Image Analysis and Computer Vision

Lane detection on curved roadways and computation of car relative position

William Stucchi
Giada Silvestrini

A.Y. 2022-2023

Contents

1 Problem definition

Nowadays one of the hot research and application fields for image analysis and computer vision is the automotive industry. In particular, the precision and reliability of this image manipulation techniques are key aspects of the transition towards the autonomous drive. The objective of our project is, in fact, related to the analysis of videos in order to detect the position of the car with respect to the lane, accounting the possibility of curved roads and challenging environmental conditions in general. This is the first step of the realization of more complex software controllers that allow the car to reach some levels of automation, such as the lane assist and trajectory following.

2 State of the art

In the literature the problem has been already approached in many different ways. The majority of them implement filtering techniques to identify the lane lines that are then estimated with some fitting algorithms. In addition, some most advanced solutions require the implementation of machine learning and deep learning algorithms that train a neural network in order to adapt the program parameters according to the characteristics of each input video and, doing so, allow the algorithm to be as general as possible.

3 Overview on our approach

Our approach is composed of different subsequent steps: first of all we proceed to calibrate the camera we used to film the videos; from there, we create the Bird-Eye View of an area of the road where we expect to find some lane lines. Successively we filter the image to discard some outliers and to highlight the points that probably are of our interest.

Next we properly identify the lane lines and we compute the offset of the car with respect to the center of the roadway. Finally, we print on screen the relevant information such as the just computed offset, the turning direction and detected lane boundary.

4 Our implementation

4.1 Camera calibration

The aim of this step is to find the intrinsic parameters of the camera, in order to correct the eventual image distortion introduced by the camera we used to shoot the videos. For this purpose, we implement a standard algorithm that requires the use of different images of the same chessboard, from different positions, taken by the considered camera. Using some OpenCv built-in functions, we implemented the Zhang method from which we obtained the calibration matrix and the undistortion parameters. While implementing in practice this method, we noticed that the camera we used already produces undistorted images, because applying a correction resulted in a worse behaviour throughout the overall execution. For this reason we chose to ignore the result of this computation because it led to worse performances.

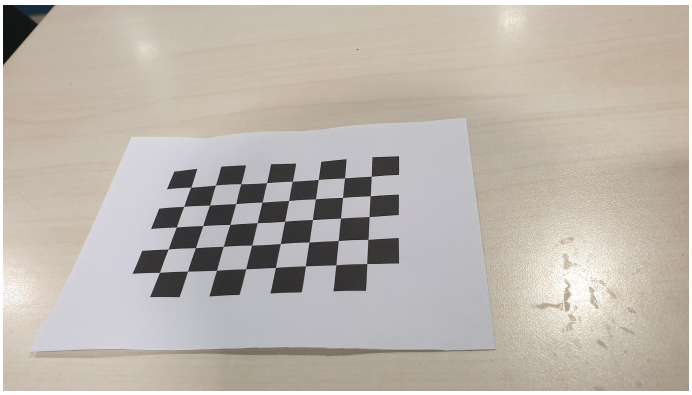

Figure 1: Chessboard used for calibration.

4.2 Preprocessing

4.2.1 Frame extraction

After the calibrations step, we can proceed to prepare the video for further computation. In particular we slice the input into consecutive frames on which the program will apply the next steps, individually.

Figure 2: First frame.

Figure 3: Consecutive frame.

4.2.2 Region of interest

One of the key steps in the process is the identification of the region of interest (ROI), which is the portion of the image in which we expect to find the lane lines. Initially it is hard-coded and fixed, while during the execution it automatically adapts, based on the position of the extracted lines in order to tightly follow the lane through the frames. In the initialization step we set the four vertices of this trapezoid such that it is highly probable to detect at least a portion of the lines, both the left one and the right one. If this condition is not satisfied the program will not start executing.
For all the successive frames, we set the ROI according to the position of the lines detected in the previous frame: in particular, we look for the horizontal position of the points that belong to the lines and also intersect the upper and lower bases of the trapezoidal ROI.

The vertices of the ROI of the next frame will be set to the just found positions, with the addition of a proper margin, that will maximize the chances of keep detecting the lane lines.
If by any chance the lines are not correctly detected due to prohibitive light conditions, disturbing horizontal signage, or any other unexpected event, the ROI modification may lead to an unfeasible status, and consequently its dimension will be reset to the original one for the next frame.
This whole procedure allows to remarkably increase the robustness of the algorithm (in fact it is able to recover from undesirable situations) and to tolerate carriageway width and curvature changes up to a reasonable amount.

Figure 4: Original ROI.

Figure 5: Adjusted ROI.

4.3 Bird-Eye view

A Bird-Eye view (BEV) is an elevated view of an object or location from a very steep viewing angle, creating a perspective as if the observer were a bird in flight looking downwards. For our purpose we have to transform the just selected ROI of the frame into it's BEV version. This step is useful because in the BEV perspective the lane lines will result parallel, in case of a straight roadway, and almost parallel in case of a curve. This feature simplifies the future identification

steps.
Firstly, we select the ROI as explained before, then we select a set of 4 points to which each vertex needs to be mapped to. After that, we can compute the transformation matrix through the getPerspectiveTransform function of OpenCv. We also compute the inverse transformation matrix because, as last step, we need to transform back the frame to its original perspective.
These two matrices are computed frame by frame due to the fact that the ROI is dynamically adaptive. This means that the four corner points change coordinates throughout the execution, while the end points are fixed. Thus, the transformation matrix and the inverse transformation matrix change frame by frame.

Figure 6: Bird-Eye view of a frame.

In figure 6 we can notice that the lines are not perfectly vertical: this is because the camera used was slightly tilted with respect to the longitudinal axis of the car.

4.4 Image filtering

A fundamental step of our recognition approach is the transformation of the image into a new colour space. In fact, in order to better identify lane lines it is useful to change color space from the standard RGB to one that highlights better the road features, in particular the difference between the road lines and the asphalt. We chose as the new colour space the HLS colour code. It is composed of three channels: the H one which represents the hue; the L which represents the lightness; the S which represents the saturation. Among these three the one we are more interested into is the lightness, because it allows to better distinguish the colour differences, as we know that the lines are usually of a very light colour, such as white or yellow, while the asphalt is typically gray or dark gray.
After the extraction of the L channel, we can proceed to apply an intensity transformation function. In particular we applied the CLAHE (Contrast Limited Adaptive Histogram Equalization) method to equalize images. CLAHE is a variant of Adaptive histogram equalization (AHE) which takes care of over-amplification of the contrast. CLAHE operates on small regions in the image, called tiles, rather than the entire image. The neighboring tiles are then combined using bilinear interpolation to remove the artificial boundaries. This algorithm can be applied to improve the contrast of images and in our case to deal with rapid luminosity changes, such as in entering and exiting tunnels or shaded areas of the environment.
The successive step is to define two thresholds that we use to select pixels that are probably part of the lines from the ones that are not. In the first case we set them to 255, which represents the white colour, otherwise we set them to 0 which is black.
As the last step of the filtering process we perform a split of the frame into its left part, which should contain the left lane line, and its right part, which should contain the right lane line. Moreover we decided to leave a central space between the left and right division in order

to cope with the fact that may be present some horizontal signage on the carriageway, such as arrows or writings, that may disturb the line recognition process.

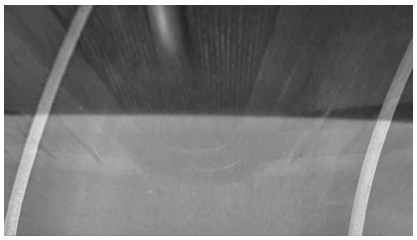

Figure 7: CLAHE applied on the l-channel.

Figure 8: Thresholding of a frame.

4.5 Lane detection

In this step we adopted a solution that requires the implementation of an iterative window approach that in order does:

1. Apply an histogram filter to detect the possible centre of the lines: this step consists in counting how many white pixels are present for each column of the image. The aim is to extract the x coordinate that represents the column that contains the most number white pixels, both for left and right line.

2. Final sample selection with window approach: the purpose of this step is to identify a region in which the pixels of the line are expected to be. At each iteration of the method, all the white pixels that are located inside the current window are added to the list of the sample points representative of that line (left or right according to the considered window centre). The window is initialized on the bottom of the image, and centered on the x coordinate

of the peak that was previously identified. At the next iteration the window is shifted in height for a number of pixels equal to its height (to avoid the overlapping of consecutive windows) and it is centered around a possible new x coordinate that is computed as the average of the x coordinates of the samples that have been found inside the window in the previous iteration. If the number of pixels found is not sufficient (it is not grater than a threshold) the new centre x coordinate will be unvaried. This step ensures that the position of the window follows the trajectory described by the line. By the fact that the height of the window is fixed, the procedure stops when the window has been moved upwards for a number of times that allow to cover the entire image height.

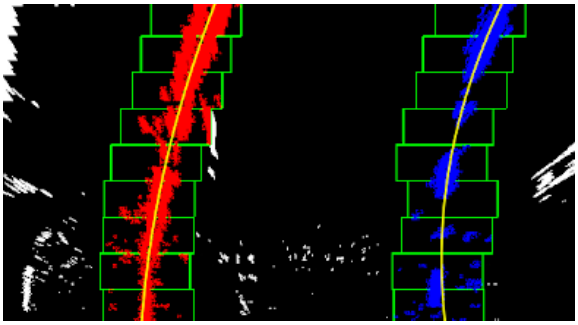

Figure 9: Window approach example.

3. Model fitting of the curve: a mathematical model of the carriageway lines is obtained by fitting a second order polynomial to the points of each line, found at the previous step, using the Numpy library and its polyfit function.

4. Draw lines and highlight detected carriageway: from the points that belong to the lines found at the previous phase, we are now able to draw the detected lines of the lane and colour a region of space that represents the detected carriageway. Such area will be

composed by all the points that have coordinates that lie among the left and right lane lines.

4.6 Car offset and curve direction computation

4.6.1 Car offset

To compute the offset of the car from the centre of the lane we need to make some assumptions:

- The camera from which we took the videos is considered to be centered with respect to the car
- The camera is also assumed to be in a fixed position that reside on the longitudinal axis of the car, even though a small rotation along the vertical axis is tolerated
- The lane is assumed to have a width of 3 meter
- We assume that 3 meters translate into 700 pixels

Under this hypothesis we can compute the car offset with respect to the centre of the lane according to the following mathematical model:

$$car\_offset = (\frac{frame\_width}{2} - \frac{xl + xr}{2}) * xm$$

$$where$$

$frame\_width$: the number of pixels in the width of the image,

xl : the x coordinate of the base of the left lane

xr : the x coordinate of the base of the right lane

xm : it represents the conversion between meters and pixels

4.6.2 Curve direction

To determine the direction of the curvature of the roadway we consider the two polynomials found previously, that represent the lane lines. Out of this two, we select the one with the bigger first coefficient because its corresponding polynomial is the one that represents better the curve direction. Of such polynomial we can make considerations about the direction of the curve.

Considering a second order polynomial of the type $ax^2+bx+c=0$, we can say that:

$$\text{curve direction} = \begin{cases} forward & \text{if } |a| \leq threshold \\ right & \text{if } a > threshold \\ left & \text{if } a < -threshold \end{cases}$$

where *threshold* is a value small enough to consider the lane to be straight.

4.7 Resulting scene

After all the computation is done, we can proceed with the reconstruction of the original scene. In fact, we can now apply the inverse transformation matrix to obtain the original perspective view of the frame. To that, we add information found at the previous points, such as the detected lane lines and roadway lane, the computed car offset and the direction of curvature.

Figure 10: Left curve.

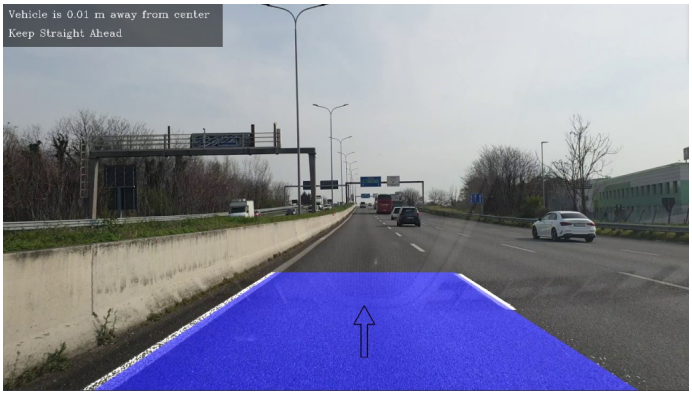

Figure 11: Straight.

5 Experimental results and analysis

During the testing phase of our project, we analysed the performances of our code with different self-made input videos. In the process we encountered some issues that we handled, for the most part, with the result of improving the resilience against unexpected behaviours.
One of the unsolved problems is the definition of the initial dimension of the ROI: if it is too small than the carriageway, then the program is not able to start because it cannot identify correctly the lane lines, while if it is too big it may include unwanted road features (edge of

the pavements, lines of the neighbour lanes, etc.). All of this may lead to a failure of the execution. A way to deal with this problem is to change manually the initialization of the ROI, but this would imply a loss of some degree of generality. This problem can also occur when the distance between two dashed lines is bigger then the height of the ROI, so that for certain frames it might happen that no dashed line is contained in it. Additionally, the generality of the program is limited by the parameters of the thresholding. In fact, some videos can require a refinement of the tuning of such parameters, but this change may result in sub-optimal performances with other videos.
Furthermore, we encountered issues when dealing with the horizontal signage on the road. To deal with this we decided to partition the frame in two sections: one section is related to the left of the lane, while the other refers to the right part. Such areas are divided by a zone where no detection is applied and all the samples are discarded. If by any chance the horizontal signage is able to escape this zone, then it is detected leading to an erroneous behaviour. Lastly, we dealt with lighting issues: the dynamic histogram algorithm helped us in managing change in light conditions when entering or exiting tunnels or when traversing shaded areas, but in some occasions it is not enough (e.g.: when a car with low beams on comes from the opposite direction).
The main solution we adopted in order to keep the program functioning despite these unexpected behaviours is to recover the original region of interest when the one computed is not a feasible and correct one.

We made some trials regarding the addition of other nice features. For example we tried to compute the radius of curvature but it resulted to be a bit too noisy, in fact it was not reliable. Also, we tried to better handle the lane changes, but at the moment it works just because we implemented the restoration of the region of interest, and not really following the trajectory of the car.

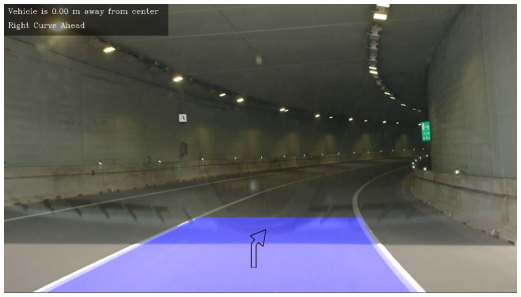

Figure 12: Entering a tunnel and change in light conditions.

Figure 13: Disturbance of a car with low beams on.

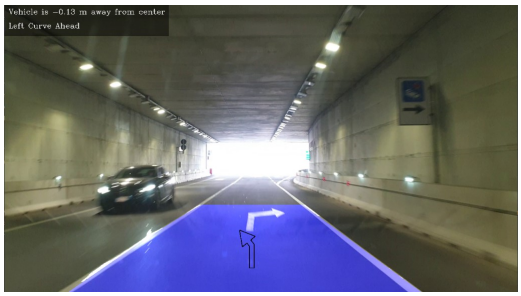

Figure 14: Correct handling with horizontal signage.

Figure 15: Erroneous behaviour with horizontal signage.

6 Conclusions

After having recorded our own videos of non-specific roads, trying to include as many different environmental conditions as possible, we arrived at a solution that satisfies us. From the empirical results we have obtained on multiple videos, we have seen that our approach is, in general, satisfactory and well working. In particular we improved in the handling of rapid changes in light conditions, for example when entering and exiting tunnels. We also put our focus on managing quite curved roads without loosing the detection of the lane. Furthermore, our implementation deals well enough with unexpected situations, being able to recover itself if the detection of the lane lines is momentarily lost. Beyond that we are well aware that our approach still has a re-

markable margin of improvement. In fact, we found issues when the direction of the camera is too rotated towards one side, because the initial frame does not include the lines in the lower region of the image. Also, if the width of the roadway is too broad to the point that it does not fit in the ROI, our solution is not able to correctly identify the lines.

In conclusion, besides the above-mentioned circumstances, the main development direction can be focused on the generalization of the parameters of the thresholding and filtering of the frames, plus the definition of the initial ROI, which should be able to detect the lane independently of its width.