Lane detection on curved roadways and computation of car relative position

Image Analysis and Computer Vision

A.Y. 2022-2023

Giada Silvestrini – William Stucchi

Outline

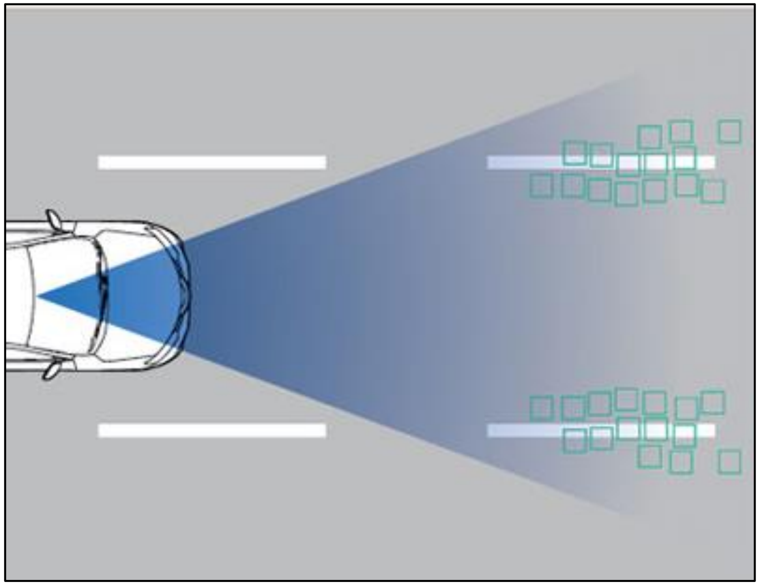
1. Problem definition
2. State of the art
3. Our implementation
    I. camera calibration
    II. Preprocessiong
    III. Bird-Eye view
    IV. Image filtering
    V. Lane detection
    VI. Car offset and cuve direction computation
    VII. Resulting scene
4. Experimental results and analysis
5. Conclusions

Problem definition

The shift towards autonomous driving requires an advanced vision and analysis of the surroundings, especially what concerns the main features of the road. One key aspect is the lane detection and the relative localization of the vehicle, which is also the subject of this project.

State of the art

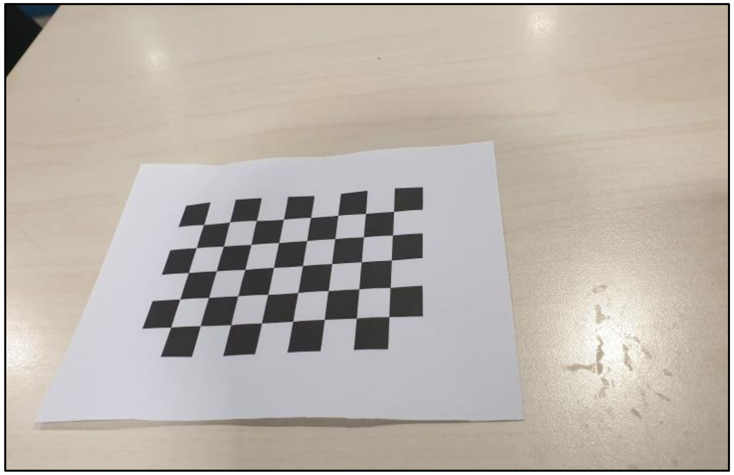
WhiteBox approach

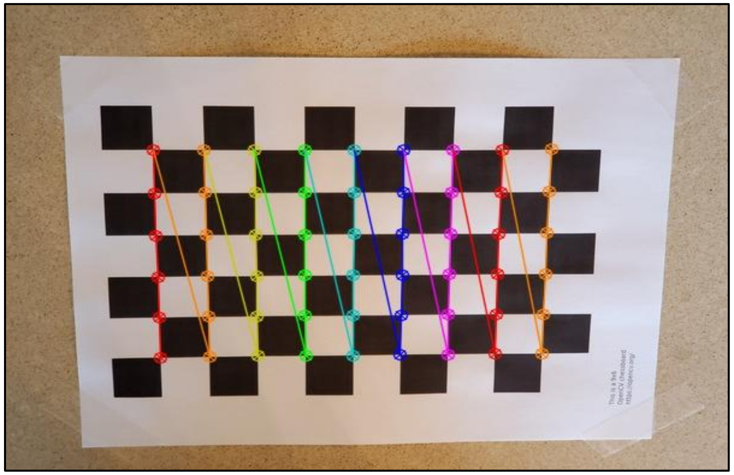


BlackBox approach

Our implementation: Camera calibration

To calibrate the camera of the phone we used to record the videos we used the Zhang method: once we obtained the intrinsic parameters, we hard-coded the calibration matrix that was used throughout the entire process.





Our implementation: Preprocessing

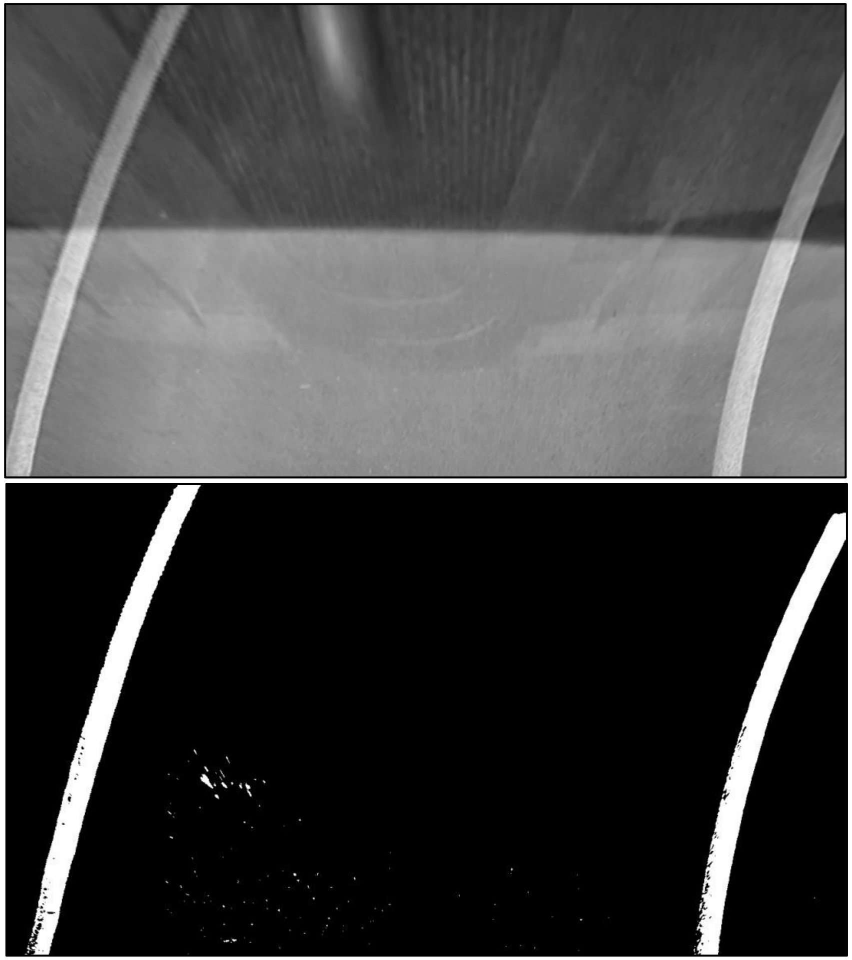
Frame extraction

Region of interest

Our implementation: Bird-Eye view

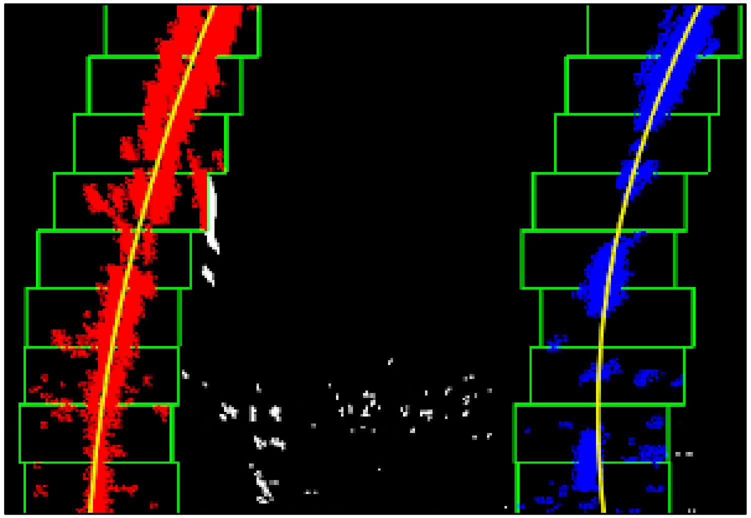
To each frame extracted from the video we perform a perspective transformation on the area bounded by the previously identified ROI.
This will be the frame on which all successive computations and considerations are performed.

Our implementation: Image filtering



1. From the BEV frame, change the colour code from RGB to HLS (hue-lightness-saturation) in order to enhance the most relevant features.
2. Apply an adaptive histogram equalization (CLAHE) to obtain a better contrast and to mitigate some of the issues related to fast changes in light conditions.
3. Apply a threashold on the lightness channel of each pixel to select only the ones that most probably belong to one of the lane lines.
4. From the frame, we define 2 areas: the left and the right, separated by a zone in which no line can lay.

Our implementation: Lane detection



The following steps are executed for both left and right areas:

1. Histogram filtering to detect the centre of the line.
2. Final sample selection with window approach: from the bottom of the image, until the top is reached, define different windows (centered on the centre of the line) in which the samples belonging to the line are located.
3. Model fitting of the curve on the extracted samples
4. Draw the line and highlight the detected carriageway (all the pixels in-between the left and right curves found).

Our implementation: Car offset and Curve direction computation

Car Offset computation

$$offset = \left(\frac{frame_{width}}{2} - \frac{xl + xr}{2}\right) * xm$$

$frame_{width}$: #pixels in the width image
xl: *x coordinate of the base of the left lane*
xr: *x coordinate of the base of the right lane*
xm: conversion between meters and pixels

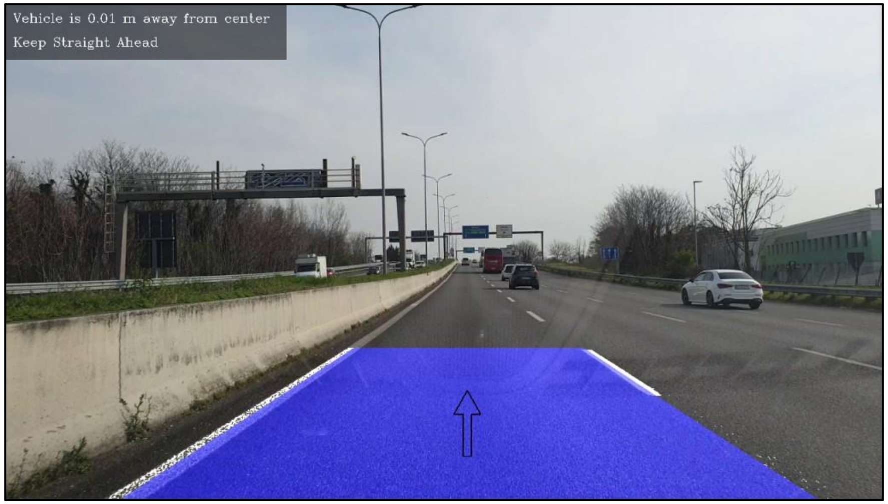
Curve Direction computation

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

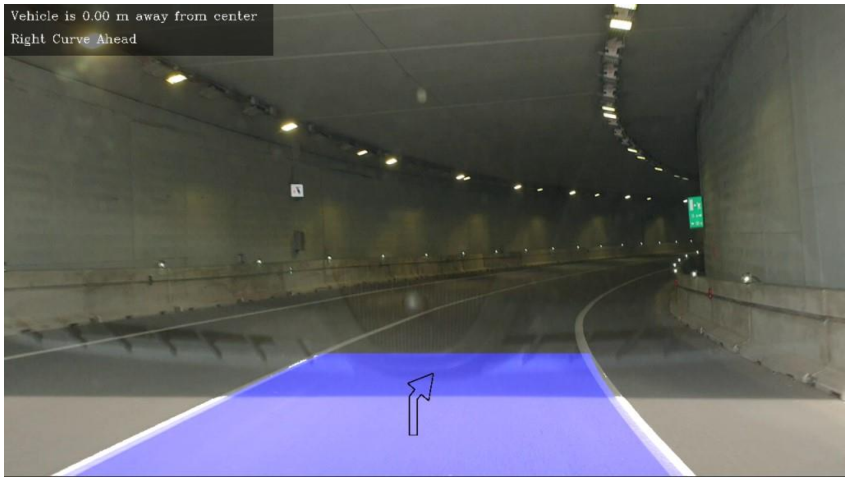
$$direction = \begin{cases} forward, & if\ |a| \leq t \\ right, & if\ a > t \\ left, & if\ a < -t \end{cases}$$

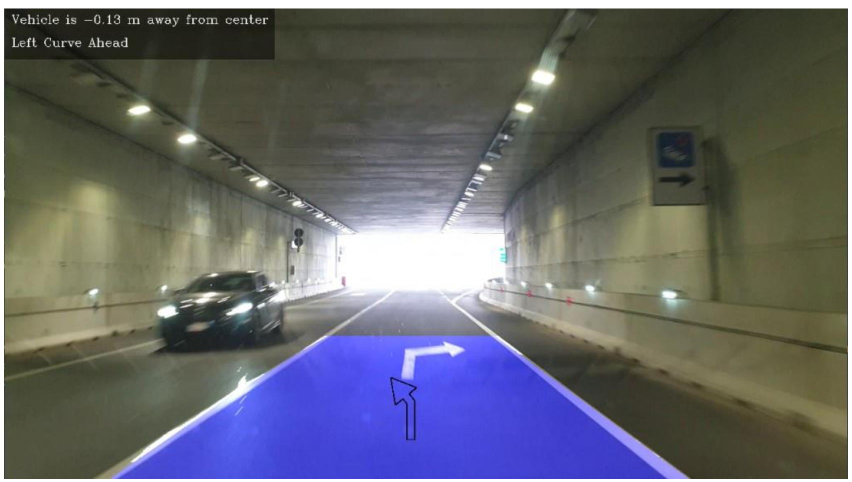
Our implementation: Resulting scene

To obtain the final frame apply the inverse transformation matrix and add the relevant computed information in a box.



Experimental results and analysis





Conclusions

Throughout the course of the project we encountered multiple issues that we were able to manage complitely, or mitigate, such that we achieved a quite robust procedure that can recover when unexpected events are detected.

The solution we proposed can be upgraded with more advanced techniques like Machine Learning algorithms and Deep Learning processes, for the black box and online tuning of the parameters of the filtering stage, in paritular.