Image Analysis and Computer Vision
A.Y. 2022/2023

Project work report

Road lane identification and vehicle position estimation

Rognoni Filippo Maria
Vittorini Francesco

1. Goals and motivations

An important application of Image Analysis and Computer Vision techniques is in the autonomous vehicle field. One of the key requirements for an autonomous car is the ability to identify the road lane limits and keep the center of the road starting from the images coming from a camera mounted on the vehicle chassis. Main goal of this project is to identify the lines that define the road lane and to estimate the offset of the vehicle with respect to the center of the lane starting from the images taken from the camera. Then the process is applied also on a video. This is the starting point for designing a controller able to make the car able to follow the trajectory described by the road and keep the center of the lane.

2. State of the art

In the literature there are already different approaches to solve problems similar to ours. The majority of them are based on the application of a Sobel-like filter in order to detect the edges of the lines (starting by the assumption that the color of the line is different enough from the color of the road) and on standard fitting algorithms such as least-squares or RANSAC to estimate the best approximation of the line starting from the candidate pixels belonging to the line.We got inspired from and followed, on the other hand, an approach based on some different ideas to select the candidate pixels belonging to the lines.

3. Outline of the algorithm

The algorithm is characterized by different subsequent steps, we will analyze each of them to identify their main purposes and the key idea behind them.

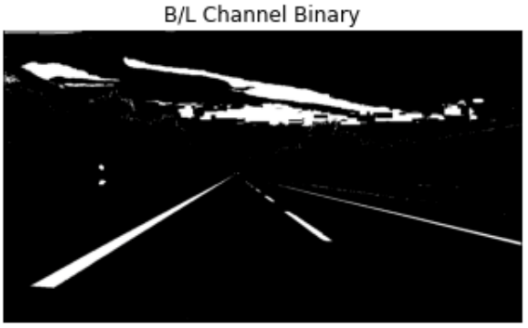
3.1. Camera calibration

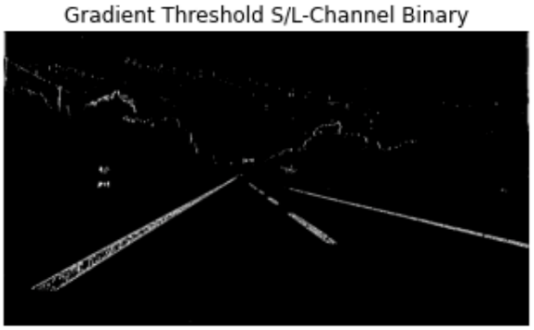
The calibration procedure is needed in order to estimate the parameters of the camera mounted on the car. Among these parameters, the most used in our application are the distortion coefficients that are used to correct some distortion effects that are present in the images coming from the camera. Some modern cameras already have an embedded distortion correction mechanism, in this case this step of the algorithm is not needed and we can directly use the images coming from the camera, without applying any modification.
The calibration needs a few images of a chessboard with a known shape (in our case a 6x9 chessboard) taken from different viewpoints and, starting from these images, some functions of the OpenCv Python library are used. As output of this stage we have the matrix representing the camera intrinsic parameters and the distortion coefficients. The next step is the distortion correction of the image taken from the camera, by using the previously estimated matrix and coefficients. Now we have an undistorted image on which we can perform our analysis and operations.
The following image represents our reference image on which we will perform the lane lines identification.

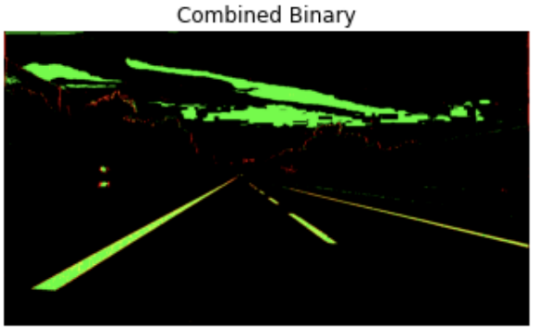
3.2. Channels transformations

One of the most important steps in the lane identification process is the phase in which we put in evidence the side lines of the road.
In order to achieve this purpose we need to work on the color space of the images and also to filter out the weak association with the colors (white and yellow) that are the most significant for us.
We tried different approaches to put in evidence the side line of the street. The first one is based on exploiting LAB and LUV color spaces.
In LAB colors are expressed with three parameters: “L” the percentage of lightness, while “A” and “B” represent the four unique colors of human vision(red, green, blue and yellow). LAB is also useful in situations when light is reflected by surfaces.
On the other hand, LUV colors are expressed as before with three parameters: “L” stands for the luminance, while the parameters “U” and “V” stand for the values of green/red and blue/yellow respectively. LUV is more useful in case we have surfaces that emit light.
At first we convert the RGB image into LAB or LUV image and after channel separation we filter out the weak association of the color of interest, in our case white and yellow.
Another approach that we used was to combine the HSL color space with the Sobel filter.
HSL is a color space in which the three components are: “H” the hue of the color, “S” the saturation and “L” the lightness of the color. This pattern is very useful in object detection techniques because the RGB components of the object color are all correlated with the amount of light hitting the object, so descriptions of the image made in terms of hue/lightness/chroma or hue/lightness/saturation are more useful to identify object.
Going through, Sobel operator is very widespread in computer vision, especially in edge detection. The Sobel operator is based on convolving the image with a small, separable, and integer-valued filter in the horizontal and vertical directions and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation that it produces is relatively crude, in particular for high-frequency variations in the image.
In our approach we converted the image from RGB to HSL and we isolated the “S” channel, filtering out the weak saturated elements of an image. After that we applied the Sobel filter to identify the side lines of the street.

After some tests we decided to use the first method described to identify side lines, because the performances in the majority of the cases were better than the other one. Some examples of the previously described process could be found in the “Threshold Binary Image” section of the code.

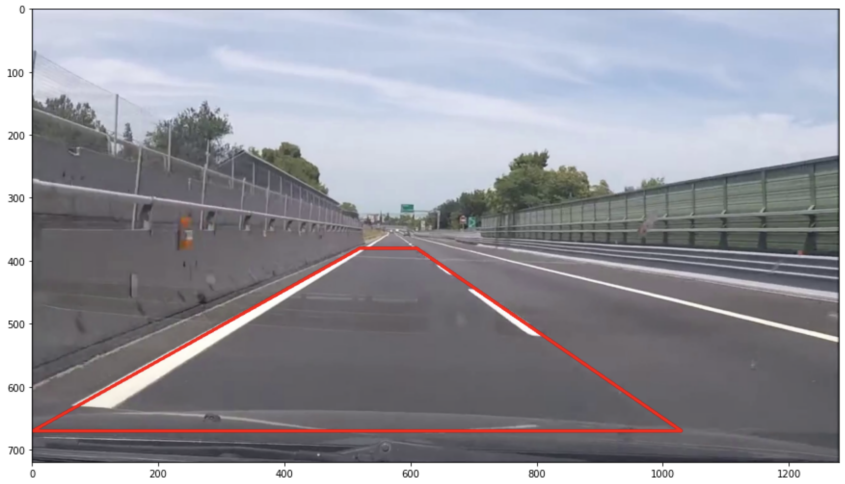




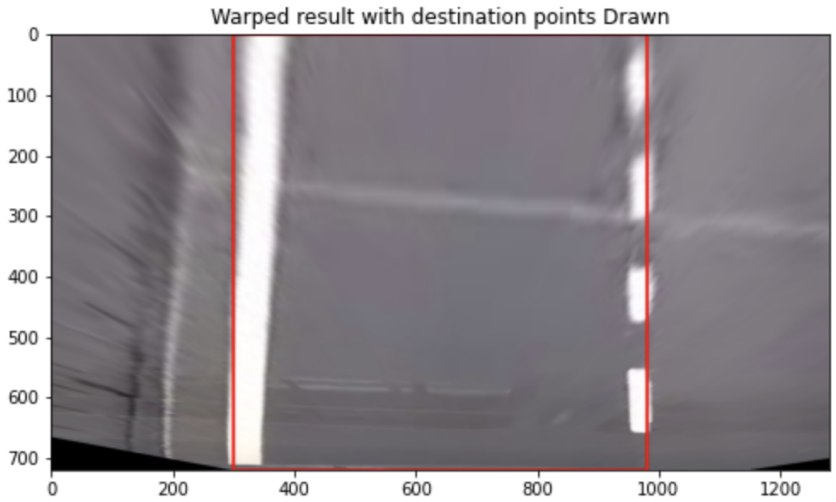


3.3 Bird-eye view

In order to find the side lines, we must perform a perspective transformation in order to have a bird-eye view of the scene, in which lane lines are parallel and the line identification procedure is simpler.
The first step is to choose a planar area which will be our region of interest on which to apply the perspective transformation. This area will be the region of the image in which the road lane is expected to be and it is assumed to be fixed during the motion of the car. The following figure shows the selected region of interest on top of the reference image.

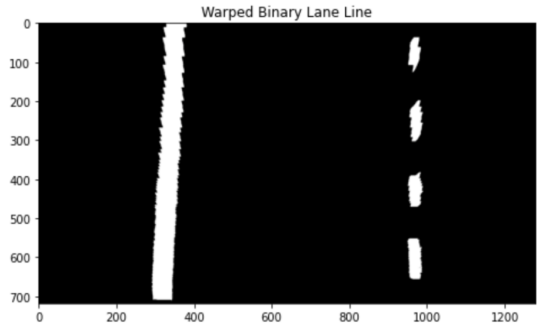


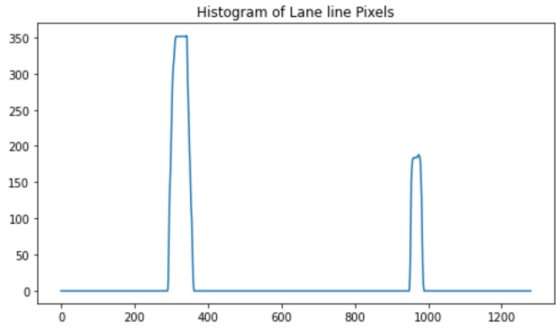
The next step is to determine 4 pairs of corresponding points in order to estimate the desired perspective transformation. They have been chosen in order to map the vertices of our region of interest onto the vertices of a rectangle and, as a consequence, to have the lane lines parallel.
Now that we have the corresponding starting and final points, we can estimate the 3x3 perspective matrix, which describes the homography between the selected points and the final ones.
The final step is to apply the perspective transformation to all the points belonging to the region of interest and to obtain the transformed image with the needed bird-eye view. We set an offset in order to also include in the projection of a margin region around our region of interest.
Below we display the obtained results on the reference image.



3.4. Pixels histogram and peaks identification

Once we have obtained a binary image of the bird-eye view of our region of interest we are now ready to identify lines. In this image white pixels represent candidate pixels belonging to the line, black pixels are those belonging to the background. The first step consists in counting how many white pixels are present for each column of our image (we count only the lower half part of pixels in each column in which the projection is less affected by noise). Here the results on our reference image are displayed.



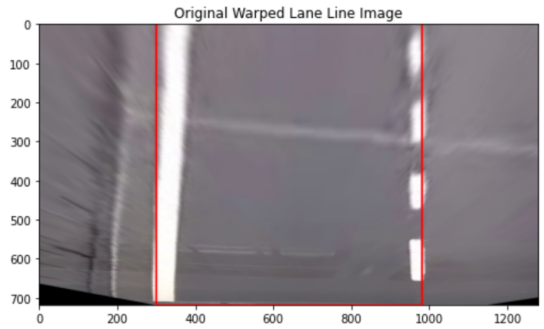


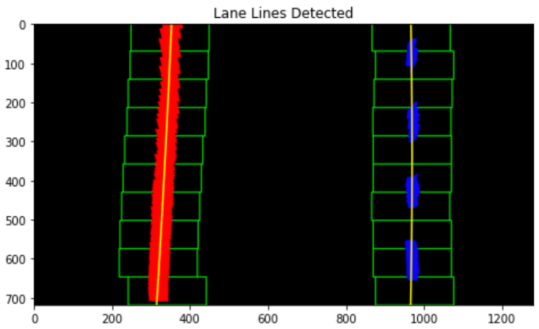
The goal is to identify the x coordinate for which we have the maximum value of white pixels, hence it is a good candidate for the center of the line. In particular, we are searching for two lines: one on the left part of the image and one on the right as we are expecting the vehicle to be between the two lines delimiting the lane.

3.5. Sliding window technique for lines identification

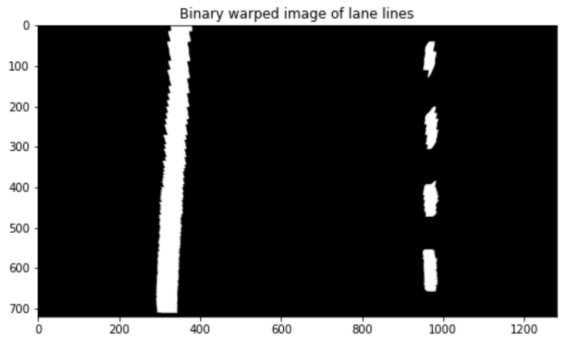
The sliding window technique is an iterative procedure to select the pixels that belong to each line that then are used as the final version of the samples on which fitting the curve representing the mathematical model of the line. The purpose of this window is to delimit the region in which the pixels of the line are expected to be and are considered valid. The height and width of it are parameters of the algorithm. In the iterative procedure, at each iteration, all the white pixels that fall inside the current window are added to the list of the sample points representative of that line. At the beginning the window is positioned on the bottom of the image and centered on the x coordinate of the peak that was previously identified (the procedure is exactly the same for the left and right line). At the next iteration the window

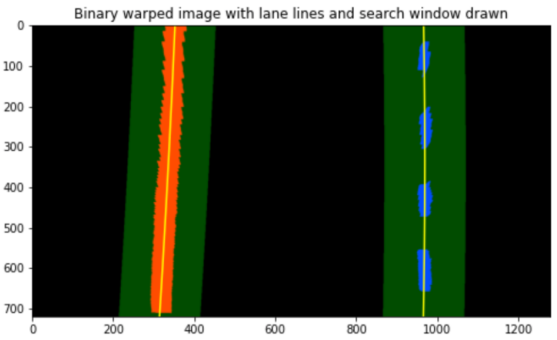
is shifted in height for a number of pixels equal to its height (so the different windows never overlap, they are exactly one on top of the other) and it is centered around a possible new x coordinate that is computed starting from the number of samples that have been found inside the window in the previous iteration. In particular, if the number of samples is greater than a prescribed threshold, the new center of the window will be the mean x coordinate of the found points, otherwise it will stay the same. This step ensures that the position of the window follows the trajectory described by the line. The procedure ends when the top of the image is reached and the entire line has been analyzed. As output of this step we have two lists (one for each line) containing the coordinates of the points that are considered representative of the line. A first mathematical expression of the curve representing the line is obtained by fitting a second order polynomial to the points of each line using the polyfit function of the NumPy Python library. Below the results obtained on the reference image are displayed.





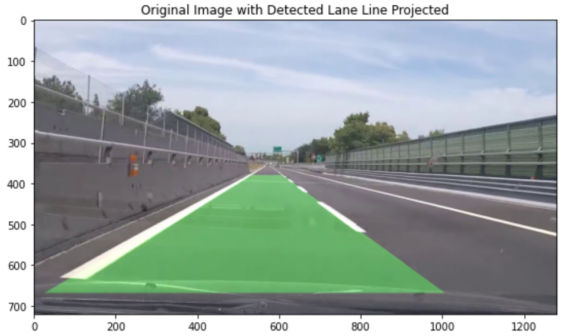
On the right image the red points are the points belonging to the left line, the blue points are the points belonging to the right line and the yellow curves represent the models obtained for the two lines by fitting the respective points. The different windows are displayed with their green borders.
After this step an additional refinement is added in order to discard as many outliers as possible, by selecting points around the fitted lines below a certain margin. The selected points are then again fitted with a second order polynomial. The image below shows the final result and puts in evidence in green the region around the fitted line in which points are considered.





3.6. Projection of the identified lines on the original image

The identified lines are projected onto the original image by simply applying the inverse projection of that applied to obtain the bird-eye view. Furthermore, a green region between the two identified lines is added in order to put in evidence the identified road lane delimited by the two lines. The following image represents the obtained results.



As it can be seen the results are pretty good and the software successfully identified the lines.

3.7 Estimation of the car position with respect to the lane center

Once the lane lines have been identified, the final step is the estimation of the position of the car with respect to the center of the road lane. Assuming that the camera is positioned exactly at the center of the car, the position of the vehicle is expressed as the distance between its center and the midpoint of the road lane. When the vehicle is exactly at the center of the lane, this distance will be equal to zero. Starting from the bird-eye view and the model of the fitted lines, the position of the car is considered as the center of the

image, while the center of the road line is computed as the midpoint between the x coordinate of the left line point at the bottom of the image and the corresponding x coordinate of the right line. The formula for computing the offset expressed in pixels is the following:

$$offset = x_{car} - ((\frac{x_{right} - x_{left}}{2}) + x_{left})$$

where x_{car} is the center of the image (representing the position of the car), x_{left} and x_{right} are the x coordinates of the left and right line point respectively.
Our final goal is to express this quantity in centimeters, so we need to define what is the correspondence between the width of each pixel and the corresponding distance in the real world. We made an assumption about the width of the road lane to be equal to 3.7 m, as this is one of the most common lane widths for highways. In the bird-eye view the width of the lane is 700 pixels, so the width in meters of each pixel is equal to 3.7 m / 700 pixels. Multiplying the value of the offset expressed in pixels by this quantity we obtain the offset expressed in meters, so we can easily obtain its value in centimeters. This is the obtained result.

4. Conclusions, problems and possible future developments

The illustrated approach for lane line identification is a rather simple solution that allowed us to reach the goal of this project. The results we obtained seemed to be satisfactory on the images and videos we found. This material, however, is characterized by a very clear distinction between lane lines and road pavement that made it easier for the algorithm to work properly. We tried to acquire our own images taken with our phone in our own car to understand what is the generalization ability of this algorithm. Unfortunately we saw that, in most of the roads around us, the lines are not as evident as in the reference image and in our experiments the algorithm has not been able to properly identify the lines. Another problem we encountered was the inability of the algorithm to work correctly when the light conditions of the scene change rapidly and with a transition phase in which no lines are visible. An example is the moment in which the car enters in or exits from a gallery. In these situations, for some frames, the light conditions are very poor and no line can be identified. Starting from these problems a possible future improvement of the algorithm can involve the development of a better strategy to distinguish lines from the road even in very difficult light conditions, as an autonomous car cannot tolerate moments in which the road lane identification is not available.