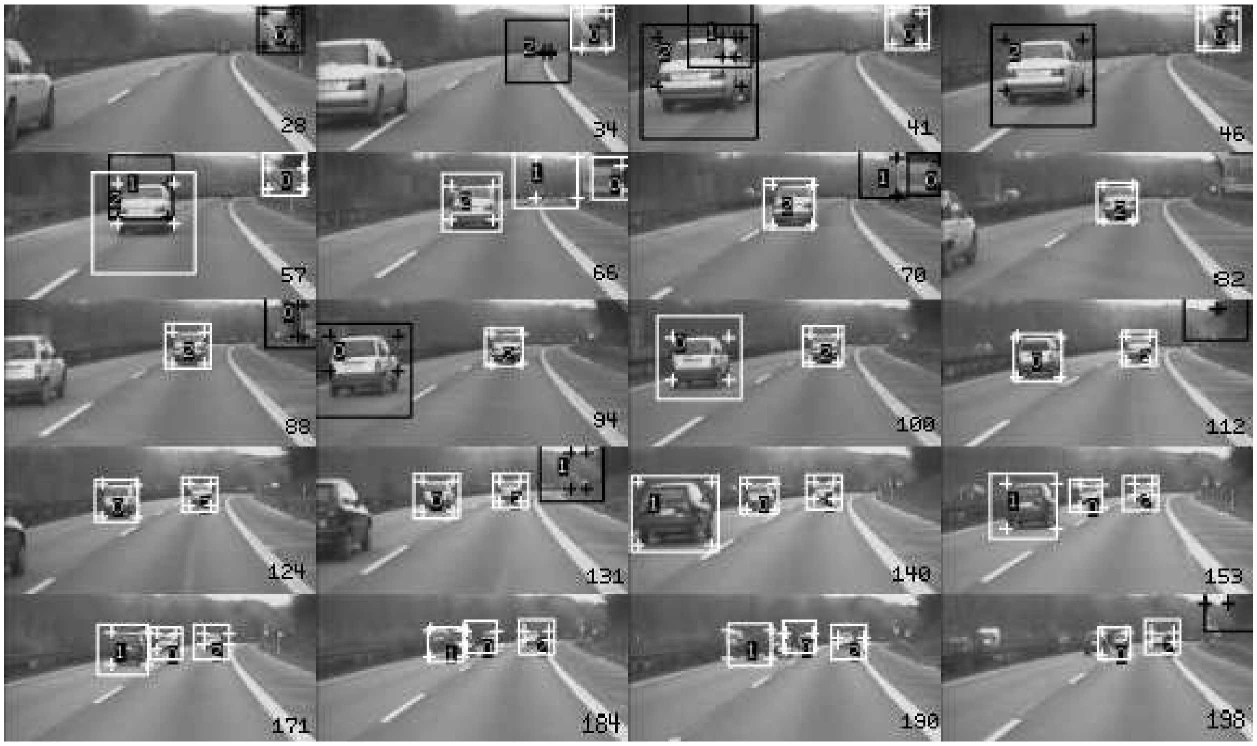


Figure 8: Analysis of Sequence 1 (Fig. 7): The graphs at the left illustrate the y-coordinates of the top and bottom and the x-coordinates of the left and right corners of the tracking windows of processes 0, 1, and 2, respectively. In the beginning, the processes are created several times to track potential cars, but are quickly terminated, because the tracked objects are recognized not to be cars. The graphs at the right show the credit and penalty values associated with each process and whether the process has detected a car. A tracking process terminates after its accumulated penalty values exceed its accumulated credit values. As seen in the top graphs and in Fig. 7, process 0 starts tracking the first car in frame 35 and does not recognize the right and left car sides properly in frames 54 and 79. Because the car sides are found to be too close to each other, penalties are assessed in frames 54 and 79.

Sequence 2



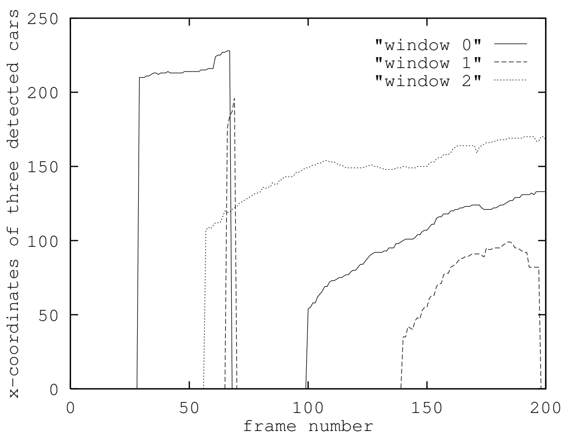


Figure 9: A second image sequence in which three cars are recognized and tracked. On the top, images are shown with their frame numbers in their lower right corners. The rectangles become white when the system recognizes the tracked objects to be cars. A traffic sign causes a false alarm and is tracked by process 0 in frames 34–66.
The graph on the bottom shows the x-coordinates of the positions of the three recognized cars and the false alarm.

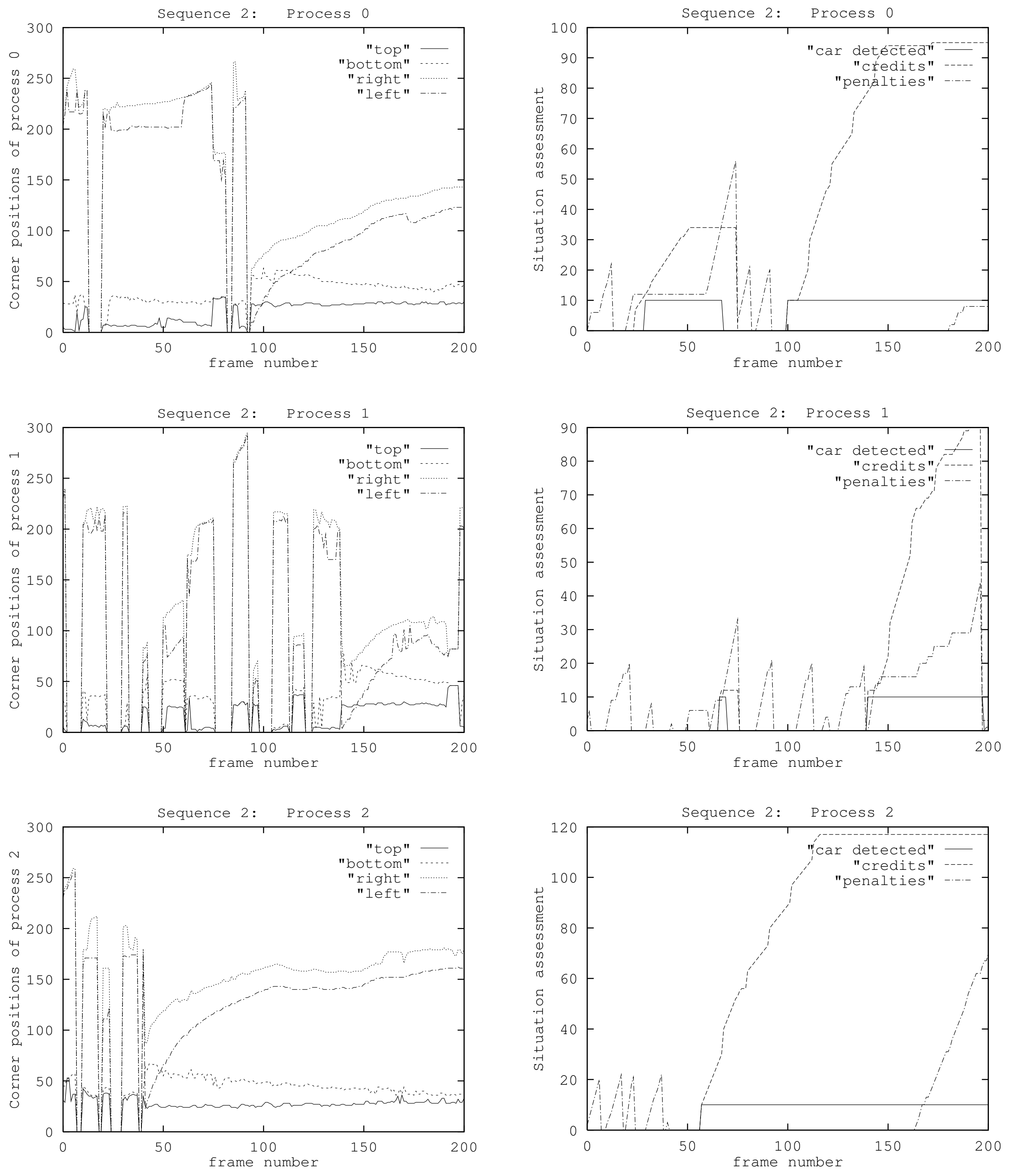


Figure 10: Analysis of Sequence 2 (Fig. 9): The graphs at the left illustrate the y-coordinates of the top and bottom and the x-coordinates of the left and right corners of the tracking windows of processes 0, 1, and 2, respectively. The graphs at the right show the credit and penalty associated with each process and whether the process has detected a car. As seen in the top graphs and in Fig. 9, process 0 tracks a traffic sign that is falsely recognized as the first car in frames 34–66. In frame 200, process 1 terminates tracking the second car, because not enough significant edges are found within window 1 for several antecedent frames (although the credits are much higher than the penalties).

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 has been described in the literature. The hard real-time system is essential for our ultimate goal of improving 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 tracks more than one car at a time, recognizes the cars automatically, and relies only on simple low-cost hardware, such as an ordinary video camera and a PC. Extensions of our work will address driving situations with difficult lighting conditions and congested traffic.

Our hard real-time vision system not only can be used to track vehicles, but also could be adapted to solve 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] Reinhold Behringer. Detection of discontinuities of road curvature changes by GLR. In *Proceedings of the Intelligent Vehicles Symposium*, pages 78–83, Detroit, MI, September 1995.

[2] Reinhold Behringer and Stefan Hötzel. Simultaneous estimation of pitch angle and lane width from the video image of a marked road. In *Proceedings of the IEEE/RSJ/GI*

International Conference on Intelligent Robots and Systems, pages 966–973, Munich, Germany, September 1994.

[3] Margrit Betke and Nicholas C. Makris. Fast object recognition in noisy images using simulated annealing. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 523–530, Cambridge, MA, June 1995. Also published as MIT AI Memo 1510.

[4] Stefan Bohrer, Thomas Zielke, and Volker Freiburg. An integrated obstacle detection framework for intelligent cruise control on motorways. In *Proceedings of the Intelligent Vehicles Symposium*, pages 276–281, Detroit, MI, September 1995.

[5] Alberto Broggi. Parallel and local feature extraction: A real-time approach to road boundary detection. *IEEE Transactions on Image Processing*, 4(2):217–223, February 1995.

[6] Alberto Broggi. Robust real-time lane and road detection in critical shadow conditions. In *Proceedings of the International Symposium on Computer Vision*, pages 353–359, Coral Gables, FL, November 1995.

[7] Jill D. Crisman and Charles E. Thorpe. SCARF: A color vision system that tracks roads and intersections. *IEEE Transactions on Robotics and Automation*, 9(1):49–58, February 1993.

[8] Ernst D. Dickmanns and Volker Graefe. Applications of dynamic monocular machine vision. *Machine Vision and Applications*, 1(4):241–261, 1988.

[9] Ernst D. Dickmanns and Birger D. Mysliwetz. Recursive 3-D road and relative ego-state recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):199–213, February 1992.

[10] M. Fathy and M. Y. Siyal. A window-based edge detection technique for measuring road traffic parameters in real-time. *Real-Time Imaging*, 1(4):297–305, 1995.

[11] U. Franke, F. Böttinger, Z. Zomoter, and D. Seeberger. Truck platooning in mixed traffic. In *Proceedings of the Intelligent Vehicles Symposium*, pages 1–6, Detroit, MI, September 1995.

[12] V. Gengenbach, H.-H. Nagel, F. Heimes, G. Struck, and H. Kollnig. Model-based recognition of intersection and lane structure. In *Proceedings of the Intelligent Vehicles Symposium*, pages 512–517, Detroit, MI, September 1995.

[13] Andrea Giachetti, Marco Cappello, and Vincent Torre. Dynamic segmentation of traffic scenes. In *Proceedings of the Intelligent Vehicles Symposium*, pages 258–263, Detroit, MI, September 1995.

[14] Sylvia Gil, Ruggero Milanese, and Thierry Pun. Combining multiple motion estimates for vehicle tracking. In *Lecture Notes in Computer Science. Vol. 1065: Proceedings of the 4th European Conference on Computer Vision*, volume II, pages 307–320. Springer-Verlag, Berlin, April 1996.

[15] Walter J. Gillner. Motion based vehicle detection on motorways. In *Proceedings of the Intelligent Vehicles Symposium*, pages 483–487, Detroit, MI, September 1995.

[16] Tetsuji Haga, Koichi Sasakawa, and Shin'ichi Kuroda. The detection of line boundary markings using the modified spoke filter. In *Proceedings of the Intelligent Vehicles Symposium*, pages 293–298, Detroit, MI, September 1995.

[17] Todd Jochem, Dean Pomerleau, and Charles Thorpe. Vision guided lane-transition. In *Proceedings of the Intelligent Vehicles Symposium*, pages 30–35, Detroit, MI, September 1995.

[18] L. Kaminski, J. Allen, I. Masaki, and G. Lemus. A sub-pixel stereo vision system for cost-effective intelligent vehicle applications. In *Proceedings of the Intelligent Vehicles Symposium*, pages 7–12, Detroit, MI, September 1995.

[19] Karl Kluge and Greg Johnson. Statistical characterization of the visual characteristics of painted lane markings. In *Proceedings of the Intelligent Vehicles Symposium*, pages 488–493, Detroit, MI, September 1995.

[20] Karl Kluge and Sridhar Lakshmanan. A deformable-template approach to line detection. In *Proceedings of the Intelligent Vehicles Symposium*, pages 54–59, Detroit, MI, September 1995.

[21] Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. In *Lecture Notes in Computer Science. Vol. 800: Proceedings of the 3rd European Conference on Computer Vision*, volume I, pages 189–196. Springer-Verlag, Berlin, May 1994.

[22] W. Krüger, W. Enkelmann, and S. Rössle. Real-time estimation and tracking of optical flow vectors for obstacle detection. In *Proceedings of the Intelligent Vehicles Symposium*, pages 304–309, Detroit, MI, September 1995.

[23] Q.-T. Luong, J. Weber, D. Koller, and J. Malik. An integrated stereo-based approach to automatic vehicle guidance. In *Proceedings of the International Conference on Computer Vision*, pages 52–57, Cambridge, MA, 1995.

[24] Ichiro Masaki, editor. *Vision-based Vehicle Guidance.* Springer-Verlag, Berlin, 1992.

[25] Carlos Morimoto, Daniel DeMenthon, Larry S. Davis, Rama Chellappa, and Randal Nelson. Detection of independently moving objects in passive video. In *Proceedings of the Intelligent Vehicles Symposium*, pages 270–275, Detroit, MI, September 1995.

[26] Yoshiki Ninomiya, Sachiko Matsuda, Mitsuhiko Ohta, Yoshihisa Harata, and Toshihika Suzuki. A real-time vision for intelligent vehicles. In *Proceedings of the Intelligent Vehicles Symposium*, pages 101–106, Detroit, MI, September 1995.

[27] Detlev Noll, Martin Werner, and Werner von Seelen. Real-time vehicle tracking and classification. In *Proceedings of the Intelligent Vehicles Symposium*, pages 101–106, Detroit, MI, September 1995.

[28] Dean Pomerleau. RALPH: Rapidly adapting lateral position handler. In *Proceedings of the Intelligent Vehicles Symposium*, pages 506–511, Detroit, MI, September 1995.

[29] Uwe Regensburger and Volker Graefe. Visual recognition of obstacles on roads. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, pages 982–987, Munich, Germany, September 1994.

[30] Manas C. Saksena, James da Silva, and Ashok K. Agrawala. Design and implementation of Maruti-II. In Sang Son, editor, *Principles of Real-Time Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1994. Also available as UMD CS-TR-3181, UMIACS TR-93-122.

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

[32] S. M. Smith. ASSET-2: Real-time motion segmentation and shape tracking. In *Proceedings of the International Conference on Computer Vision*, pages 237–244, Cambridge, MA, 1995.

[33] Thomas Zielke, Michael Brauckmann, and Werner von Seelen. Intensity and edge-based symmetry detection with an application to car-following. *CVGIP: Image Understanding*, 58(2):177–190, September 1993.