Driver assistance based on lane detection and road markings

Bogdan-Mihai Podaru
Technical University of Cluj-Napoca
Oradea, Romania
podaru.bogdan98@gmail.com

***Abstract*—As the automotive industry evolves in the direction of fabricating completely autonomous vehicles, techniques like Adaptive Cruise Control and Lane Departure Warning Systems are taken into consideration for further improvement. Producers' main concern is the reaction of such systems in general frames with unseen situations. Despite the solutions that the engineers choose to implement for autonomous guidance systems, the problem of lane detection and road markings classification consists one of the most important parts.**

This paper addresses the problem of driver assistance by extracting information about the road. The article presents a solution for the driver to stay informed during the travel, about the lane borders and road markings such as the lane change arrow markings.

Keywords—Driving assistance, Autonomous vehicle guidance, Image segmentation, Particle filter, Scanline

I. Introduction

Driving assistance gives knowledge about the road the driver is traveling on to enhance driving control and reduce chances of car accidents. Referring to a Lane Departure Warning System (LDWS) a general driving assistance system incorporates the logic of detecting and analyzing the geometry of the road and its markings to produce support for decisions such as the posibility of a lane change, an overtake or a turn-off.

The main contribution of this paper is the analysis of the road markings resulting in different messages informing the driver about the actions that can be made. Messages can be divided into informative and warnings. If the driver tries to overtake the car in front of him by crossing the solid line the system will deliver a warning message. This paper addresses the the warning type of messages. The possibility of a lange change in a direction is signaled when conditions like dashed line and arrows referring to that direction are met. In order for the system to deliver the mentiond type of messages the lanes need to be analyzed and their associated markings, so the main objective of this article is the detection of lanes and their markings. The secondary objective is the warning system described above. [1] - Road markings can be devided into markings of lane borders and painted arrows on the road.

In the field of lane detection a lot of related work and approaches exist. The common approach is to isolate the white lines and find structures like segments for further combination. Techniques like Lucas-Kanade, Hough Transforms, Canny edge detectors, Region of interest (ROI) determination, fog elimination, Bilateral features, K-means clustering and many more are well-known in this domain. This article is inspired from the approach in [1] where a particle filter is used based on cues representing hints about the observed scene, but instead of cues the lane detection will conform to isolating the white lanes with other approach described in section *II*. The idea of cues is presented in many computer vision articles. In [2] the backbone of the paper presents the Distillation algorithm that extracts reusable cues from the image and combines them in a fusion process to produce a robust solution for lane detection and tracking. Another cue-based approach can be found in [4] which makes use of three image cues: Canny edge filter-Hough transform Cue, Laplacian of Gaussian Edge Cue and Colour cue in order to make assumptions about the position of the vehicle .

Many articles use different lane models but this particular one assumes that the road is straight and flat. So does the article [3] where a robust feature extractor is proposed based only on the geometry of the lane-markings. For a more complex road model (i.e geomtry, radiometry, topology and context) [5] describes a model for the extraction of roads from aerial images for geometric information systems (GIS).

This paper is organized as follows. First the approach of lane detection described in [1] is detailed and modified according to the objectives in II. The analysis is than switched over three main parts. In part 1 the estimation of the type of lane border markings is presented in section III-A, the classification of painted arrows in section III-B and the Driver assistance system in section IV.

II. Lane detection

Lanes are the regions of the road delimited by two lines according to some regulations. The lane model for estimating each lane assumes that the lane is straight and flat and it is described by 4 parameters (as depicted in [1]) :

Lane model $\mathbf{M} = \{x_0, \psi, w, \varphi\}$, where : x_0 represents the lateral shift of the car's longitudinal axis with respect to the middle of the lane, ψ sets the rotation between the car's main axis and one paralel line to the lane, w is the width of the lane and φ is the tilt angle (the angle obtained by the looking

direction of the agent and the plane of the road). The agent can be the drivers' eyes or a mounted camera on the inside rear view mirror. For simplicity this paper makes an abstraction and considers as existing such a camera but never makes use of it.

As we can have more than one lane each lane will be indexed with a number:

$$\mathbf{M}^{\mathbf{i}} = \{x_0^{\mathrm{i}},\ \psi^i,\ w^i,\ \varphi^i\}$$

For simplicity the case of *i=1* (the lane in front of the car) is taken for analysis and further development could consider this as basis (*i>1*).

The process begins by acquring a sample of the road. The sample is passed to a rule-based systems in order to keep track of the all lanes identified, therefore a set of rules is used to start new trackers and to terminate outdated ones[1]. At this point each lane uses a single lane tracker. For tracking, a particle filter can be used as described in [1]. However this article follows other steps for determining the lane the car is traveling on :

1. Isolate white lines on the image : turn RGB into HSV (in HSV, Hue component will treat the white lines as one color, while in RGB different parts of the lane lines may be lit with different lights). Then lift out all the white colors from the image. This is done by specifying a range of the color .
The result from this step is a white mask.

2. Detect edges of a lane line (int the white mask) with Canny edge detection method resulting in all edges of all white areas.

Canny edge detector is an edge detection operator that uses a multi-stage algortihm to detect a wide range of edges in images. It consists of 5 steps :

2.1) *Noise reduction* by applying Gaussian blur to smooth it. This is done by convolving with a Gaussian Kernel of size **(2k+1) x (2k+1)** :

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right)$$

2.2) *Gradient calculation*: this step detect the edge intensity and direction. When the image is smoothed, the derevatives ***Ix*** and ***Iy*** can be implemented by convolving the image ***I*** with Sobel kernels ***Kx*** and ***Ky*** *:*

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Then the magnitude ***G*** and slope $\boldsymbol{\theta}$ are computed as following :

$$|G| = \sqrt{Ix^2 + I_y}^2$$

$$\theta\,(x_1, y) = \arctan\left(\frac{Iy}{I_x}\right)$$

2.3) *Non-maximum supression* : thins out the edges.
2.4) *Double threshold* find strong, weak and non-relevant pixels.

High threshold is used to identify the strong pixels and a low threshold is used to identify non-relevant pixels and everything in between the thresholds are flagged as weak pixels. The aim is to obtain two pixel intensities corresponding to *weak* and *strong*.
2.5) *Edge tracking by hysteresis :* trasnform weak pixels into strong ones.

3. *Isolate region of interest (ROI):* detect lane lines that are closer to the car (bottom of the screen) and crop the top half. This way the outliers are cropped out. The ROI has a trapezoid shape that can be adjusted according to the road by the following parameters : trapezoid height, high base width and low base width, and two offsets relative to the top and bottom margins of the screen.

The results can be seen in:

4. *Detect line segments*. At this point we have four lines (two lane lines) represented as a group of white pixels. Apply Hough Transform to extract features like lines (fit many straight lines through a set of pixels that seem to form a line and return the set of lines that is subject to mininum threshold constraints).

Hough transformis a feature extraction method for detecting simple shapes such as circles or lines in an image. The line is represented in polar coordinates because the parameters need to be bounded and the slope-intercept form of the equation can lead to the cases of infinity in terms of the slope.

$$\rho = x\cos\theta + y\sin\theta \qquad \textbf{(5)}$$

where $\boldsymbol{\rho}$ represents the perpendicular distance of the line to the origin in pixels, and $\boldsymbol{\theta}$ is measured in radians.
The following steps are performed to detect lines in an image :

Step 1: Initialize accumulator

An accumulator is similar to cartesian space with $\boldsymbol{\theta}$ as horizontal axis and $\boldsymbol{\rho}$ as vertical axis. This can be seen as a 2D array where 1 squared unit is named a bin which is used to collect evidence about lines existing in the image.

Expermiental results lead to the choice of a (AFTER IMPLEMENTATION ex:10x10) accumulator. This means that ρ takes up 10 distinct values and θ take up 10 distinct values resulting in 100 cells in the accumulator. This can be interpreted as the ability of the algorithm to detect 100 different kinds of lines.

Step 2: Detect edges

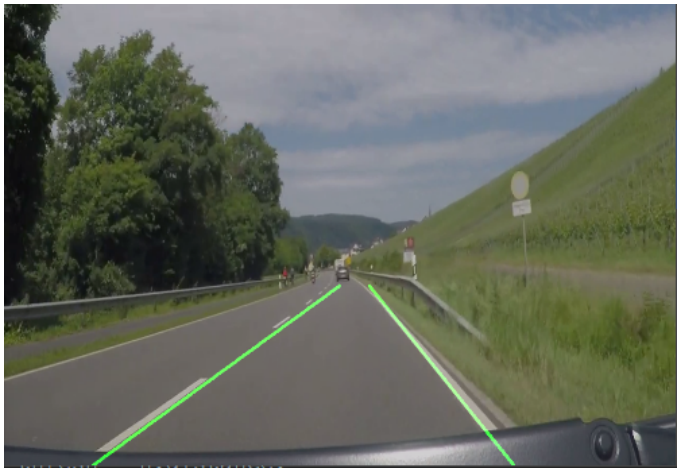
Every cell of the accumulator corresponds to one line. If there is a visible line in the image, the edge detector fires at the boundaries of that line and the edge pixels provide evidence for the presence of a line. The result of this step is an array of pairs denoting the edge pixels.

Step 3: Voting by Edge Pixels

For every edge pixel *(x, y)* in the above array the values of θ are varied from 0 to π and plut it in equation 5 to obtain value for ρ. The accumulator is used to find the intersection of all the curves generated by the edge pixels. Then the bins above a certain threshold (VALUE) is used to find all the lines in the image.

5. Combine line segments into two lane lines obtaining left and right lane lines. From step 4 we have many small line segments with their endpoints (x_1, y_1) and (x_2, y_2). We need to classify the segments by their slopes: all line segments belonging to the left lane line should be upward sloping and all segments belonging to the right lane line should be downward sloping. Once the line segments are classified into 2 groups the average of the slopes and the average of the intercepts of the line segments are computed to get the slopes and intercepts of left and right lane lines.
However there is a special case worth discussion. There are cases where segments have a slope of infinity. Because their occurance is rare the average does not take them into consideration.The system will display the last computed lane lines giving the illusion of continuous displaying; this is performed to address the problem of dashed lines where slopes for regions with inexistent lines cannot be calculated.

The results after performing all of these chapter steps:

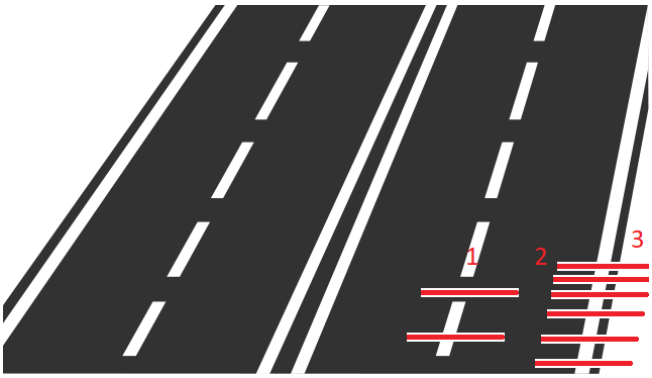


III. Road markings

Road markings analysis is focused on solid/dashed lines and road arrows. The type of arrows is examined in this section in order to determine the type of the lane and the legal actions to consider. The whole sample is restricted to 2 regions of interest (ROI): lane borders ROI and middle of lane ROI (for arrows) which are the expected regions of such markings.

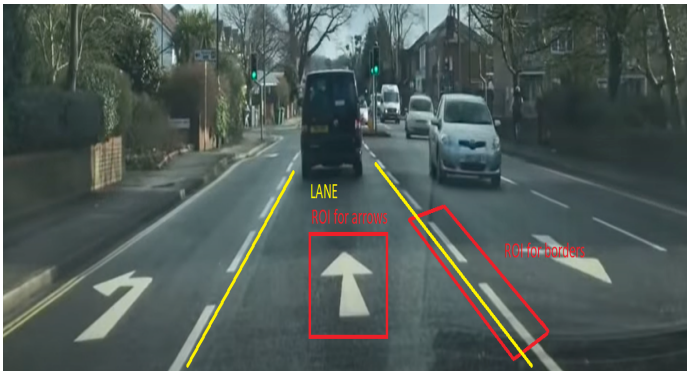
A. Classification of lines

The pipeline of road lines analysis is the following:
Lines are sampled using scanlines. Each scanlines is classified in order to find the type of line it represents Then the outliers are removed. The final step is to analyze all scanlines and determine if there is a solid or dashed border line underneath the scanline. As a drawback we cannot use global threshold because the road surface and markings have different brightness at different distances so a binarization along the scanline is performed resulting in three regions. Intensities along projected scanline is mainly influenced by the three road regions mentioned above: road marking, road surface and roadside. These three regions corresponds to three peaks in distribution of intensities which can be estimated with k-means clustering (as in [1]). After the segmentation all pixels of the lane markings are set to 1 whereas the rest is set to 0. However isolating the white lines from the conversion to HSV space solves this problem so we can take the processed sample from that step and make further computations on that sample in order to cast scanlines but slows down the whole processing because the classification of lane lines and the detection of the lane are now sequentially linked.

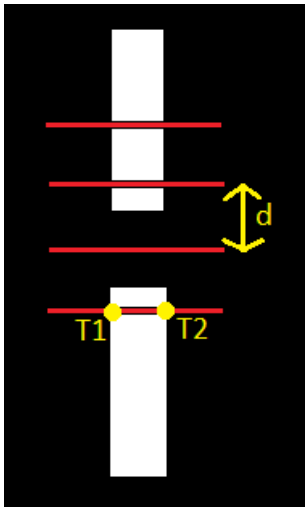


1- road markings
2-road surface
3-road side

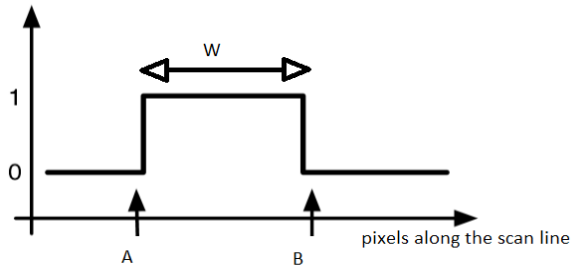
As mentioned above the regions where a road marking is expected define our ROI. For the arrow markings analysis, this kind of symbol is expected to be present in the middle of the lane whereas the lane borders whether they are dashed or solid are expected to be found at the boundaries of the lane. Step 5 from section II presented the way of extracting these two boundaries (left and right lane lines). So two types or regions of interest are to be analyzed :



This section focuses on the analysis of the ROI for borders as the ROI for arrows is detailed in the next section.
For this analysis two approaches are to be considered relative to the type of sample being analyzed. At the aquisition of the road sample two decisions can be made: split the pipeline into two independent pipelines, one processing the lane boundaries and the other processing the type of lines (**A**) or consider a sequential processing consisting of the extraction of road markings followed by the classification of the border lines(**B**).
A) After extracting the region of interest scan lines are casted on the image to determine the type of road markings. The following picture depicts the extracted region from the above image after segmentation and casts 4 scan lines on it. The parameter d stands for the distance between scanlines which is experimentally adjusted and T1,T2 represent the particular threshold for that scanline.



The algorithm makes further computations to obtain an array of intersections between the scanlines and road marking, for example after processing the above image will result in an array of the form : **{ s1 : 1, s2 : 0, s3 : 1, s4 : 1}** where 0 stands for road surface, 1 for road marking and 2 is an extension for further improvements of the project where road side will be taken into consideration. Allowing a third type of region could solve problems like dust, shadows, grass, et al.
As depicted there is a change in image intensity along the scanline resulting in 2 specific threshold T1 and T2, but after segmentation 2 regions are highlighted : the road marking region and others .

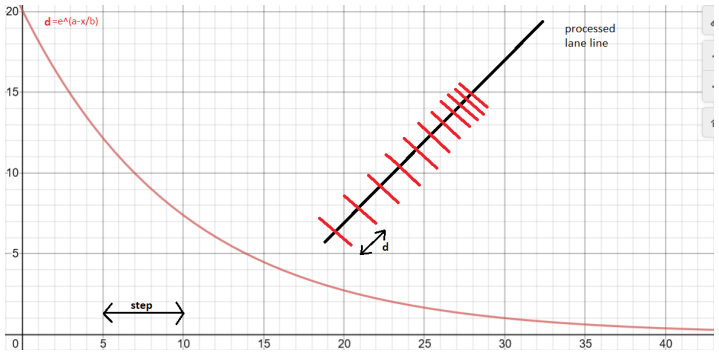


The continuous stream of region identifiers (0,1,2) obtained with every sample is analyzed and the group width is determined in order to classify the markings. For example with a group width of 10, ten intersection of the scanlines with the markings will be analyzed in order to obtain the type of marking. At this point the offset of the group (starting position for analysis) is experimentally adjusted.

(**B**) However the approach of using two thresholds is expensive in terms of parallelism. Better results were obtained by using the same sample from the conversion to HSV. From step 5 of section II the final lane lines are to be expressed by slope and y-intercept. These parameteres are to be comined with the approach in (**A**) where we could sample directly the lane lines with a step of length d.
This approach can be adjusted by using one of the functions from the Exponential Decay class. For this particular project the following function presents the best results :

$$e^{(a-x/b)}$$

This function descrease over time which suits the situation of dashed segments getting close to each other as the distance from the car increases. The parameter d mentioned above is the ssampling step along the lane line, called the distance between the scanlines. As the exponential decay function decrease over time its value is subtracted from the distance between scanlines. The paramater **a** describes the starting value to be subtracted from **d** and **b** specifies the how fast the function decreases, a good value for **b** would be initializing it with the length of the lane line. The alorithm is similar to Gradient Descent:



Sampling the lane line means taking the intersection between the step of sampling on the line and the scanline. This intersection will result in a stream of values in the domain {0,1} as the value 2 is not treated in this paper.

The final logic states that a transition between 0 to 1 or 1 to 0 means a transition from the road surface to the line surface or vice-versa. A total of three transitions or more will make the decision that the line above the scanlines is dashed.
The results after this step and the scan lines expressed by the mentioned Exponential Decay function can be seen in the following picture:

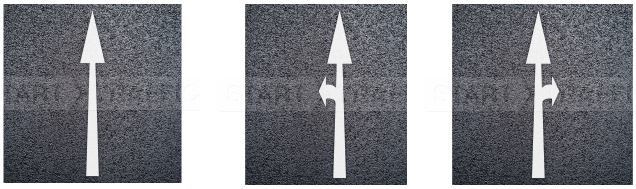
B. Classification of arrows

For each lane in the image a region is determined where an arrow is expected. Segmentation of this region is then performed to extract the arrow. This article uses the sample processed after isolating the white lines and the segmention is no longer required as it was already performed in that step.

The analysis of arrows states the problem of generality because different roads could have different arrow shapes, surfaces or colors. The aquisition of the road sample could result in different lighting, positions and orientations for the arrows. Threrefor four methos were experimentally used : computation of shape elongation axis (**B.1**), template matching with known templates **(B.2)**, feature matching by Scale-Invariant Feature Transform (**B.3**) and one original approach. From all these four the last one (**B.4**) performed the best.

B.1) The elongation axis gives information about the arrow orientation however the aquisition of samples from different positions made confusions between a straight-left arrow or straight-right arrow. This is a robust method only if the aquisition is made in the same conditions.

B.2) The bounding box of the extracted arrow is scaled to the size of templates using sum-of-squarred-differences. Template matching is then performed on both the final bounding box and the known arrows (templates) [1].



Template matching identifies parts on a input image that match a predefined template (reference image or arrows in our example). The aim is to find all the input image locations at which the object from the template image is present. To express the success of identification a measure of similarity is used between the template and inspected image, called *image correlation.* The method choosen for correlation is *Normalized Cross-Correlation* which is a robust measure of image similarity scaled in [-1,1] and invariant to brightness changes:

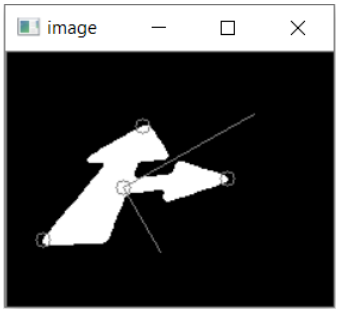
$$NCC(I_1, I_2) = \frac{1}{N\sigma_1\sigma_2} \sum (I_1(x,y) - \mu(I_1)) \times (I_2(x,y) - \mu(I_2))$$

where I_1 represents the inspected image, while I_2 the template image. $\mu(I_t)$ is the average of the image intensities.

B.3) SIFT(Scale-Invariant Feature Transform) is a feature detection algorithm used to detect and describe local features in images. SIFT keypoints are extracted from reference images and stored in a database. A test image is used to find objects by individually comparing each feature from the test image with the database and finding candidate matching features based on Euclidian distance of thei feature vectors. However this approach did not distinguish between the arrows poiting to left or right and arrows points straight because they present the same features with different orientations, as depicted in :

B.4) After extracting the ROI for arrow from the sample on which segmentation was already applied the system expects an arrow to appear in this region. The system captures a sample only when it detects that some white pixels are in this range (an empty road region is not of interest). If a cluster of pixels is located at certain offsets from the bottom and top margins of the arrow ROI could result in a possible arrow being in the region. This is a good way for speed up the processing.
For better results a Guassian filtering is performed of the image to reduce the chances of capturing something else other than an arrow to zero.

After the aquisition of an arrow in the ROI for arrows we need to perform classification in order to determine the type of arrows and influence the decisions to be made. For this this paper makes the following assumptions: the orientation angles of the arrows are between **(0, 70)** degrees relative to the middle axis of the lane, the arrow is situated in the middle of the ROI with minor errors and the arrow markings are of three types (straight, straight-right and straight-left). With these assumptions set the arrow type is completely determined by 6 paramters: its shape center, its minor axis, major axis and three extreme points (the top most point. the right most and the left most one).
For example capturing the following arrow will result in these axis and extreme points:

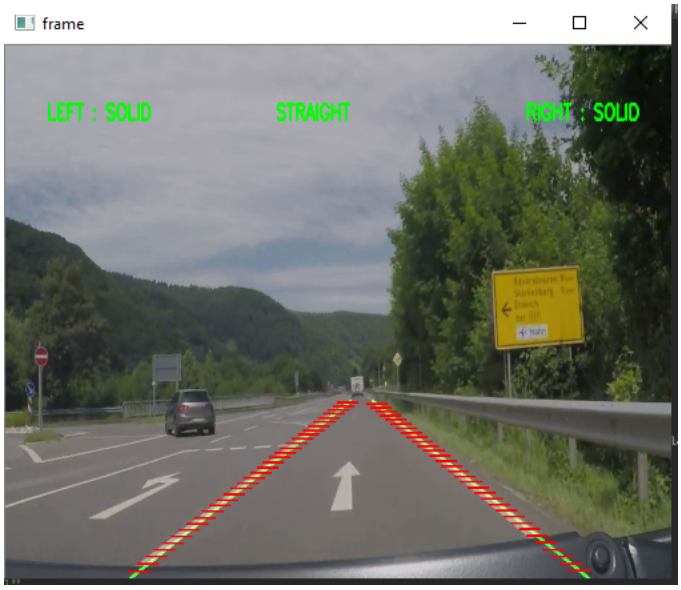


Two vectors are created : the vector of the major axis and the vector from the center of the arrow to the left most point. These two vectors will give an angle that is being analyzed:
if the the angle < 40 degrees then the arrow will be "straight-left" , if it is in [40,90] the type will be "straight" and if the arrow angle is above 90 degrees it will consider the arrow to be "straight-right".

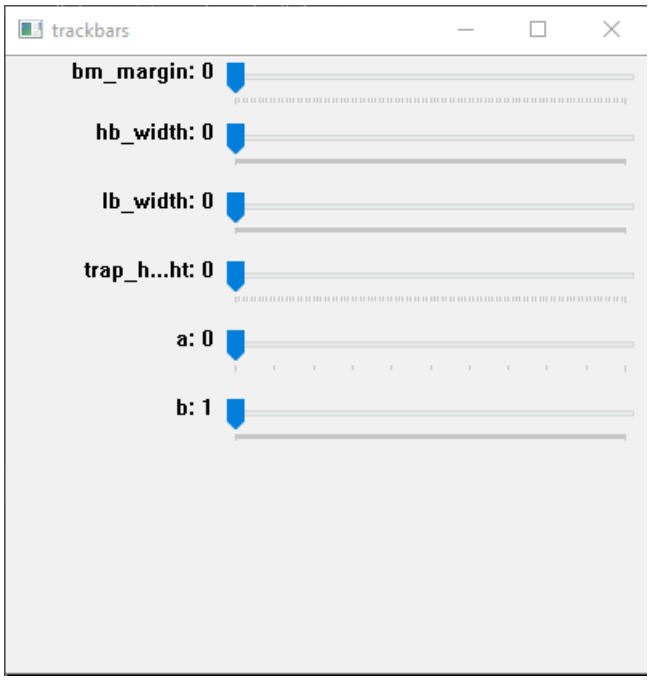
IV. Warning System

At this point the paper addresses only informative messages like announcing the type of arrows, the type of margins and delimiting correctly the lane lane.

For example :



The parameters describing the road could differ from road to road because machine learning was not used here they need no be manually set up. The system was tested on german roads where every kind of road markings were present.
The system parameters can be manually adjusted by the means of the sliders:



Further improvements of the system would be to use a rule-based system that can combine all the informations obtained from the systems described above to assist the driver in taking any actions. This way the messages could be split into

informative and warnings. For example a warning message pops out to inform the driver of possible negative outcomes when trying to overtake a car by crossing the solid line or an informative message is purely informational presenting the driver the option of a lane change.
Rules are written in form of First Order Logic (because it is flexile and variable-based) and the inference engine is the *Forward Chaining* algorithm to find out the action that needs to be taken according to the active rules resulted from the sample.
Forward Chaining is a reasoning method, considered an inference engine and described as repeated application of modus pones. Forward chaining starts with the available data, for example: was there any arrow marking, did the car crossed a solid line, is there any error in the system, et al; and uses inference rules to extract more data until a goal is reached. The goal can be interpreted as a suggestion materialized in a warning or informative message.

RERENCES

[1] *Stefan Vacek, Constantin Schimmel,R¨udiger Dillmann, Road-marking analysis for autonomous vehicle guidance*

[2] Nicholas Apostoloff , Alexander Zelinsky, Robust Vision based Lane Tracking using Multiple Cues and Particle Filtering

[3] Sio-Song Ieng, Jean-Philippe Tarel, Raphael Labayrade, On the Design of a Single Lane-Marking Detector Regardless the On-board Camera's Position

[4] Kristijan Ma˘cek, Brian Williams, Sascha Kolski, Roland Siegwart, A lane detection vision module for driver assistance

[5] Carsten Steger, Clemens Glock, Wolfgang Eckstein, Helmut Mayer, Bernd Radig, Model-Based Road Extraction from Images