Tunnel Entrance Recognition for Video-based Driver Assistance Systems

Christopher Claus, Hoo Chang Shin and Walter Stechele
Institute for Integrated Systems
Technical University Munich
Arcisstr. 21, D-80290 Munich, Germany
Phone.: + 49 89 289-23862, Fax: + 49 89 289-28323, {Christopher.Claus, Walter.Stechele}@tum.de

Keywords: Tunnel entrance recognition, object recognition, driver assistance system, lane detection.

Abstract - What most matters for driver assistance systems is more reliability. From this view we introduce here our dependable tunnel entrance recognition system. Various algorithms are used to detect a tunnel entrance and mark it as Region of interest. In this paper we also present the results from experimental videos which we used to verify the correctness of our approach.

1. INTRODUCTION

Computer-based automotive systems are getting more and more important in the engineering field. The same applies to automotive driver assistance systems. Many researches are done to develop vision based driver assistance systems which is also in the focus of our project. It is much more challenging to recognize and track moving objects taken by a moving camera than that by a stationary one. The pedestrian detection in [1] shows about 90% of recognition rate and in [2] the vehicle recognition using supervised learning shows about 98.3% best classification rate. The authors in [3] have also done vehicle recognition using vertical and horizontal edges on the back face of vehicles as well as the edge of its shadow and its side boundaries. We have developed a vision system which recognizes and tracks a tunnel entrance from sequences of images which are taken from a camera, mounted on a moving car. We want to recognize a tunnel entrance if it is far away as well if it is near to us. That makes the tunnel entrance recognition difficult, because the shape of the tunnel entrance changes a lot dependant on the distance between driver and tunnel. For pedestrian recognition or vehicle recognition it is sufficient to recognize them, if they are in a certain maximal distance. When these are far from our vehicle its influence is not critical. So for recognition for those in a certain maximal distance, its shapes are restricted, which is not the case for tunnel entrance recognition.
Here we show our simple system for tunnel entrance recognition. It does not need segmentation processes which are usually needed for object recognition. This paper is organized in the following manner. In section 2, our system is overviewed. In section 3, the method for recognizing the tunnel entrance of our system is introduced. Experiments and processing time for each part of the system are analysed in S section 4. And finally section 5 summarizes our conclusions.

2. SYSTEM OVERVIEW

In Figure 1 the flow chart of our algorithm for tunnel entrance recognition is shown. At first the system looks for a tunnel entrance. When it is found and it is bigger than the tunnel entrance found in the previous frame, then the Region of Interest (ROI) is marked and the frame is saved in a buffer. When the tunnel entrance is not found although we know that the tunnel is near or the found tunnel entrance is smaller than the previous tunnel entrance, then we search for the tunnel entrance a second time. This is the case when the tunnel is not found because of 2 considerable cases: 1st case - Tunnel lights in the tunnel are found and these lights are seen dominant when the tunnel is near or some cars are positioned in front of the tunnel entrance. 2nd case - When the tunnel entrance is not found in the 2nd search process and when we know that tunnel entrance is near, then the tunnel entrance is located too near to be found with our method. When the tunnel is very close the boundary between dark regions and non-dark regions cannot be recognized. Our system figures out that our car is near to the tunnel from the memory history, the number of frames from the last tunnel found and the size of the ROI. Some not vision-based information would be very helpful to figure out if we are approaching a tunnel, for example GPS. In this case we estimate the new position and size of the tunnel entrance using motion estimation. Otherwise when it is none of the cases above, the tunnel entrance is not found. But when it was found in a previous frame, we obtain get the ROI from the last frame, which was saved in a buffer. This is reasonable because when the tunnel is not near, then the ROI does not change much when the vehicle moves forward. Otherwise, when the tunnel entrance was not found and our car is not located inside the tunnel, then it is figured out from our system, that there is no tunnel in front. For deciding whether the car is in the tunnel or outside of the tunnel, we calculate the mean value of the frame picture, and can decide whether it is bright or dark on the average. From this point of view our system works only for day time, but improvements are possible, for example using the algorithm of [4] for detection of changing driving environments. We have also implemented a first version of a tunnel exit recognition. After a car comes out of the tunnel, it searches for another tunnel entrance. So it works also for the case, that the car passes several tunnel entrances.

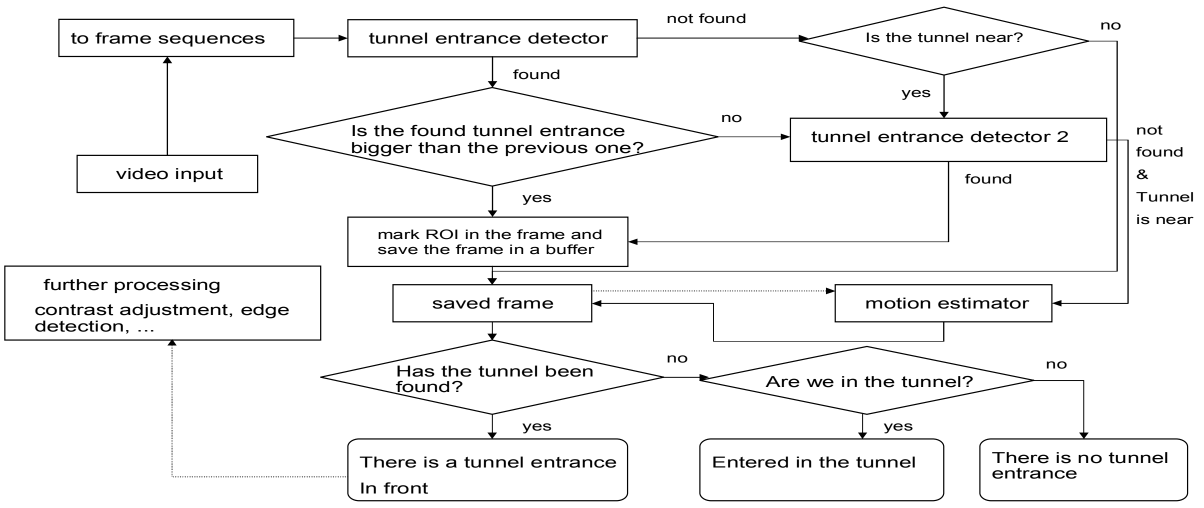
to frame sequences
tunnel entrance detector
not found
Is the tunnel near?
no
found
yes
video input
Is the found tunnel entrance bigger than the previous one?
no
tunnel entrance detector 2
not found & Tunnel is near
found
yes
mark ROI in the frame and save the frame in a buffer
further processing contrast adjustment, edge detection, ...
saved frame
motion estimator
Has the tunnel been found?
no
Are we in the tunnel?
no
yes
yes
There is a tunnel entrance In front
Entered in the tunnel
There is no tunnel entrance

Fig. 1: Outline of tunnel entrance recognition method

3. TUNNEL ENTRANCE RECOGNITION

Tunnel entrance recognition is done with the procedure described below.

3.1 Edge filtering

The first step to find the tunnel entrance is to filter the image with the Sobel Edge Filter [6] which was used because it is a common Filter to detect edges in pictures.

3.2 Structuring Lines

To reduce the processing time our system filters out all the lines which cannot be edges of a tunnel entrance, e. g. zigzagged or too short. The candidate lines should be curves or lines with certain minimum length, which have either only negative or positive grades. So almost only the lines which look like edges of a tunnel entrance or edges of lanes remain in the resulting image. We call this method 'Candidate Filtering'. All lines are undersampled twice, because we do not need to process all pixels. Fig. 2 shows an edge filtered image from one of our test videos. And Fig. 3 shows the undersampled and 'Candidate Filtered' image.

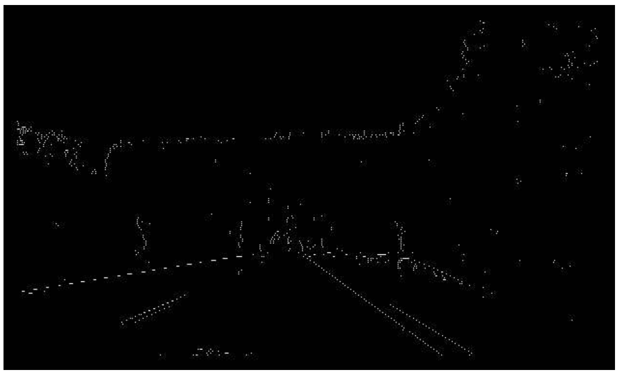

Fig.3: Candidate Filtered and undersampled image of Fig. 2

In Fig.2 there are 604449 pixels left over after edge filtering. After ½ undersampling 302220 pixels and after 'Candidate Filtering' 11335 pixels remain. Using these steps we can reduce the number of pixels to be processed by (1-11335/302220) *100% = 96.29 percent. After this processes all lines are arranged in a data structure, so that the system doesn't have to process the whole image again in further steps.

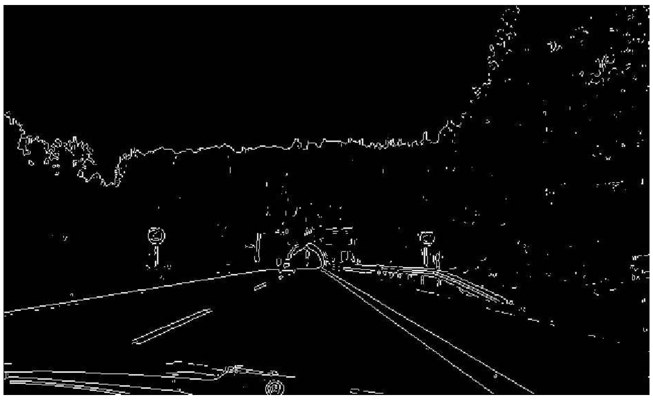

Fig. 2: Sobel edge filtered image

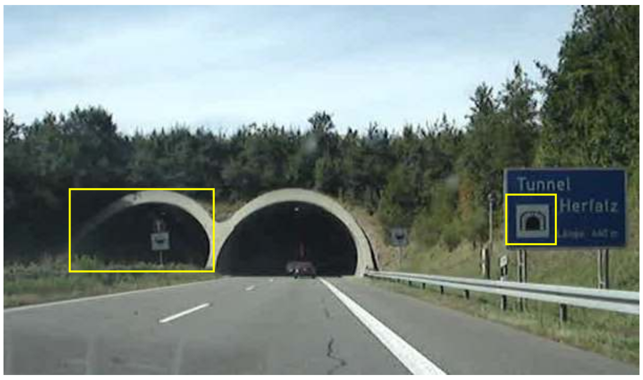

Fig. 4: Other objects which can be recognized as a tunnel entrance, which are none of our interests.

3.3 Lane Detection

Lane Detection is done to avoid the cases, that things different than the desired tunnel entrance are detected, like the cases shown in Fig. 4. Our Lane Detection uses the Radon Transformation [7] on the Candidate Filtered and undersampled image.

3.4 Tunnel Entrance Detector

As can be seen in Fig.1 the Tunnel Entrance Detector is the first image processing step in our algorithm. Between the candidate lines, we look for the lines which are arranged in pairs and where dark regions between these lines appear. When the found region is on the lane, which is detected in the previous step, it is considered to be the tunnel entrance.

3.5 Tunnel Entrance Detector 2

When there are vehicles in front of the tunnel entrance or the tunnel light interrupts the dark region, then it happened that only a part of the tunnel entrance was recognized (Fig. 5). In that case the detected ROI is smaller than the one found previously. If this happens we search for the tunnel entrance a second time. In this process, vertical lines that are surrounded by a dark region, represent tunnel lights or cars and have a distance between each other less than a certain threshold, are ignored. Then the tunnel entrance is defined as a dark region between the two remaining vertical lines.

3.6 Motion Estimation

If the tunnel entrance is near, which means that the ROI has a certain size, the boundaries between the dark and the bright region cannot be recognized, like shown in Fig. 6. In this case we estimate the position and size of the tunnel entrance from the velocity and the direction of the vehicle, which we get observing the circumstances of the tunnel entrance. We have implemented a motion estimator in an improved way for our considered case. Our system looks for obstacles in the ambiance of the ROI found before, and observes and compares, how much they have moved in the next frame. From this information our system calculates how the ROI has to be augmented and how much it should move.

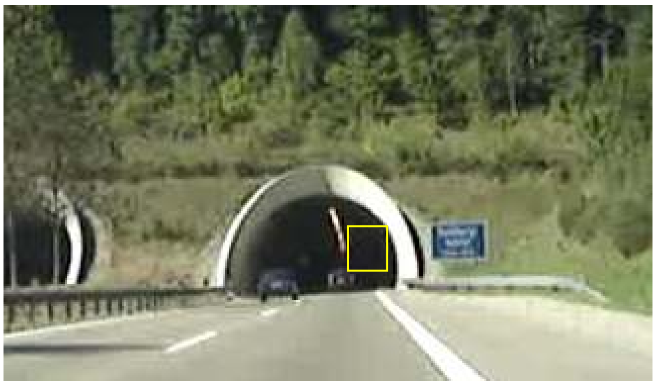

Fig. 5: A case where a part of the tunnel entrance can be found as a tunnel entrance

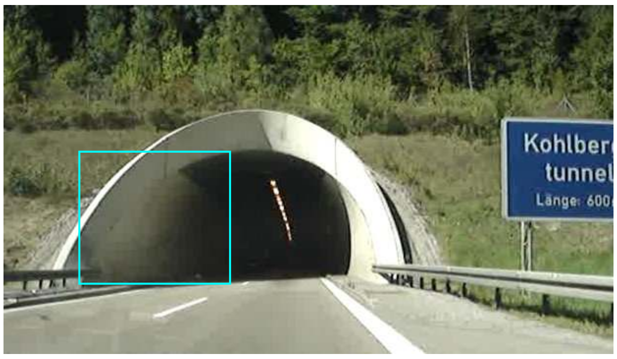

Fig. 6: The region where the boundary between dark and bright region cannot be recognized.

4. EXPERIMENTAL RESULTS

Our data consists of 2 videos which are taken by a video camera installed in a moving car on German highways. The videos are evaluated in the laboratory on MATLAB® environment. Fig. 7 shows the results of the tunnel entrance recognition of our system for each highway in 3 different distances from the vehicle. We can see that the tunnel entrances are recognized and the ROI is defined well corresponding to the size of the tunnel at each image. In whole video sequences nothing else than the tunnel entrance was recognized. Table 1 shows the CPU-time measured for each part of the system. Table 1 and table 2 show the times to process one image averaged over all images of our test videos. Table 2 shows the average real time for the three most time consuming parts. The times were measured under the following conditions:

Intel® Pentium® 4 CPU 3.00 GHz
2.99 GHz, 1,00 GB of RAM
MATLAB® Version 7.1.0.246 (R14) Service Pack 3
Operating System: Microsoft Windows XP Version 5.1 (Build 2600: Service Pack 2)
Java VM Version: Java 1.5.0 with Sun Microsystems Inc. Java HotSpot(TM) Client VM mixed mode

If we would examine a corresponding implementation in C and not an implementation in MATLAB®, we assume that the execution times are much faster.

5. CONCULUSION & FUTURE WORK

We have developed and implemented a vision system which recognizes and tracks a tunnel entrance from video sequences taken from moving car. To our knowledge, this is the first attempt to recognize and track a tunnel entrance from a moving car that is described in the literature. Our method is simple and it does not need the process of segmentation. Thus we can achieve less processing time which is essential for driver assistance systems. As the motion estimation part is the most time consuming one, we can think of realizing that part in hardware, which would be one extension of our work. Our future work in addition could be parallel object recognition system focused on driver assistance systems, which is done in [5] for general object recognition.

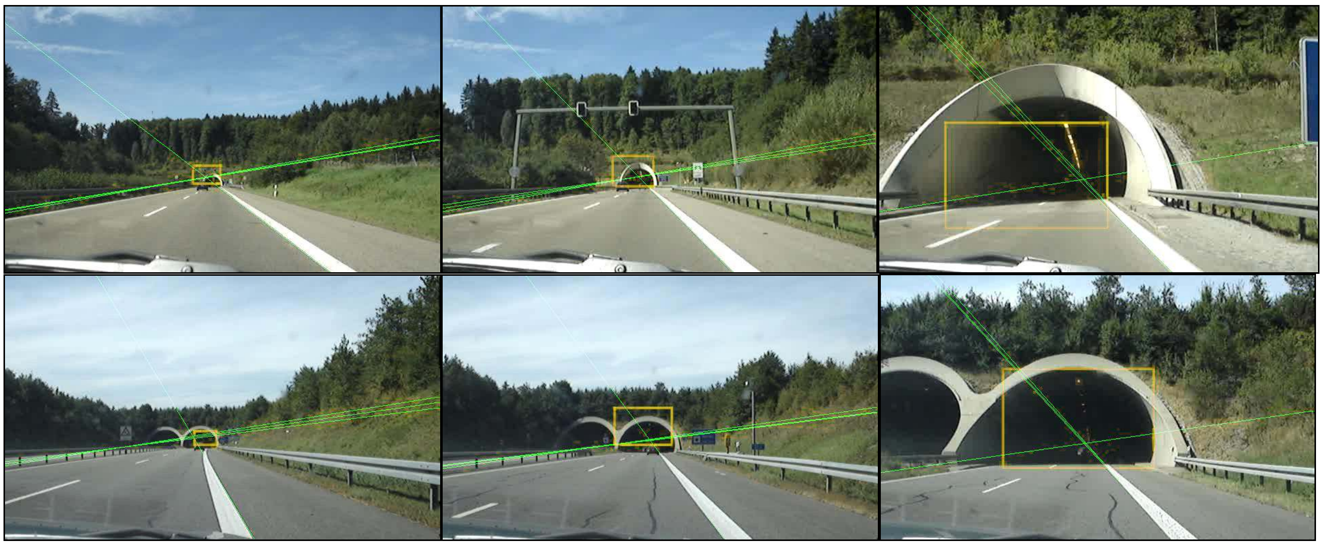

Fig. 7: Example of 2 image sequences where tunnel entrances are recognized and defined as ROI

Function Name	**Calls**	**Time**
Tunnel entrance detector (including lane detection)	809	1.568 s
Tunnel entrance detector 2	72	2.896 s
Edge Filtering	809	0.583 s
Structuring lines	809	0.451 s
Motion estimation	54	5.393 s

Table 1: Mean processing CPU time in Matlab for each part of the system, measured from 2 videos for each frame.

Function Name	**1st Video**	**2nd Video**
Tunnel entrance detector	1.1496 s	1.277 s
Tunnel entrance detector 2	2.4816 s	2.803 s
Motion estimation	4.7073 s	5.465 s

Table 2: Mean processing time in Matlab measured for three main parts of the system measured from 2 videos for each frame.

We are also considering of real-time implementation, HW/SW partitioning of our system with testing from more videos of driving conditions.

6. AKNOWLEDGEMENTS

We would like to thank COST 292, DFG SPP1148 for supporting our research, and BMW for the test videos.

REFERENCES

[1] Amnon Shashua, Yoram Gdalyahu and Gaby Hayun, “Pedestrian Detection for Driving Assistance Systems: Single-frame Classification and System Level Performance,” in Intelligent Vehicles Symposium, 2004 IEEE, pp. 1-6, June 2004

[2] Takeo Kato and Yoshiki Ninomiya, “Approach to Vehicle Recognition Using Supervised Learning,” in IEICE TRANS. INF. & SYST., VOL.E83-D, NO.7 JULY 2000.

[3] M. Betke, E. Haritaoglu and L. S. Davis, “Multiple vehicle detection and tracking in hard real-time,” in Proc.IEEE Intelligent Vehicles Symp., Tokyo, Japan, Sept. 1996, pp. 351-356

[4] Chiung-Yao Fang, Sei-Wang Chen and Chiou-Shann Fuh, “Automatic Change Detection of Driving Environments in a Vision-Based Driver Assistance System”, in IEEE TRANS. ON NEURAL NETWORKS, VOL. 14, No.3, MAY 2003

[5] Yongwha Chung, Seonil Choi and Viktor K. Prasanna, "Parallel Object Recognition on an FPGA-based Configurable Computing Platform," *camp*, p. 143, 1997 Computer Architectures for Machine Perception (CAMP '97), 1997.

[6] Rafael C. Gonzalez, Richard E. Woods and Steven. L. Eddins "Digital Image processing using Matlab", p. 393-404, Pearson Prentice Hall, 2004

[7] Sigurdur Helgason "The Radon transform”, Progress in Mathematics, 5. Birkhäuser, Boston, 1980