Vehicle Detection, Classification and Position Estimation based on Monocular Video Data during Night-time

Jonas Firl, Marko H. Hoerter, Martin Lauer and Christoph Stiller

Keywords: *Automotive Lighting, Light-based Driver Assistance, Detection, Tracking, Classification, Light Sources.*

1 Abstract

This study describes an effective method for detecting, tracking and classifying vehicles during night-time in order to support automotive adaptive illumination applications. The hereby described software framework, which computes the relative position, velocity and estimated class of all detected vehicles, integrates multiple processing stages. Firstly, an image segmentation using a threshold method to detect all light sources in the image. Secondly, possible pairs of head- and taillight are clustered using geometrical information. Thirdly, all detected objects are tracked using a Kalman-Filter to increase resolution and robustness of the algorithm. Lastly, a method for computing distance and velocity for all classified objects (e.g. cars, trucks, bikes…) is presented. The system is tested to run in real-time and some results and conclusions are offered at the end.

2 Introduction

Driving a vehicle during night-time is relatively more hazardous than at day-time, which was recently reviewed in detail by the German Federal Office of Statistics (DESTATIS, [1]). Technically inadequate illumination of the present road scenery as well as wrongly utilized light function (e.g. pure usage/ alternation between low- und high-beam) are just some examples on that.

In order to run lighting based driving assistance system acting upon the maxim “best illumination without dazzling other traffic attendance”, it is mostly important to estimate the spatial position, velocity as well as the classified category of the detected light source. By having such data reliably acquired from the sensor side, different lighting based driving assistance system can be realized, such as *Adaptive Cut-Off-Line*, *Predictive Illumination Distance Control* or *Masked High Beam* derivates [2].

2.1 Objective of research project

The intention of the hereby described research project is to detect, track and finally classify light-sources based on real-world monocular image sequences.
To implement a most suitable and efficient detection algorithm, different detection methods have been evaluated to each other with respect to their performance. For tracking purposes a linear and discretized Kalman-Filter has been utilized, which represents an iterative and stochastic state estimator by minimizing the middle error square. The applied classifier calculates the probability of error concerning to the found light source based on real-world image sequences.

2.2 Utilized Hard- and Software

The above mentioned image sequences were acquired by utilizing the dedicated testing vehicle at the Department of Measurement and Control (see Figure 1 and Figure 2). The hereby used visual sensor (Point Grey, type “Firefly MV”) has been mounted on a camera platform close to the rear-view mirror and was driven at 30 frames per second. To guarantee timely synchronous data (e.g. vehicle immanent plus video data), a real-time data base (RTDB) framework, introduced in [4], has been exploited.

Figure 1 - AUDI Q7 as a testing vehicle at the Department of Measurement and Control. [3]

Figure 2 - Dedicated camera platform mounted behind front window. [3]

3 Light source detection

Within this section the used light sources detection algorithm will be described in detail. As a requirement, the detection algorithm should be capable not only to detect vehicle front lights (e.g. cars, trucks, motorcycles), but also other light

sources with sufficient light intensity, like street lamps. In [3] different detection methods, like SURF, HOUGH-transformation or dedicated (multi-level) threshold operation with respect to performance and applicability have been evaluated. As a result of this evaluation finally a picture-row-based threshold operation, described in [5] has been utilized in the further proceeding (see Figure 3).

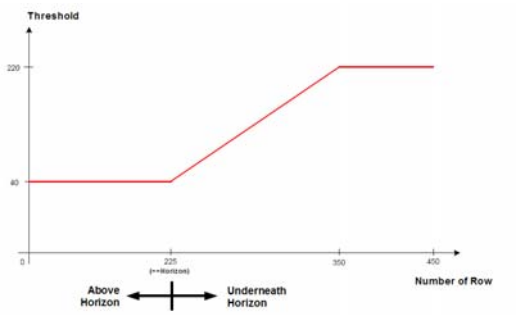

Figure 3 - Found parameter curve of the picture-row-based threshold method; here applied on a 640x480 pixel picture. [3]

Figure 4 - Outcome of the picture-row based threshold filtering method. [3]

3.1 Light source detection with threshold filter

As described above, a picture-row-based threshold operation will be utilized to set the threshold value corresponding to the vertical pixel position. Like pointed out in [5], one of the most important benefits of this method is to detect even low-level light emitting sources (e.g. tail lights) in far distances in comparison to methods with a constant threshold. Based on the threshold filtered pictures connected blobs, which are basically two or more detected light sources, can be extracted. Therefore, the so called *connected component analysis*, more specifically detailed in [6], has been used in this work. Hereby one calculated blob integrates values as follows: midpoint, standard deviation and number of pixel. Due to their occurrence, detected light sources above the horizon are completely described by these values (see Figure 4), whereas the blobs underneath the horizon still need further processing to extract single light sources out of it.

3.2 Detection of reflections

If single light sources do have sufficient distance between each other, the mentioned detection will be done by the threshold filter operation itself. If not, two or more light sources appear as a single light source. This happens most likely within overexposed picture regions, for instance caused by reflection or dense traffic situations. The challenge on that is to evaluate the detected blobs of the given the picture, whether they consist of one or multiple light sources (e.g. two head lights plus two corresponding reflections, see Figure 5).

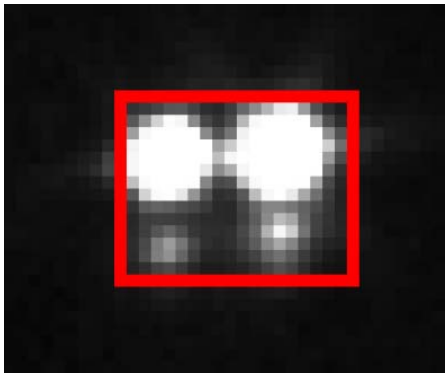

Figure 5 - Consolidated blobs resulting from threshold filtering. [3]

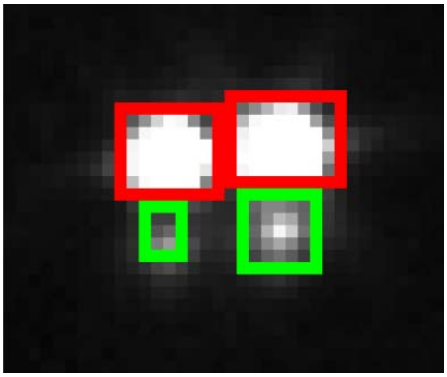

Figure 7 - Separated blobs (red) plus corresponding reflections (green). [3]

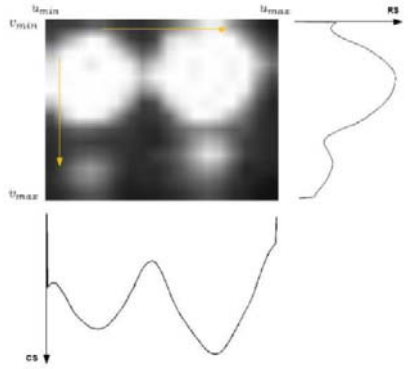

Figure 6 - Pair of head lights with corresponding reflections. [3]

Figure 8 - Pairs of light sources grouped together by clustering algorithm. [3]

Within this work, a proper separation between single blobs and their reflections have been accomplished by summing up the pixel values column by column as well as row by row. A blob detected at its position $[u_{min}, u_{max}] \times [v_{min}, v_{max}]$ with-

in a picture I results with its corresponding row sum vector $RS = [RS_{v_{min}}, \dots, RS_{v_{max}}]$ as well as its column sum vector $CS = [CS_{u_{min}}, \dots, CS_{u_{max}}]$:

$$RS_{v_i} = \sum_{u=u_{min}}^{u_{max}} I(u, v_i) \qquad v_i = v_{min}, \dots, v_{max}$$

$$CS_{u_i} = \sum_{v=v_{min}}^{v_{max}} I(u_i, v) \qquad u_i = u_{min}, \dots, u_{max}.$$

As next step those stated vectors can be inspected for local maxima to get the number of separate light sources out of it. By illustrating Figure 6, we can detect two local maxima for each pixel direction, whereas at their intersection points the four corresponding light sources can be found. This makes it easy to distinguish between light sources and reflections by simple heuristic knowledge (reflection are most usual underneath its corresponding light sources) by considering a maximum gap between those two groups (see Figure 7).

3.3 To cluster pairs of front- and tail lights

As a result of the detection stage, a list of detected light sources $L = [L_1, \dots, L_n]$ per picture can be extracted. To be able to track obstacles within the traffic context, a clustering method has to pool potential pairs of front- and tail lights together. In [7] a method is presented, which clusters light sources within a given picture based on geometrical properties. In our approach, this method has been adapted to the effect that the input data of two light sources L_i and L_j with geometric dimension is given as follows $(k = i, j)$: The maximum and minimum pixel position $[u_{min}, u_{max}] \times [v_{min}, v_{max}]$, the correlated height and width $H_k = v_{k_{max}} - v_{k_{min}}$, $W_k = u_{k_{max}} - u_{k_{min}}$ as well as the number of pixel corresponding to the light source A_k. The two given light sources will be pooled together as soon as the following four properties are positively evaluated:

(1) **Vertical projections have adequate intersection areas:**

$$\left| v_{i_{\min}} - v_{j_{\min}} \right| < c_{vp} \max(H_i, H_j),$$

$$\left| v_{i_{\max}} - v_{j_{\max}} \right| < c_{vp} \max(H_i, H_j).$$

(2) **Height approximately at the same range:**

$$\left|H_i - H_j\right| < c_j \max(H_i, H_j) .$$

(3) **Number of pixel approximately at the same range:**

$$\frac{\min(A_i, A_j)}{\max(A_i, A_j)} > c_a .$$

(4) **Horizontal distance as follows:**

$$D_h = \frac{\left|(u_{i_{\max}} - u_{i\min}) - (u_{j_{\max}} - u_{j\min})\right|}{2} < c_d \max(H_i, H_j, W_i, W_j) .$$

The parameters c_{vp}, c_h, c_a, c_{hd} have been justified manually through evaluating a huge amount of test data. As two major differences in comparison to [7], the calculation of the horizontal distance D_h of a dedicated light source has been adapted in two ways: Firstly, by not only taking the maximum height under consideration, but also the maximum width. This has been done due to the fact that light sources appear in reality not always in ideal round shape. Secondly, the threshold c_{hd} has been increased to a higher level to be able to detect also very tiny appearing tail lights. As a result on that, a list of potential pairs of front- and tail lights can be stated as well as applied to the given sequence of pictures (see Figure 8).

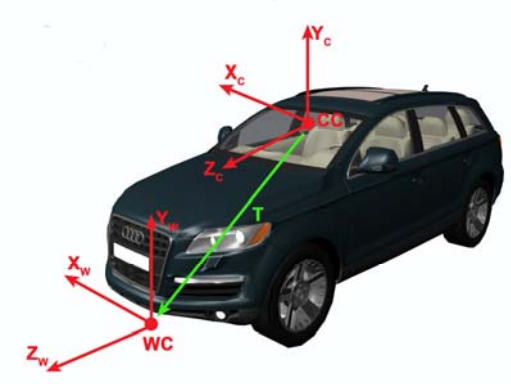

Figure 9 - Camera and world coordinate system. [3]

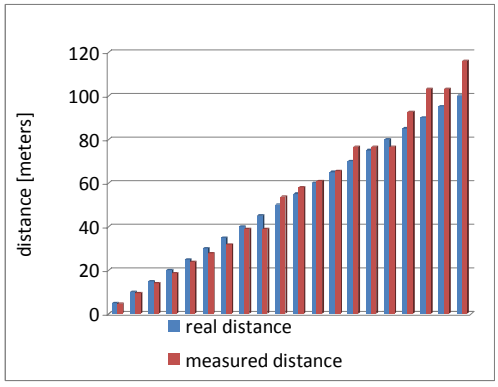

Figure 10 – Results of the position estimation of an object in different distances. [3]

4 Position- and velocity estimation

The aim of this section is to present a method to compute 3D positions in a fixed world coordinate system by given 2D image coordinates of detected light sources. These data are essential to get parameters like distance or velocity of

detected vehicles in the road scenery in front of the own car. At the same time they will be the input data of the tracking algorithm described in section 5.
In most cases the problem of the under-determined transformation system is solved by using a stereo vision approach. Due to the hardware setup of this project - usage of only one camera in order to keep costs and packaging down - and to general problems using stereo vision systems in night time scenarios a restriction of the world has to be done: we assume a flat (2D) world. Hence the horizon can be assumed to be a constant horizontally line in the image.
With the world coordinate system given in Figure 9, all objects have to be coplanar parallel to the (X_W, Z_W)-plane, and therefore have a constant "height" d. This plane can be described by using e.g. the hessian normal form

$$\left\langle (X_W, Y_W, Z_W)^T, (0,1,0)^T \right\rangle - d = 0 .$$

With the known parameters from the camera calibration (i.e.: rotation matrix R, translation vector T, camera matrix A) we can describe the transformation from a given image point (u,v):

$$(X_W, Y_W, Z_W)^T = \lambda R^T A^{-1} (u,v,1)^T - R^T T ,$$

with:

$$\lambda = \frac{d + \left\langle R^T T, (0,1,0)^T \right\rangle}{\left\langle R^T A^{-1} (u,v,1)^T, (0,1,0)^T \right\rangle}$$

In *Figure 10* the results of the transformation are shown (distance $d = \sqrt{X_W{}^2 + Z_W{}^2}$). We can easily see the quadratic error increase due to small pixel errors, which normally occur with increasing object distances (see also [8]).
After the positions are estimated, the next step is to compute the velocity relativ to the own car. With avaible CAN-data (velocity and steering angle) it is easy to compute the vectoriell velocity using given positions $\vec{p}_t = (X_W, Z_W)_t$ at time t, e.g. using the formula presented in [9]: $v_t = \frac{\sum_{k=1}^{n} p_{t-k} - \sum_{k=m+1}^{m+n} p_{t-k}}{mn}$.

5 Clustering and Kalman-tracking

In this section an algorithm to track possible vehicle-like objects in a sequence of images will be presented. Prior to the tracking it is required to extract possible vehicles like cars or bikes out of the blob list and generate a list of all objects in

an image. Therefore we have to recognise, that vehicles can appear in the image as objects consisting of one (e.g.: bikes, far away cars) or of two blobs. Another important fact is that, with the flat world assumption of section 4 and the therefore constant skyline, vehicles can only appear under the constant image row of the horizon.

One-blob objects can be detected using information about there size and location in the image. Two-blob objects have to be clustered using the technique described in section 3.3.

After creating a complete list of objects in the image, a tracking algorithm can be applied, which will recover all vehicles over a sequence of images as well as compute their relative physical values. Therefore objects of the last image have to be associated with the current image data, taking account of the information gained during the past recovering process. One of the difficulties of this process is the fact that one object doesn't have to consist of the same number of blobs over a set of frames.

After this data association a linear and discretized Kalman-Filter is used to increase robustness of the results and to improve spatial resolution of the distance estimation. The state vector x and the measure vector y are defined as follows, considering the flat world assumption:

$$x = (_W, Z_W, v_x, v_Z)^T \qquad \text{and} \qquad y = (Z, X)^T,$$

with the velocities in X_W- and Z_W-direction v_X and v_Z. Assuming the model of approximately constant velocity (between two frames), the system matrix is given by:

$$A = \begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

with the camera frequency $1/T$. The error, which occurs using this modell, is handled with the error matrix Q, which can be computed by using the modell of piecewise constant accelaration, simplified resulting in:

$$Q = \begin{pmatrix} \frac{T^4}{4}a_Z{}^2 & 0 & \frac{T^3}{2}a_Z{}^2 & 0 \\ 0 & \frac{T^4}{4}a_X{}^2 & 0 & \frac{T^3}{2}a_X{}^2 \\ \frac{T^3}{2}a_Z{}^2 & 0 & T^2 a_Z{}^2 & 0 \\ 0 & \frac{T^3}{2}a_X{}^2 & 0 & T^2 a_X{}^2 \end{pmatrix}.$$

Here, a_Z denotes the maximal velocity in Z_W-direction, a_X respectively.

In *Figure 11* the results of the Kalman-Filter are shown. The estimated and Kalman-filtered distances are plotted over some frames from a scenario of one departing vehicle to point out the advantages of this tracking algorithm.

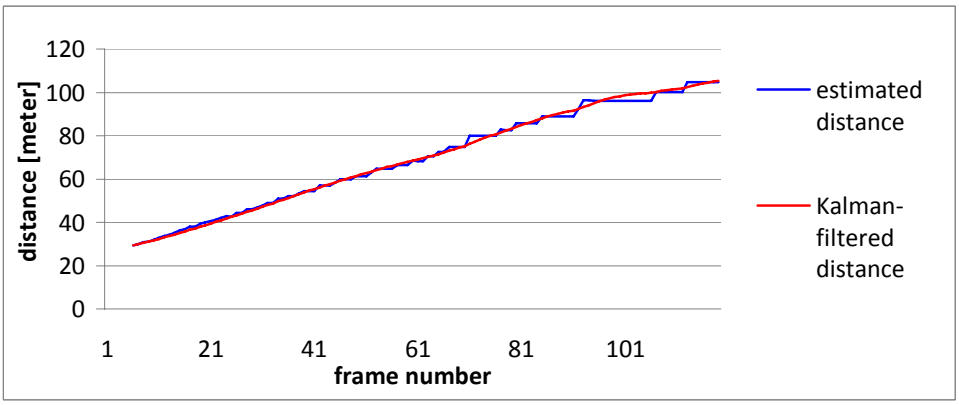

Figure 11 - Results of the Kalman-Filter. [3]

6 Object classification

The final stage of this project is to execute a classification process to all detected and tracked objects. We sort the detected objects into the following classes: *Bike*, *Car*, *Truck* and *Unknown*.

Firstly a feature-vector $\vec{m}_i$ is defined for every object $O_i \in O$, which has to be computed befor starting the classification:

$$\vec{m}_i = (n_{blobs}, lt, d, sq, v, a)^T,$$

with:

- n_{blobs}: Number of blobs the object consists of [-]
- lt: Lifetime of the object (number of tracked images) [-]
- d: Distance of the object [meters]
- sq: Squareness of the object (width[pixel] / height[pixel])

- v: Velocity of the object [meters/sec²]
- a: Area of the object [pixel²]

After interpreting a huge amount of training data, it is possible to formulate a probability function $p(c \mid m)$ for every class c and for every element m of the feature-vector $\vec{m}_i$. The resulting probabilities are weighted and summed up to get the complete probability $p(c \mid \vec{m}_i)$ for every class and allow using a simple maximum a posteriori classifier. Because of the difficulty in distinguishing between lights of a truck and those of a passenger car, the process shown in Figure 12 has delivered good classification results, which executes a detection of clearance lights for all possible trucks as a distinctive means.
As a result we obtain the corresponding class for each object (class *Unknown* is an indicator for an uncertain classification or any kind of noise) including its probability.

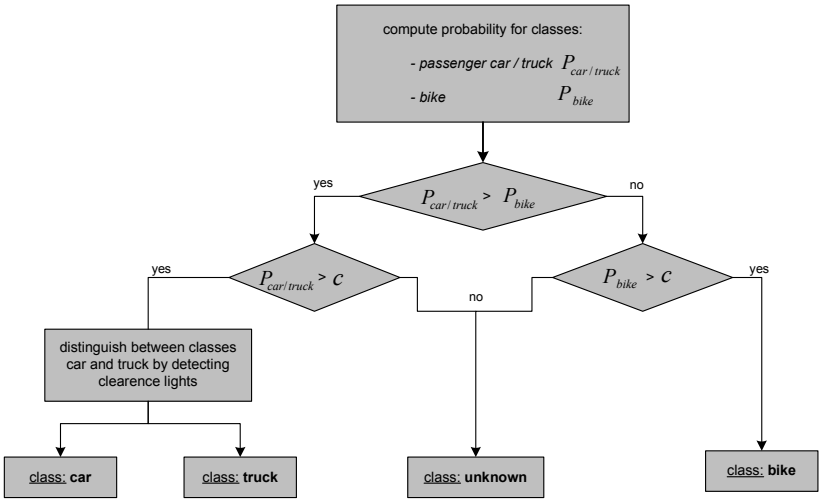

Figure 12 – Classification process, the constant threshold c is set experimentally. [3]

7 Summary and Outlook

With this paper we have presented a novel software framework to detect, track and classify light sources within a traffic context. By evaluating the computation time of the different software modules, it can be seen, that the over-all algorithm

is capable to run under real-time conditions due to its highly efficient implementation [3].

In the future, light-based driving assistance systems will be highly supported by advanced sensor systems, such as video, radar, laser, or lidar units to estimated physical values of the driver`s environment to guarantee an optimum of illumination while driving at night-time. The presented approach in this paper shows first promising results, nevertheless further enhancement in the field of detection range and reliability have to be done in the future.

8 References

[1] Statistisches Bundesamt Deutschland, *„Entwicklung der Zahl der im Straßenverkehr Getöteten 1953 bis 2008“*, Section „Verkehr/ Verkehrsunfälle“, Germany: Wiesbaden, 2009

[2] Marko H. Hoerter et. al, *„A Hardware and Software Framework for Automotive Intelligent Lighting“*, Proceedings of IEEE Intelligent Vehicles Symposium 2009, China: Xi`an, 2009

[3] Jonas Firl, *„Lichtquellendetektion, -klassifikation und Positionsbestimmung in nächtlichen monokularen Bildsequenzen“*, University of Karlsruhe(TH), Department of Measurement and Control, Germany: Karlsruhe, 2009

[4] Matthias Goebl et al., *“A Real-Time-capable Hard- and Software Architecture for Joint Image and Knowledge Processing in Cognitive Automobiles”*, In Proc. IEEE Intelligent Vehicles Symposium, pages 734-740, Turkey: Istanbul, 2007

[5] M.Y. Chern et. al, “*The lane recognition and vehicle detection at night for a camera-assisted car on highway*”, In Proc. of IEEE International Conference on Robotics and Automation 2003, volume 2, 2003.

[6] F. Chang et al., “A linear-time component-labeling algorithm using contour tracing technique”, Computer Vision and Image Understanding, 93(2):206–220, 2004.

[7] Yen-Lin Chen et al., „*Nighttime vehicle detection for driver assistance and autonomous vehicles*”, Proceedings of the 18th International Conference on Pattern Recognition, pages 687–690, USA: Washington, DC, 2006.

[8] Gideon P. Stein et. al, „Vision-based ACC with a Single Camera: Bounds on Range and Range Rate Accuracy“, Proceedings of the IEEE Intelligent Vehicles Symposium 2003, June 2003

[9] Kaiqi Huang et. al, „A real-time object detecting and tracking system for outdoor night surveillance“, *Pattern Recognition*, 41(1):432 – 444, 2008

[10] Yen-Lin Chen et. al, „Night-time Vehicle Detection for Driver Assistance and Autonomous Vehicles“, In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 687–690, Washington, DC, USA, 2006. IEEE Computer Society.