POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Road Line Detection and Inverse Perspective Mapping Stabilization

Course Project
Image Analysis and Computer Vision

Milad Naseri Langehbiz, 10645952

Advisor:
Prof. Caglioti

Academic year:
2021-2022

---

1. Introduction

Autonomous vehicle, or more commonly a self-driving car, is a technology that many analysts believe will be widely available by the end of 2050. It is a technology that could enable advanced navigation assistance, smart traffic management, and most importantly, enhanced safety, through systems such as Driver Assistance System (DAS), Adaptive Cruise Control (ACC), Blind Spot Monitoring (BSM), Lane Departure Warning (LDW), etc. Understanding advantages of such technologies, and considering the economic opportunities, nowadays many car manufacturing or computer technology companies such as Tesla, NVIDIA, etc. are investing heavily in these technologies, however, barriers to widespread adoption remain.
One of the main barriers is the cost of this technology, which is mainly due to technologies used for real-time analysis of the surrounding environment in which the vehicle is traversing. Various methods have been developed to reliably detect and analyze the roads and obstacles utilizing a different combination of sensors such as Radar, LiDAR, Ultrasonic, and computer vision-based technologies. Among the above-mentioned sensor systems, a camera setup appears to be the most cost-effective and available sensor setup for commercial vehicles, appropriate for a large-scale manufacturing schema as well as after-sale maintenance, placing computer vision-based systems at the center of attention.
The main objective of such a system is to first, detect the state of the vehicle with respect to the lanes, and then, detect obstacles and objects in the environment, traffic, pedestrians, signs, etc.
Within the realm of real-time computer vision, there are many methods using different techniques and camera setups, ranging from a very expensive multi-camera setup to a very low-cost single front-viewing camera. Considering that nowadays many cars are rolling off the production lines already featuring a front viewing camera setup, developing a robust real-time mono-vision computer vision-based technology would have economic justification suited for products marketed for the low end of the price range.
Utilizing modern AI techniques, it is possible to detect objects and lines with a high degree of accuracy, however, the main challenge in a mono vision setup, is the 3D reconstruction of the scene. 3D reconstruction is defined as transforming information from a 2D perspective space, i.e., the images, or other sensors to a 3D metric space. 3D reconstruction is a necessary step in order to determine the state of each object with respect to the vehicle, for instance, if a car is detected in the front image, then the distance and relative speed between the two cars is the important information that needs to be extracted.
In order to 3D reconstruct the scene, some form of depth information is needed. This information could be acquired through an array of dedicated sensors or using a stereo vision setup, however, in a mono-vision setup and without any kind of dedicated depth sensor, performing a 3D reconstruction becomes challenging.

Nevertheless, it is always possible to define a homography transform between two planes, hence, even in a mono-vision setup and by assuming that the road, lies on a flat plane, it becomes possible to define an Inverse Perspective Mapping (IPM) transform to map the detected lines from the image plane to the ground plane, with a metric scale. In the field of autonomous vehicles, especially in a mono-vision setup, IPM [REF] is very important, since it provides a scale-invariant orthographic view of the scene, which aids in several tasks such as lane marking detection, path-planning and intersection prediction, etc.
This manuscript will explore some of the different methods proposed in the literature for the real-time line detection and IPM in a mono-vision setup, implementing and improving on the state of the art methods by proposing a novel technique for real-time estimation of the relative attitude (i.e., roll and pitch angles) of the vehicle and the camera w.r.t the ground plane and adjusting the IPM transform in real-time for an increased accuracy.

2. Conventions and Dataset

2.1. Reference Frames

Within the entire scope of this work, 5 different reference frames are used these frames are depicted in Figure 4.1 and for more clarity, a brief explanation is given below:
Vehicle: Vehicle's frame of reference, or sometimes referred to as the Body frame, is the reference frame which is centered on the vehicle's Center of Gravity (CG), with x and y axis pointing in forward and left direction of the vehicle's chassis, and z axis pointing upwards. The Inertial Measurement Unit (IMU) installed on the car provides measurements within this frame of reference.
World: The world frame of reference is represented with the coordinate systems indicated by the ISO 8855 [REF], with the origin on the ground below the center of mass of the vehicle, and axis x, y and z pointing respectively forward, left and up.
Camera: The camera reference frame has its origin on the optical center of the camera, with x-axis stretching along the optical axis of the camera. The other two axis y and z have the same orientation as the world frame, pointing left and up, respectively.
Image Plane: The origin of the image plane is set in the upper-left corner, with the u axis pointing right and v pointing down.
Earth: Earth frame, or sometime referred to as the Map, is exactly the Earth reference frame represented with the Global Geodetic Reference Frame (GGRF).

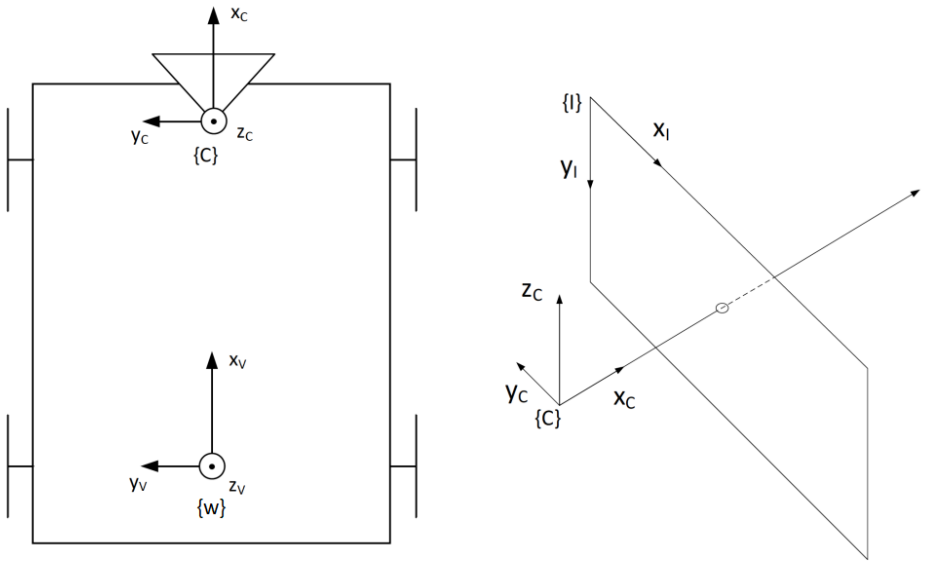


Figure 1: Coordinate systems adopted for world W, camera C and image I reference frames. Adapted from ISO 8855.

2.2. Dataset

The dataset used in this work is [REF]. Recorded in Monza and Arese circuits as well as and Ciruito Giovanni in Napoli, the dataset provides footage and various related data including:

- the images of the vehicle's front viewing camera;
- camera's internal and external calibration parameters;
- centimeter-accurate RTK GPS pose of the vehicle;
- wheel and steering wheel encoders data as well as filtered odometry data;
- inertial measurement data from a 6-axis Bosch MEMS IMU (gyroscope and accelerometer);
- the ground truth of the lines for the inner, outer, and centerline of the Monza and Arese circuits.

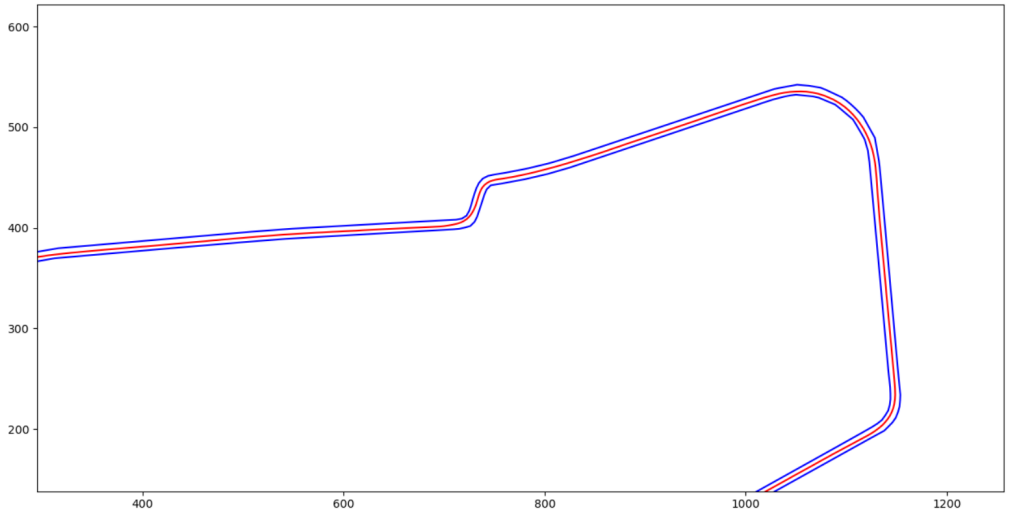


Figure 2: Part of the Monza circuit. Provided by the dataset, the ground truth of the lines (in blue) and the accurate vehicle trajectory (in red) are plotted.

2.3. Line Points

Line points are a set of ordered points sampled on a line, roughly representing the shape of a line.
World points and Front points are respectively, the line points in world frame and image plane.
world/front points right and left are two set of line points representing the right and left lane markings, i.e., the line in the right and left side of the vehicle in the world or image plane, respectively.

3. Preparing Input Images

The dataset used in this project is captured using a 360 camera which incorporates 6 cameras with wide lenses. As seen in Figure 3 the road lines that are straight in reality, appear curved, which is a big problem. In order to be able to use these images, the first step is to rectify the distortion caused by the wide lenses, however, after studying the camera calibration file provided by the manufacturing company, I noticed the actual rectification is highly non-linear with many parameters and the only way to rectify the images was using an application provided by the manufacturer. Figure **??** shows an example of a rectified image.

Figure 3: Original wide image of front viewing camera.

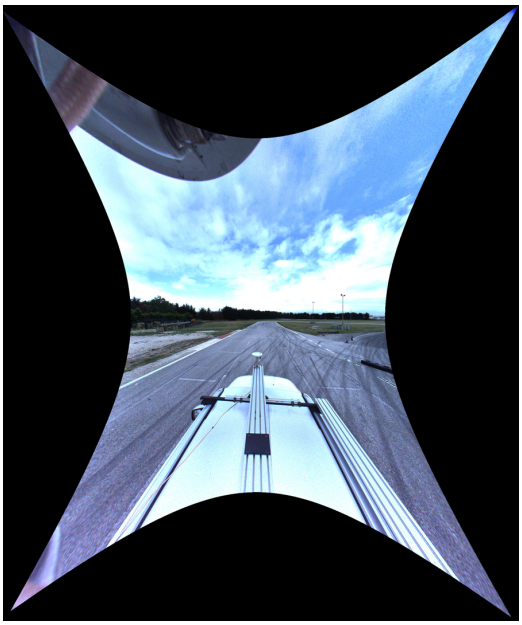


Figure 4: Rectified image of front viewing camera.

3.1. Panorama view

In order to make the rectified images suitable for the CNN they need to be cropped and resized. Moreover, for a better visibility of both of the road lines, especially in the curved sections of the road, we needed a wider view of the road in front that a single camera simply would not provide. The solution for this problem is to stitch the images of the left and right camera with the image of the front camera to create a panoramic view of the front.
The first step in stitching the images is to project the left and right images from their respective planes into the front viewing camera plane, which means we need two homography transforms that we had to compute. In order to compute the homographiy transforms the initial step is to detected the matching keypoints in both of the images.
For this purpose I tried, using the famous invariable feature extraction algorithm such as, however non of these methods provided a reliable and matching keypoints, as shown in Figure 5, these methods would often extract not matching feature points on the road pavement which is mainly considered noise. Having explored using the automatic feature extraction, I decide to manually select the matching keypoints in the images.

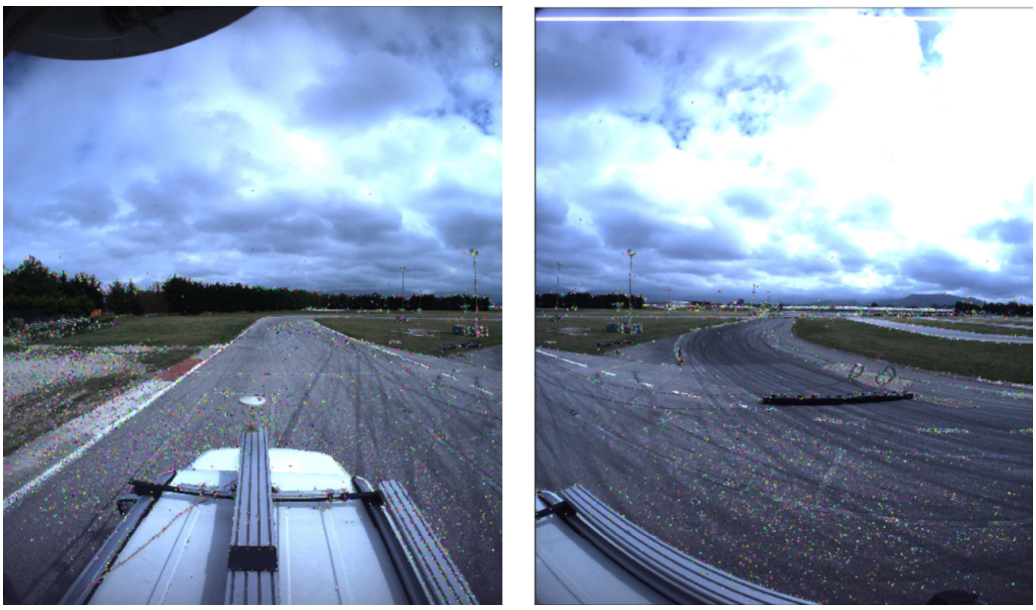


Figure 5: Noisy keypoints extracted via sift.

By going through the images I was able to find a frame that features good amount of orthogonal lines visible on the road, and by manually extending these lines as shown in Figure 6 I was able to use the intersection of the lines as the matching keypoints.

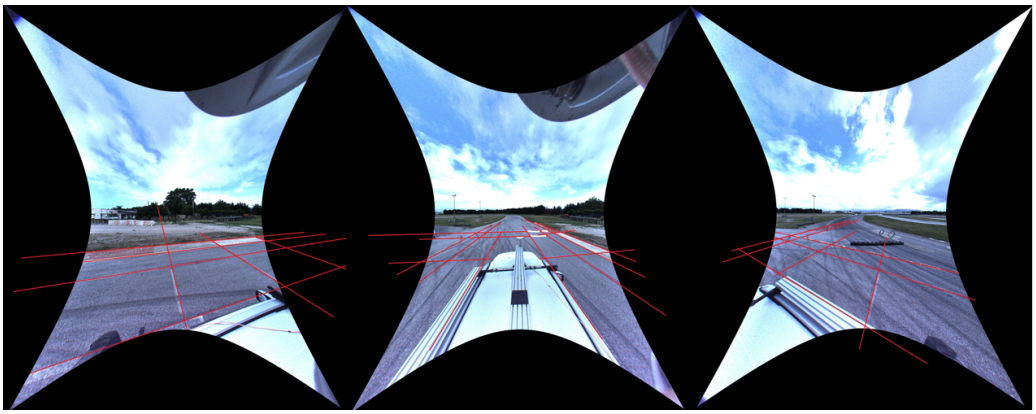


Figure 6: Extending road lines for manual selection of matching keypoints.

After selecting the matching keypoints, I was able to solve a constrained least square problem via SVD and identify the homography transforms. Using the homography transforms and by cropping and resizing I was able to create the perfect panoramic images.

Figure 7: Final panorama image.

3.2. Intrinsic calibration

After creating the panoramic images the intrinsic calibration of the camera is not the same as it was. The new images need a new calibration, which in the ideal case should be done using calibration objects, however I did not have access to such images in the dataset, moreover, I was unable to find frames with good enough information and constraints to perform a self-calibration, therefore I resorted to a different method. Because I had access to calibration of the original images, and the only unknown was the focal length and principal point, I was able to estimate the new calibration by analyzing and comparing the resulted Birds Eye View (BEV) images generated using new images and new calibration against the BEV images using old calibration.

4. Inverse Perspective Mapping

In automotive applications, camera calibration is required to the Inverse Perspective Mapping (IPM) [4] to obtain a top-down view, called Bird's-Eye View (BEV) of the scene. In computer vision, a plane to plane Homography transform is the mathematical relation between two planes, most famously in the form of a 3x3 matrix H or maybe $X(u, v)$ and $Y(u, v)$ when applied on a plane, maps it to another plane (or image). Assuming that the road in front is on the flat ground plane as well as a pinhole camera projection model, Inverse Perspective Mapping is a defined as the homography transform between the image plane and the flat-earth plane. The major benefit of projecting road features in BEV space is to remove perspective effects [5]. In this work, two type of IPM transforms are used: First, an IPM which maps the image plane to the x-y plane in the world frame (REF) with metric scales. Second, an IPM which maps the image plane to the birds-eye view image (BEV), with pixel scales. Although similar in nature, however these two transforms serve two different purposes. The goal of the first IPM transform is to compute a metric conception of the road in front, for the control purposes, but the goal of second IPM transform is to create BEV images be used for feature extraction and line detection. Within the scope of this work we always refer to the first transform simply as the IMP transform, and the latter as the BEV transform to avoid confusion.
As discussed above, the IPM transform is a function of camera calibration with respect to the ground plane and using a fixed pre-calibrated IPM, for a moving vehicle in a dynamic environment could potentially introduce large amount of error in reconstructing the lines. Figure (number) shows an example of such error. In order to fix this problem, the IPM transform need to be adaptively adjusted to account for changes the camera calibration and in particular, the changes in relative attitude (i.e., roll and pitch) of the camera with respect to ground plane.

4.1. Inverse Perspective Mapping transform

The most straightforward way of finding the IPM transform is by inverting the standard projection matrix P. We define the projection matrix as a transform mapping any 3D homogeneous point $\mathbf{X}$ from the world frame to their corresponding 2D homogeneous position $\mathbf{u}$ in the image plane.

$$\mathbf{u} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \in \mathbb{P}^2, \quad \mathbf{X} = \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} \in \mathbb{P}^3$$

$$\mathbf{u} = P\mathbf{X} \tag{1}$$

In order to find the projection matrix, we first have to define the camera matrix C and view matrices V:

$$C = \underbrace{T(c_x, c_y, c_z)}_{\text{translate}} \underbrace{R_z(\psi)}_{\text{yaw}} \underbrace{R_y(\theta)}_{\text{pitch}} \underbrace{R_x(\phi)}_{\text{roll}} \tag{2}$$

$$V = C^{-1} = R_x(-\phi)R_y(-\theta)R_z(-\psi)T(-c_x, -c_y, -c_z) \tag{3}$$

The camera matrix is the matrix which rotates and translates the camera form origin to its desired pose in the world frame, in other words, matrix C is the transform converting camera frame to world frame. The view matrix V defines how the world is seen from the view point of the camera, hence, it is the transform converting world frame to camera frame which is the inverse of the camera matrix. Using the 3D homogeneous coordinate

systems, the matrices C and V are expanded as the following:

$$C = \begin{pmatrix} 1 & 1 & 0 & c_x \\ 0 & 1 & 0 & c_y \\ 0 & 0 & 1 & c_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 & 0 \\ \sin(\psi) & \cos(\psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) & 0 \\ 0 & \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

$$V = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) & 0 \\ 0 & \sin(-\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & 0 & \sin(-\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\psi) & \sin(\psi) & 0 & 0 \\ \sin(-\psi) & \cos(\psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(5)

In the standard definitions, by the default the camera is centered on the origin and is looking along the z-axis, however, in this application we need the camera to be looking along the positive direction of x-axis. Therefore, the camera's internal calibration matrix K is defined in a way to account for the change of axis:

$$K = \begin{pmatrix} f_x & 0 & U_0 \\ 0 & f_y & V_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{pmatrix} \tag{6}$$

Standard projection matrix P is a 3×4 matrix that is the product of the camera's intrinsic and extrinsic parameters.

$$P = K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} V \tag{7}$$

Supposing that the ground plane π_G is the $(x-y)$ plane of the world frame having $Z = 0$, the third column of P matrix gets multiplied by zero. Deleting the third column of P, we obtain the 3×3 ground to image projection matrix P_{π_G} which is basically a plane to plane homography, transforming points on the ground plane to their corresponding position in the image plane.

$$\mathbf{u} = \begin{pmatrix} | & | & | & | \\ P_X & P_Y & P_Z & P_W \\ | & | & | & | \end{pmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ W \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \underbrace{\begin{pmatrix} | & | & | \\ P_X & P_Y & P_W \\ | & | & | \end{pmatrix}}_{P_{\pi_G}} \underbrace{\begin{pmatrix} X \\ Y \\ W \end{pmatrix}}_{\mathbf{x}_{\pi_G}} \tag{8}$$

The standard IPM transform H_{ipm} is obtained by inverting the ground to image projection matrix.

$$H_{ipm} = P_{\pi_G}^{-1} \tag{9}$$

4.2. Birds Eye View transform

By definition, the BEV transform H_{bev} directly maps the front image plane, with a pixel coordinate to the BEV image also having a pixel coordinate:

$$\mathbf{u}_{bev} = H_{bev} \mathbf{u}_{front}$$

Thus the first step is to define the parameters of the BEV image:

$$\mathbf{ROI}_w = \begin{pmatrix} ROI_{w_{x_{min}}} \\ ROI_{w_{x_{max}}} \\ ROI_{w_{y_{min}}} \\ ROI_{w_{y_{max}}} \end{pmatrix}, \quad \mathbf{RES} = \begin{pmatrix} RES_{width} \\ RES_{height} \end{pmatrix} \tag{10}$$

The first parameter of the BEV image is called the region of interest $\mathbf{ROI}_w$, describing, in the metric scale, the boundaries of the region on the ground plane that will be visible in the BEV image. The second parameter is the resolution **RES** of the BEV image.

After defining the parameters of the BEV image, through the following equations, the BEV projection matrix P_{bev} is computed which projects the ground plane to the BEV image, accounting for the change of axis as well.

$$\begin{cases} s_x = \dfrac{RES_{width}}{ROI_{w_{y_{max}}} - ROI_{w_{y_{min}}}} \\ s_y = \dfrac{RES_{height}}{ROI_{w_{x_{max}}} - ROI_{w_{x_{min}}}} \\ t_x = s_x \cdot ROI_{w_{y_{max}}} \\ t_y = s_y \cdot ROI_{w_{x_{max}}} \end{cases}$$

$$P_{bev} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{pmatrix} \tag{11}$$

At last, the BEV transofrm H_{bev} is computed as the product of the IPM and the BEV projection matrices:

$$H_{bev} = P_{bev} H_{ipm} \tag{12}$$

5. Line Detection Pipeline

The line detection pipeline that is used in this work is based on an approach proposed by (ref) in work dedicated to this purpose. In this section, only a brief explanation of this method is provided, and thus for a more comprehensive discussion, it is highly recommended to refer to the original work. Briefly, the line detection pipeline is comprised of three steps:
The first step is Feature Extraction to identify the pixels which have the most probability of belonging to road lines
The second step is to unwrap the perspective effect and create the Birds Eye View image and perform a series of morphological operations to further clean the image and create a mask of the features.
The third step is to apply a Window-based Line Following (WLF) algorithm to the feature points in order to select the points belonging to each road line of interest, and produce a small but accurate set of line points, for the right and left lines each.

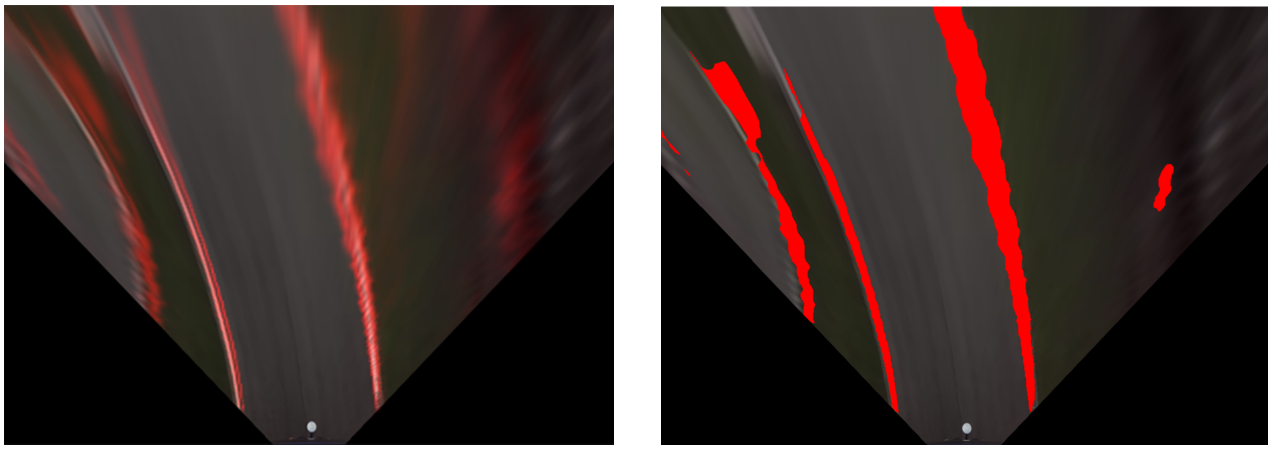
5.1. Feature Extraction

In computer vision, the feature extraction step is to extract and highlight the characteristics of the image most relevant in identifying the desired object or information. Within this application the features are simply the set of pixels belonging to the lines, hence, are called the feature points. In this pipeline, instead of using image filters or other complex feature extraction algorithms used in older computer vision techniques, the Feature Extraction step uses a costume built Convolutional Neural Network (CNN) to perform segmentation on the images and identify the pixels which belong to the road lines.
The used CNN is based on the famous U-Net architecture, with some modifications to optimize it for the real-time application running on low-level hardware with limited computational resources. The modifications include down-scaling the input to 512x256 and reducing the depth of the network to only two, to achieve a high prediction speed, and more explicitly, almost 100 Hz on an NVIDIA Jetson Xavier device which is a well known embedded computing board.
The network is trained using the well-known Berkeley DeepDrive Dataset (REF), widely used in the field of autonomous driving for various tasks such as lane marking detection as well as road objects and derivable area identification. The dataset offers raw driving footage in excess of 1100 hours in various driving scenarios having naturally accruing obstructions with different weather and lighting conditions, making it the ideal dataset to train the desired robust model to work reliably regardless of illumination changes, adverse weather, and clutter in the image.
The output of the network for each pixel in the image, is a value from 0 to 255, indicating the confidence of the network in assigning that pixel to road lines. Hence, the output is a single-channel integer image that we call a Feature Image.

Figure 8: CNN segmentation output laid on top of the input image.

5.2. Birds Eye View Mask

After extracting the feature image, using the transformation number 12 discussed in the previous section, the feature image is converted to a Birds Eye View (BEV) image where the scene is rectified and the shape of the lines is reconstructed.It is important to note that the feature image is not thresholded into a binary mask before converting to BEV, in fact, due to interpolation, having a binary image converted to BEV might create undesired effects in BEV.
Thereafter, the BEV feature image is thresholded and morphologically cleaned to limit the presence of artifacts, and the BEV binary mask is produced.



(a) BEV feature points.

(b) BEV binary mask.

Figure 9: Result of BEV transforming the raw feature points extracted via CNN, which is then post processed to obtain the BEV mask

5.3. Window-based Line Following

The BEV binary mask obtained from the previous stage contains a large number of feature points as well as noise and false detection, however, grouping together these feature points into individual lines and avoiding the noises is a challenging task performed by the WLF.
Considering that the road lines, are meant to guide the driver and visually indicate how the road develops from the current position of the vehicle, the WLF algorithm exploits this fact and searches for the beginning of the lines in the lower end of the BEV mask. After detecting the beginning points of the lines, then it is possible to easily retrace them upwards and make sure to always follow the right marking, independently of its shape. The

Window-based line following (WLF) algorithm, initiates, two windows w_r and w_l at the bottom of the BEV mask, respectively to the left and right side of the vehicle.
Starting from the beginning points of the lines, these widows independently follow the lines along the most relevant feature points, with no restrictions on the shape of the line. At each position visited by a window, an analysis of the enclosed image patch is conducted and the centroid of the largest connected component found is selected. The analysis of the connected components is based on the algorithm proposed by Grana et al. [REF]
For each position of the window, the centroid P_i is saved and then the window is translated upwards in the BEV, while adaptively adjusting its size and orientation, based on the position of the last few feature points, to ensure that the lines are properly followed. Hence, as the feature points evolve along the line and change in shape and orientation, the window also responsively rotates, shrinks, or enlarges in width to match the size of the most relevant connected component found.
If at any point along the line, the window is located on a completely empty patch, its size is temporarily increased and the operation is repeated a few times, in order to find the closest features and recenter However, if after a few iterations none are found, the line is considered lost and the algorithm terminates. Otherwise, the computation ends as the windows touch the upper edge of the BEV image.
The output of the line detection pipeline is the sequence of centroids that are collected along the line $P = \{P_i = (x_i, y_i)\}_{i=0,\dots,n}$ (i.e. the line points), for the right and left lines separately (P_{right} and P_{left}).

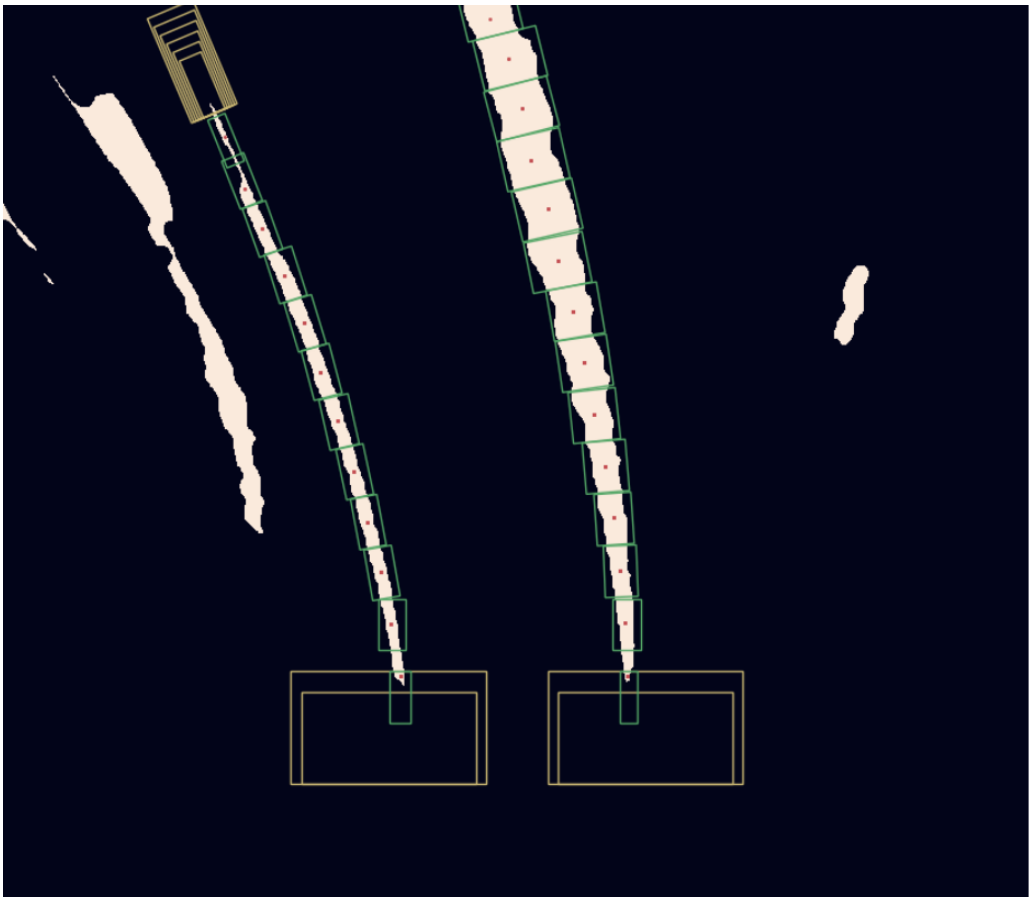


Figure 10: Window based line following.

6. Pitch Estimation

As discussed above, in most other works, pitch estimation is done via enforcing parallelism on straight lane markings, i.e., the road lines, with an inherent limitation that such methods only work correctly when the lines are straight. In this section, a novel approach is proposed to solve the shortcomings of the earlier works, allowing pitch estimation to reliably work on curved road sections, as well as straight sections. This novel technique is based on estimating pitch by enforcing road lines to, momentarily, be part of two concentric circles.
The input of the proposed algorithm is the extracted line points, i.e., the points on the road lines, in the front viewing image. Throughout this manuscript, the line points on the front image will be referred to as the front points, or more explicitly, the front points right or left, for the right and left lines, respectively.

Briefly, for a particular value of pitch angle, an Inverse Perspective Mapping is applied to the front points to compute the corresponding world points. Thereafter, two constant-curvature line models (i.e., circles) are fitted to the collected world points, using which, two sets of comparison points are generated. The comparison points are then used to compute a loss metric, called the parallelism loss, that is minimum when the comparison points and thus the world points fall on two concentric circles. Pitch estimation is done by minimizing the loss metric. (image of the road with concentric circles)

6.1. Comparison points and loss metric

A pair of comparison points $(compts_{r,i}, compts_{l,i})$ are defined as two points on the intersection of a particular radial line $radial_i$ with the inner and outer circles and having a Euclidean distance of d_i (see, Figure 11). Ideally, if the road lines are part two concentric circles, the distances for all pairs of comparison points should be equal, hence, the error $e_i = d_i - d_0$ should be zero. The loss function is defined as the sum of errors for all pairs of comparison points.

$$L_P = \sum_{i=0}^{N} e_i = \sum_{i=0}^{N} |d_i - d_0| \tag{13}$$

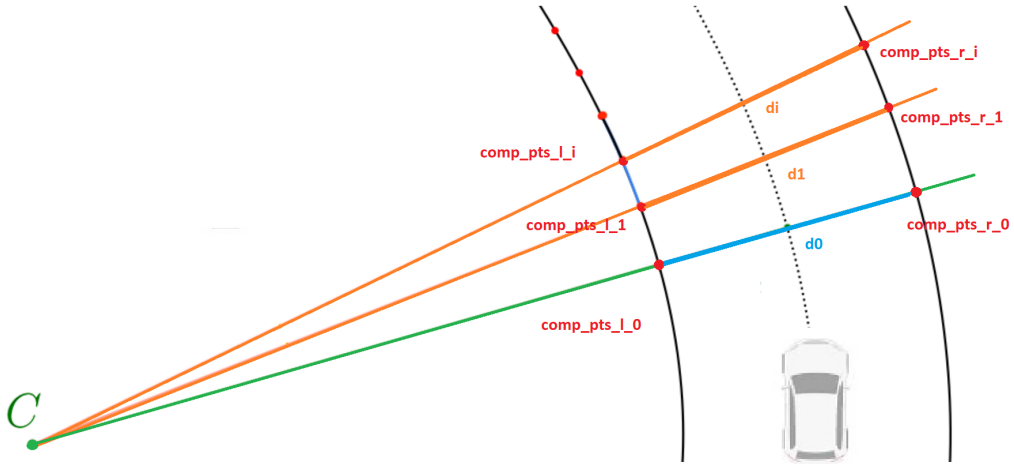


Figure 11: A pair of comparison points are the intersection points of the radial line, from the center of bend, with the road lines.

6.2. Why comparison points

Although in theory, it is possible to fit Cartesian circle equations to the world points and calculate the loss metrics using these fitted circles directly (i.e., with no need for comparison points), in practice, due to noise and error in the line detection it proved to be very difficult and unreliable, in particular, when the road lines are straight or with a slight curvature. Fitting a circle to straight-line results in a very large diameter circle, and considering the noise in the line detection, the fitting becomes unreliable and pitch estimations using these fittings would result in an incorrect estimate [FIGURE].
The key to overcoming this problem is to use the fitting only in the vicinity of the vehicle, within the visible range, i.e., in a range in which the world points are visible. Furthermore, the exact Cartesian equations of the circles is not needed, merely estimating the curvature value of the lines, within the visible range, would suffice. In the proposed method, instead of fitting circles in the Cartesian space, the fittings are done in the Curvilinear space and comparison points are generated within the visible range.

6.3. Curvilinear framework

In the Curvilinear framework $(s - \vartheta)$ lines are represented via two parameters S and ϑ as well as an origin point O. In this representation, the tangent angle to the line ϑ at any point s along the line, starting from the origin, is represented as a function of s $(\vartheta(s))$. The origin O is the starting point of the line represented via its Cartesian coordinates $O = (x_O, y_O)$ in the world frame.

In order to, fit a constant curvature line model to a set of line points $P = \{(x_i, y_i)\}_{i=0,\dots,n}$, the initial step is to set the first point $P_0 = (x_0, y_0)$ as the origin of the line. Then between each point P_i and its successor P_{i+1} the Euclidean distance Δs_i and the tangent angle of the connecting segment ϑ_i with respect to the X-axis, are calculated.

$$\Delta s_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{14}$$

$$\vartheta_i = \arctan2\,(y_{i+1} - y_i, x_{i+1} - x_i) \tag{15}$$

For small values of Δs_i the following holds:

$$s_i = \int ds \approx \sum_{k=0}^{i} \Delta s_k \tag{16}$$

Equations 15 and 16 will produce a set of points $S_{(s,\vartheta)} = \{(s_i, \vartheta_i)\}_{i=0,\dots,n}$ in $(s - \vartheta)$ space which will be used to fit the model $\vartheta(s)$.
Figure number 12 and 13 illustrate an instance of the extracted world points, which are then converted to Curvilinear space and a constant-curvature line fitting is performed.

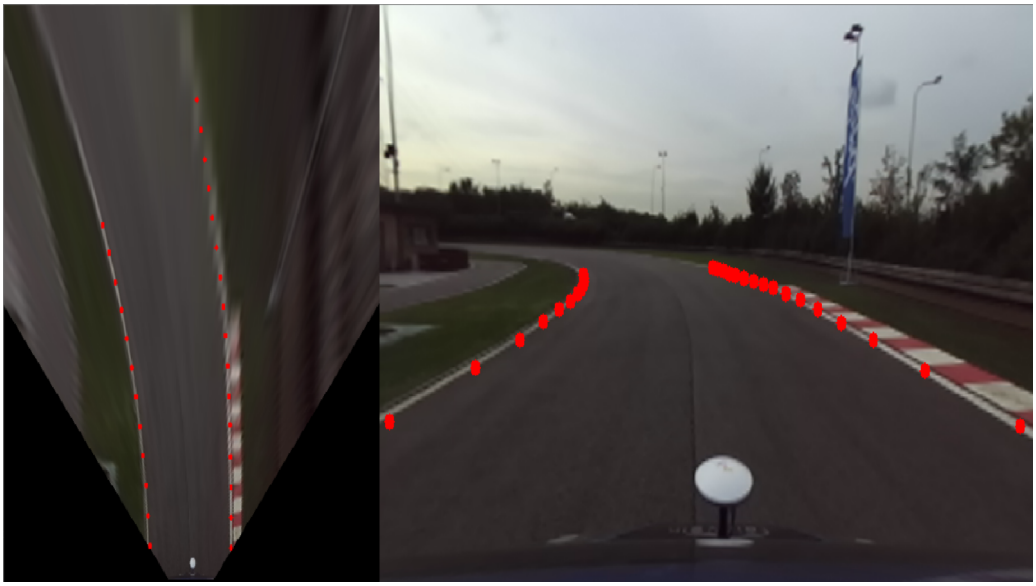


Figure 12: Final output of the line detection pipeline.

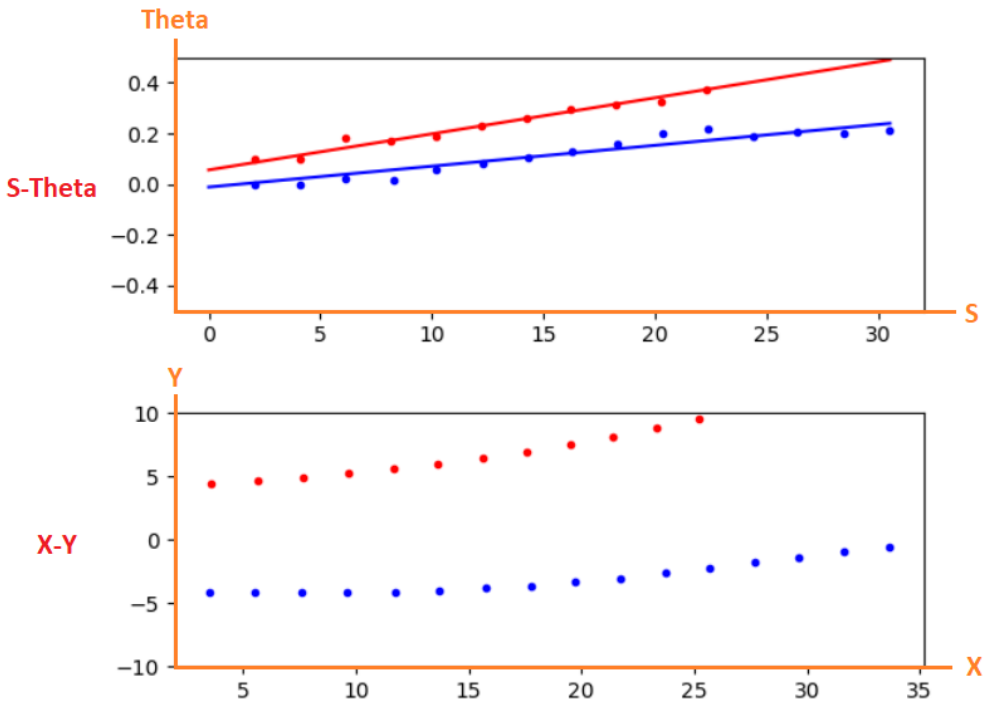


Figure 13: S-Theta fitting of world points. The lower subplot shows the world points in x-y space and the upper subplot is the S-Theta conversion and fitting of the same world points, shown in red and blue colours, for left and right lines, respectively.

6.4. $(s - \vartheta)$ model

Within the Curvilinear space the derivative of the model $\frac{d(\vartheta(s))}{ds}$ is the curvature value of the line in Cartesian space. Since both the straight lines and the circular lines have a constant curvature value, a linear relationship between the $s$and ϑ in the form of $\vartheta(s) = as + b$ could effectively model straight and curved road section. Therefore after calculating $S_{(s,\vartheta)}$ set of points via equations 16 and 15, a first-order polynomial (i.e., a line) is fitted to the points. In this model, a is the curvature value and b is the initial tangent angle at the origin.

6.5. Conversion to Cartesian

The $(s - \vartheta)$ model could be converted back to Cartesian space $(X_\vartheta(s) - Y_\vartheta(s))$ via following equations:

$$\begin{aligned} X_\vartheta(S) &= \int_0^S \cos(\vartheta(s))ds + x_O \\ X_\vartheta(S) &= \int_0^S \cos(as+b)ds + x_O \\ X_\vartheta(S) &= \frac{\cos(c)}{a}\sin(aS) + \frac{\sin(c)}{a}(\cos(aS) - 1) + x_O \end{aligned} \tag{17}$$

$$\begin{aligned} Y_\vartheta(S) &= \int_0^S \sin(\vartheta(s))ds + y_O \\ Y_\vartheta(S) &= \int_0^S \sin(as+b)ds + y_O \\ Y_\vartheta(S) &= \frac{-\cos(c)}{a}(\cos(aS) - 1) + \frac{\sin(c)}{a}\sin(aS) + y_O \end{aligned} \tag{18}$$

6.6. Curvature ratio

While generating the comparison points, the curvature ratio between the lines needs to be considered, to make pairs of comparison points fall on the same radial line. In the ideal scenario, the curvature ratio could simply be computed by dividing curvature values extracted from S-Theta fittings $C_{ratio} = \frac{a_{inner}}{a_{outer}}$ where a_{outer} and a_{inner} are the extracted curvature value for the outer and inner lines, respectively. However, since the extracted curvature values are affected by the noise, in practice, computing curvature ratio by a simple division does not work, especially in the case of straight lines where a_{outer} and a_{inner} are close to zero, dividing $C_{ratio} = \frac{0}{0}$ could be problematic. To solve this problem the curvature ratio is estimated considering the average curvature of both lines through following equations:

$$d_0 = |P_{r,0} - P_{l,0}| = \sqrt{(x_{r,0} - x_{l,0})^2 + (y_{r,0} - y_{l,0})^2} \tag{19}$$

$$R_{inner} \approx \frac{a_{inner}^{-1} + a_{outer}^{-1} - d_0}{2} \tag{20}$$

$$R_{outer} \approx R_{inner} + d_0 \tag{21}$$

$$C_{ratio} = \begin{cases} \frac{R_{outer}}{R_{inner}}, & \text{if } a_{outer} \times a_{inner} > 0 \\ 1, & \text{otherwise} \end{cases} \tag{22}$$

R_{outer} and R_{inner}, are extracted radius of the lines and d_0 is the Euclidean distance between the first points of the left and right lines. It is important to note that the ratio is defined to always be larger than one. Moreover, if the extracted curvature values for the lines do not agree, i.e., one is positive the other negative, then C_{ratio} is set to 1.

6.7. Making comparison points

To make comparison points, the initial step is to determine the position of the first pair, defined by two variables $S_{outer,0}$ and $S_{inner,0}$, for the outer and inner lines, respectively.
The starting point of comparison points is an influential parameter, on a condition that the vehicle is aligned with the center line, i.e., the angle between X-axis and the centerline is zero, the starting point should be at x=0. If the vehicle is not aligned with the centerline, having relative yaw of ψ degrees, w.r.t. centerline, a rotation needs to be applied to the first pair of world points, to account for the misalignment. The following equations provides the starting points of comparison points.

$$P'_{inner,0} = R_\psi P_{inner,0} \tag{23}$$

$$P'_{outer,0} = R_\psi P_{outer,0} \tag{24}$$

$$S_{inner,0} \approx -x'_{inner,0} \tag{25}$$

$$S_{outer,0} \approx -x'_{outer,0} \tag{26}$$

In the above equations, R_ψ is a $2D$ rotation matrix applied only to the first pair of world points. Obviously, it is also possible to apply R_ψ to all world points prior to fitting the S-Theta model, as shown in Algorithm 1.
An in-depth discussion about estimating the centerline estimation and relative yaw is provided in (advanced centerline estimation), however, most of the time the value of relative yaw is so small that the rotation step could simply be ignored.
Thereafter, the i^{th} pair of comparison points are computed using the following equations:

$$S_{min} = \min(S_{max_{outer}}, S_{max_{inner}}) \tag{27}$$

$$step_{small} = \frac{S_{min}}{N} \tag{28}$$

$$step_{large} = C_{ratio} \times step_{small} \tag{29}$$

$$S_{outer,i} = S_{outer,0} + i \times step_{large} \tag{30}$$

$$S_{inner,i} = S_{inner,0} + i \times step_{small} \tag{31}$$

In the above equation, N is the number of comparison points and S_{min} is the minimum of the maximum visible length, for both lines.
Outer and inner lines are determined depending on the sign of their curvature. If both lines have positive curvature values, then the right line is the outer one, and left line is the inner one, and vice versa. After computing the positions of all the comparison points in the S-Theta space, using equations 17 and 18, their Cartesian positions are calculated.

6.8. Pre-processing world points

While calculating the S-Theta model of the lines, it is imperative to make sure that the tangents that are computed from longer segments have more weight in the fitting process. Furthermore, since the computation of tangent angle is sensitive to noise, it is recommended to pre-process the world points before fitting the model. Before the fitting phase, we have to make sure that the Euclidean distances between the consecutive points are within the sensible range, from 0.5 meters up to 2 meters, having an equal distribution. If the segments do not have an equal length, then the tangents computed from each segment need to be assigned a weight, in accordance with their length. Figure 14 shows an example of how such short segments with incorrect tangents and no weight could ruin the entire S-Theta and X-Y fittings.
Moreover, performing Moving Average filtering, only on the Y-axis of world points, with a reasonable bandwidth of about 3 meters, is proved to be highly beneficial.

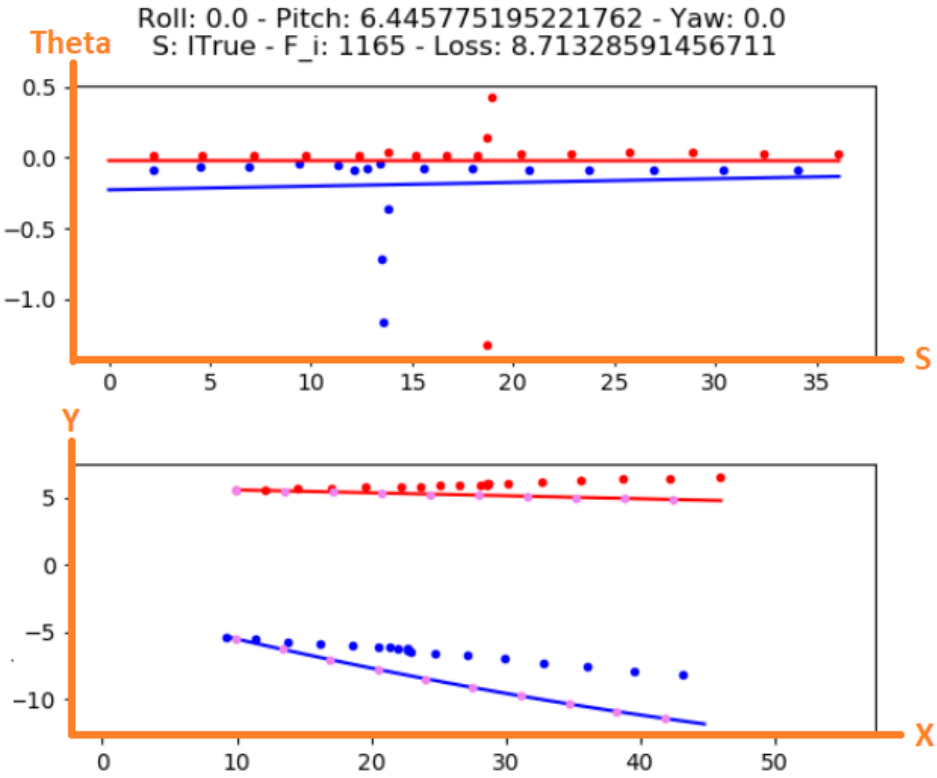


Figure 14: Due to uneven distribution of points, the unreliable and noisy tangents that are computed from smaller segments have the same contribution as the other sections in the fitting process, and thus create large errors.

6.9. Protection

In order to avoid incorrect estimates, it is crucial to set in place a series of protection measures and checks. Failing any of these protection checks results in a positive infinity loss value, effectively discarding the estimate. The first check, is performed on the number of detected world points. On each one of lines, roughly a minimum of seven meters needs to be visible, to proceed with the frame.
The second check is called the safe range check. Considering the fact that the maximum sensible visible range of the lines, considering the quality of camera and line detection pipeline, is a known parameter, if after applying IPM to the front points, the computed world points do not fall with in the safe range, the IPM and its corresponding pitch angle are discarded. For example, if any of the world points end up having a negative X value, i.e., a point behind the vehicle, it is an indication that the IPM is not correct.
The third check is performed on the mean residual of the S-Theta fittings. If for any of the lines, the mean residual of the fitting is above a certain threshold, it suggests that the lines, either truly or due to error in line detection, do not have a constant curvature, therefore it is for the best to not carry on with the estimation.

6.10. Optimization

The optimization is done iteratively, by taking the last estimated pitch angle as the starting point and performing a local search around it to find the pitch with minimum loss. The recommend approach is to set a scan resolution $\Delta p = 0.1$ degree, and perform evaluation for only 3 iterations per frame, the last estimated pitch $pitch_{t-1}$, an step-up $pitch_{t-1} + \Delta p$ and an step-down $pitch_{t-1} - \Delta p$.

$$pitch_t = \text{argmin}(L_P(pitch_{t-1}), L_P(pitch_{t-1} + \Delta p), L_P(pitch_{t-1} - \Delta p)) \tag{32}$$

This approach is especially quick since it is only performed 3 times per frame, and most importantly, it ensures the temporal consistency between consecutive estimates, preventing the estimated pitch from having an abrupt change between two consecutive frames, i.e., within less than 0.033 seconds for a 30 FPS video. Hence, the optimization method also acts as a protection mechanism, protecting the estimate from gross errors that could happen in single frames due to line detection errors.

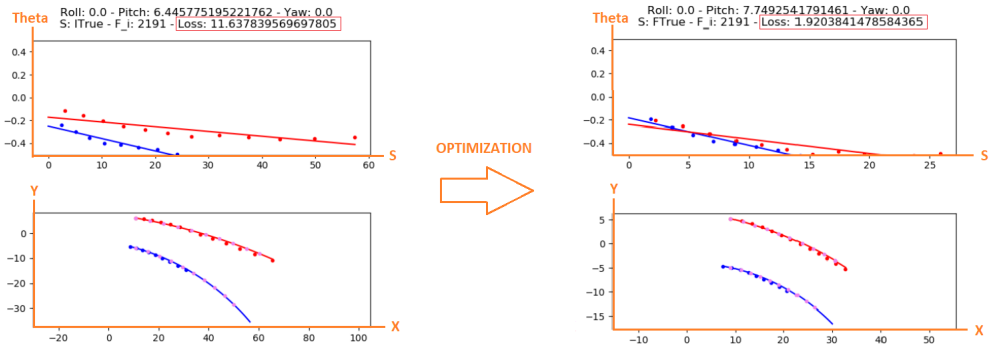


Figure 15: Using the comparison points, shown in pink, a loss metric is calculated and optimized resulting in more parallel road line.

6.11. Algorithm

The pseudo-code of the proposed pitch estimation method is presented by Algorithm 1 below.

Algorithm 1 Pitch Estimation

if $length(front\_pts\_r) < min\_visible\_length$ **or** $length(front\_pts\_l) < min\_visible\_length$ **then**
  min_loss = inf
  pitch(t) = pitch(t-1)
  return
end if
scan_array(0) = pitch(t-1)
scan_array(1) = pitch(t-1)+delta_pitch
scan_array(2) = pitch(t-1)-delta_pitch
for *temp_pitch* **in** *scan_array* **do**
  world_pts_r = project_image_2_world(front_pts_r,roll(t-1),temp_pitch)
  world_pts_l = project_image_2_world(front_pts_l,roll(t-1),temp_pitch)
  world_pts_r = pre_process(world_pts_r)
  world_pts_l = pre_process(world_pts_l)
  if $is\_within\_safe\_range(world\_pts\_r)$ **and** $is\_within\_safe\_range(world\_pts\_l)$ **then**
    min_loss = inf
    pitch(t) = pitch(t-1)
    return
  end if
  world_pts_r = rotate_2D_points(world_pts_r, relative_yaw)
  world_pts_l = rotate_2D_points(world_pts_l, relative_yaw)
  s_theta_model_r = fit_s_theta_model(world_pts_r)
  s_theta_model_l = fit_s_theta_model(world_pts_l)
  if $s\_theta\_model\_r.mean\_residual > max\_mean\_res$ **or** $s\_theta\_model\_l.mean\_residual > max\_mean\_res$ **then**
    min_loss = inf
    pitch(t) = pitch(t-1)
    return
  end if
  comp_pts_in_s_theta = make_pitch_comp_pts_in_s_theta(s_theta_model_r,s_theta_model_l,N)
  comp_pts_r = convert_s_theta_2_xy(s_theta_model_r,comp_pts_in_s_theta.r)
  comp_pts_l = convert_s_theta_2_xy(s_theta_model_l,comp_pts_in_s_theta.l)
  loss_array(temp_pitch) = compute_parallelism_loss(comp_pts_r,comp_pts_l)
end for
min_loss = min(loss_array)
pitch(t) = argmin(loss_array)
return

7. Roll Estimation

Estimating roll angle in a monovision setup, using the road lane markings, is a subject that is very little discussed in the literature. Compared to pitch angle which has a more pronounced effect on the parallelism of the lines even for small variations, for the roll angle an error of roughly 1 degree has a very subtle effect on the lines and is not easily detectable.

Most researches in IPM stabilization or camera calibration, often ignore roll angle setting it to zero. Presumably, the main reason for ignoring roll angle is the inherent difficulty, if not the impossibility, of extracting roll angle from road lane marking alone.

In this chapter, a new approach is proposed, which attempts to estimate roll angle with a good degree of precision, albeit, by placing a more strict assumption than what was in place for pitch estimation.

As discussed in the section 6, pitch estimation is done by enforcing road lines to be part of two concentric circles, however, for roll estimation, it is not enough and there needs to be some form of other information. In the proposed method not only the road lines need to be part of two concentric circles, but also, the visible length

for left and right lines needs to be approximately related via the curvature ratio $S_{max_{outer}} \approx C_{ratio} \times S_{max_{inner}}$. It is essential to note that, the above assumption on the relation between the visible length of the lines, may seem too much strict for real-life scenarios, as the full length of lines could be obscured or not detected for many reasons, however without it, it is impossible to extract roll angle from road lines. Nevertheless, the estimated roll from vision, coupled with proper checks and filtering, for example using inertial methods, proved to be useful in increasing the accuracy of IPM in real-life trials.
Furthermore, for any correct IPM the resulting world points should also fall within the safe range, not having negative X values or end up very distant from the vehicle.
Taking advantage of all the above assumptions, in a method very similar to the one proposed for pitch estimation, a set of comparison points are created and projected back into the front image plane, becoming the front comparison points. The actual detected front points and the generated front comparison points, constitute four sets of points representing the front lines and the front comparison lines for left and right lines, respectively. The loss metric is computed, by comparing the front lines against the front comparison lines, for the left and right lines independently.

7.1. Front comparison points

Front comparison points $I_{comp} = \{(u_i, v_i)\}_{i=0,\dots,n}$ are the image of world comparison points, which are created by a method similar to section 6.1. Briefly, for a certain roll angle value, by applying the IPM to front points, the world points are computed. If there is a significant relative yaw, w.r.t. the centerline, similar to the section 6.7 a $2D$ rotation is applied to align the X-axis with the centerline. Then, using the equations 15 and 16 the $(s - \vartheta)$ model is fitted and the curvature ratio computed. World comparison points, are created in the $(s - \vartheta)$ space using the following equations and are converted back to Cartesian space via equations 17 and 18.

$$S_{max} = \max(S_{max_{outer}}, S_{max_{inner}}) \tag{33}$$

$$X_{start} = \frac{|x_{outer,0}| + |x_{inner,0}|}{2} \tag{34}$$

$$S_{inner,0} \approx X_{start} - x_{inner,0} \tag{35}$$

$$S_{outer,0} \approx X_{start} - x_{outer,0} \tag{36}$$

$$step_{large} = \frac{S_{max}}{N} \tag{37}$$

$$step_{small} = \frac{step_{large}}{C_{ratio}} \tag{38}$$

$$S_{outer,i} = S_{outer,0} + i \times step_{large} \tag{39}$$

$$S_{inner,i} = S_{inner,0} + i \times step_{small} \tag{40}$$

Using projection matrix (number), the front comparison points I_{comp} are created by projecting world comparison points back to the front image plane.

7.2. Partial Ellipse

Presuming that the road lines are part of two concentric circles, having a constant curvature, the image of the lines in the front plane could be part of an ellipse, having a non-constant curvature, with points of minimum and maximum curvature. In the Curvilinear space, using a second-order polynomial model would allow the curvature of the lines to linearly increase or decrease, enabling the S-Theta model to effectively be fitted to curved lines that partially fall on an ellipse. A partial ellipse is defined as a second-order polynomial S-Theta model, fitted to a set of line points with a limited length, defined by a dedicated variable S_{pe}. The $(s - \vartheta)$ fitting is only valid within the range visible length of the line, from zero to S_{pe}. Prior to fitting the partial ellipse, it is necessary to rotate the front point by 90 degrees, essentially, flipping the $(X - Y)$ axis of the image plane.

$$I'_{comp} = flipXY(I_{comp}) = \{(v_i, u_i)\}_{i=0,\dots,n} \tag{41}$$

After flipping the axis, using equations 15 and 16 a set of S-Theta points $S_{(s,\vartheta)} = \{(s_i, \vartheta_i)\}_{i=0,\dots,n}$, to which a second-order polynomial fitted $\vartheta(s) = as^2 + bs + c$. The partial ellipse is defined by following equations:

$$\begin{cases} \vartheta_{pe}(s) = as^2 + bs + c \\ S_{pe} = \max(\{s_i\}_{i=0,\dots,n} \subset S_{(s,\vartheta)}) \end{cases} \tag{42}$$

7.3. Comparing Partial Ellipses

The comparison of two partial ellipses against each other is done by differencing the S-Theta polynomials and integrating the absolute difference in the range $[0 : S_{pe,max}]$, where $S_{pe,max}$ is the maximum visible length of Partial Ellipses.

$$S_{pe_{max}} = \max(S_{pe_1}, S_{pe_2}) \tag{43}$$

$$D_{pe}(pe1, pe2) = \int_0^{S_{pe_{max}}} |\vartheta_{pe_1}(s) - \vartheta_{pe_2}(s)| ds \tag{44}$$

7.4. Loss metric

In order to compute the loss metric, for each set of line points in the front image plane, a Partial Ellipse is fitted. For the left and right lines, separately, the Partial Ellipses of comparison points are compared against the Partial Ellipses of front points, and two independent difference values D(pe1,pe2) are computed. The loss is the sum of Partial Ellipse differences for right and left lines. Of course, comparing identical partial ellipses will yield a zero difference value, therefore, in the ideal case with a correct roll angle value, the total loss should be zero.

$$L_R = D_{pe}(PE_{original_{right}}, PE_{comp_{right}}) + D_{pe}(PE_{original_{left}}, PE_{comp_{left}}) \tag{45}$$

Where $PE_{original_{right}}$ and $PE_{original_{left}}$ are the Partial Ellipses of original front points, and $PE_{comp_{right}}$ and $PE_{comp_{left}}$ are for the front comparison points.

7.5. Protection

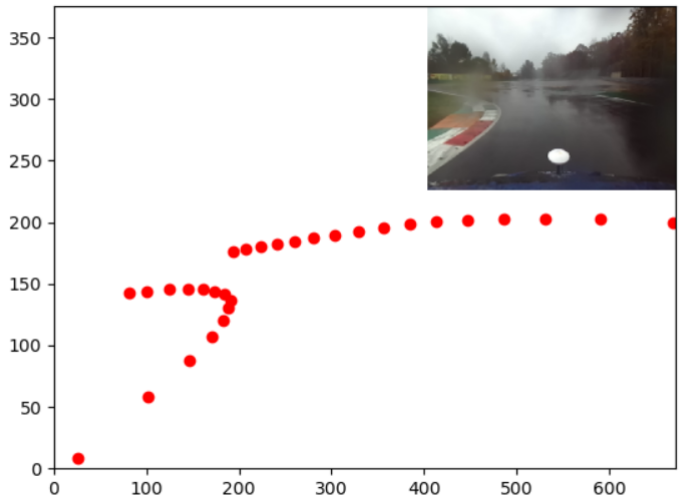
Checking the visible length of lines and checking the mean residual of the S-Theta fitting is performed in a similar manner to section 6.9. The only difference is that the checking the safe range for world points is not performed explicitly, as it is embedded within the roll estimation procedure itself, i.e., the world comparison points are always made to fall within the safe range, now if the original world points are not in the safe range to some extent, comparing the resulted Partial Ellipses will indicate an incorrect roll, in a continuous space rather than a conditional check.

7.6. Optimization

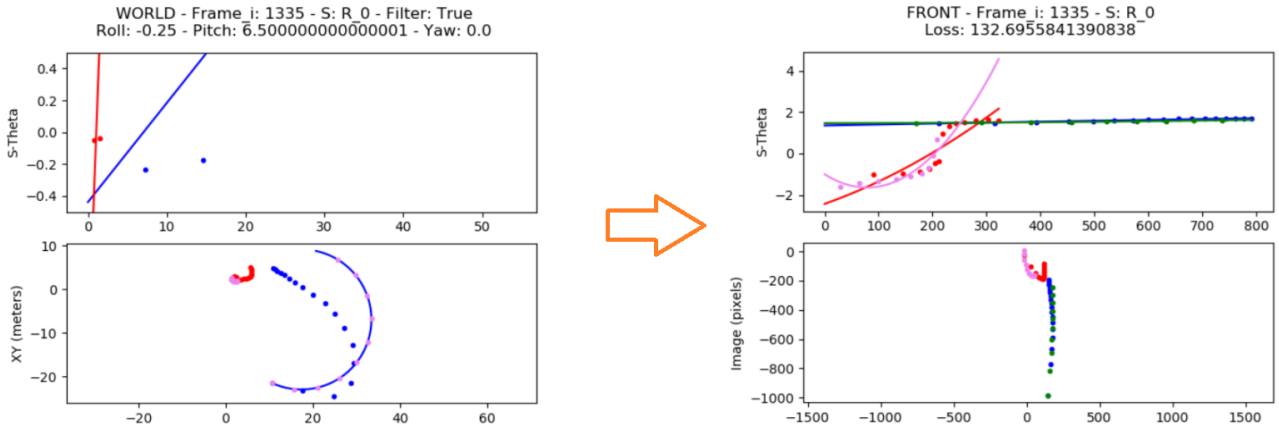
Similarly, optimization is done by performing a 3-step local search around the last estimated roll angle, ensuring fast time performance and temporal consistency.

$$roll_t = \text{argmin}(L_R(roll_{t-1}), L_R(roll_{t-1} + \Delta r), L_R(roll_{t-1} - \Delta r)) \tag{46}$$

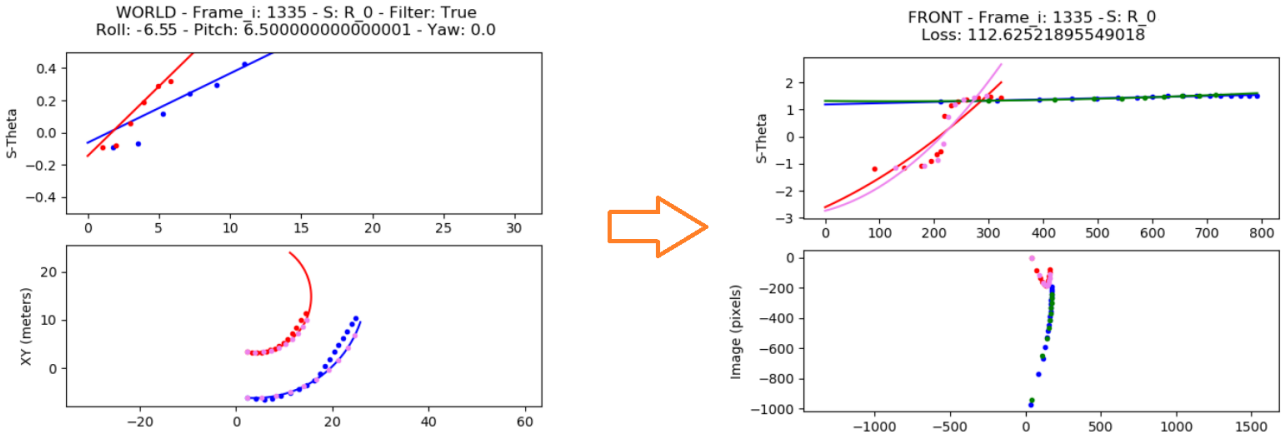
.



(a) A rendered frame using the virtual camera with, see section 9.1, with a ground truth roll of -7 degrees.



(b) Due to incorrect roll, the front comparison points, in particular for left line, are separated from the actual detected front points and comparing the corresponding partial ellipses yields a large loss value.



(c) The same line points from previous figure, projected using an IPM with more accurate roll value, resulting in similar partial ellipses and smaller loss value.

Figure 16: Making front comparison points from world comparison points, fitting the partial ellipses, and computing the loss metric. Shown on the left sides, are the S-Theta fitting and the world comparison points and on the right side the corresponding front comparison points and the fitted partial ellipses. Actual detected line points for left and right lines are plotted in red and blue respectively. The front comparison points are plotted in pink and green for the left and right lines, respectively.

7.7. Algorithm

Algorithm 2 Roll Estimation

if $length(front\_pts\_r) < min\_visible\_length$ **or** $length(front\_pts\_l) < min\_visible\_length$ **then**
  min_loss = inf
  roll(t) = roll(t-1)
  return
end if
scan_array(0) = roll(t-1)
scan_array(1) = roll(t-1)+delta_roll
scan_array(2) = roll(t-1)-delta_roll
for *temp_roll* **in** *scan_array* **do**
  world_pts_r = project_image_2_world(front_pts_r,temp_roll,pitch(t))
  world_pts_l = project_image_2_world(front_pts_l,temp_roll,pitch(t))
  world_pts_r = pre_process(world_pts_r)
  world_pts_l = pre_process(world_pts_l)
  world_pts_r = rotate_2D_points(world_pts_r, relative_yaw)
  world_pts_l = rotate_2D_points(world_pts_l, relative_yaw)
  s_theta_model_r = fit_s_theta_model(world_pts_r)
  s_theta_model_l = fit_s_theta_model(world_pts_l)
  if $s\_theta\_model\_r.mean\_residual > max\_mean\_res$ **or** $s\_theta\_model\_l.mean\_residual > max\_mean\_res$ **then**
    min_loss = inf
    roll(t) = roll(t-1)
    return
  end if
  comp_pts_in_s_theta = make_roll_comp_pts_in_s_theta(s_theta_model_r,s_theta_model_l,N)
  world_comp_pts_r = convert_s_theta_2_xy(s_theta_model_r,comp_pts_in_s_theta.r)
  world_comp_pts_l = convert_s_theta_2_xy(s_theta_model_l,comp_pts_in_s_theta.l)
  front_comp_pts_r = project_world_2_image(world_comp_pts_r,temp_roll,pitch(t))
  front_comp_pts_l = project_world_2_image(world_comp_pts_l,temp_roll,pitch(t))
  front_comp_pts_r_flipped = flip_axis(front_comp_pts_r)
  front_comp_pts_l_flipped = flip_axis(front_comp_pts_l)
  front_pts_r_flipped = flip_axis(front_pts_r)
  front_pts_l_flipped = flip_axis(front_pts_l)
  pe_original_r = make_partial_ellipse(front_pts_r_flipped)
  pe_original_l = make_partial_ellipse(front_pts_l_flipped)
  pe_comp_r = make_partial_ellipse(front_comp_pts_r_flipped)
  pe_comp_l = make_partial_ellipse(front_comp_pts_l_flipped)
  diff_pe_r = compare_partial_ellipses(pe_original_r,pe_comp_r)
  diff_pe_l = compare_partial_ellipses(pe_original_l,pe_comp_l)
  loss_array(temp_roll) = diff_pe_r+diff_pe_l
end for
min_loss = min(loss_array)
roll(t) = argmin(loss_array)
return

8. Filtering

Despite all the protection measures and various filtering's already in place within the proposed pitch and roll estimation methods, the raw estimates from vision are still affected by the error in the line detection pipeline and have a considerable incorrect oscillation (figure). Therefore, it is necessary to set in place an external filtering mechanism to further smoothen the estimates as well as to prevent gross estimation errors from single frames to extensively impact the overall estimate.

8.1. Moving Average

The first and most basic filter is the Moving Average (MA), it has been observed that having a MA would hugely improve the performance of the system, albeit, by introducing a lagging effect which could potentially hurt the system responsiveness if the MA band is too wide. A MA band equal to half a second, proved to be providing a good compromise between the responsiveness and the stability of the system, for example if the camera feed frequency is 30 FPS then the recommend MA band would be 15 frames.

8.2. Kalman Filter

In order to further improve the performance and reliability of system, a more advanced Kalman Filter (KF) based approach is proposed in this chapter. Instead of solely relying on the vision-based estimate and a MA, by taking advantage of inertial measurements provided by a MEMS IMU and fusing the IMU data and estimates from vision through a simple KF, it is possible to provide a far more robust and reliable estimation.

8.3. Inertial Model

The ideal inertial model, would be the model of the suspension dynamics of the vehicle, taking the acceleration measurements as input, outputting the roll and pitch (dive and squat) angles for the vehicle.
In order to find a reasonably simple dynamical model, various methods where tested, with limited success.
For instance, attempting to simplify the pitch and roll dynamics, independently, as an inverted pendulum model with springs and dampers, proved to be not working as intended, especially on curved road sections where the test vehicle, having a very short wheel base, experiences rotational motion analogous to a conical pendulum, proving the roll and pitch dynamics need to be modeled jointly.
On the other hand, having a more accurate dynamical model with four springs and dampers and taking into account the suspension mechanics of the wheels, proved to be cumbersome, adding too many parameters to the system, which need to be meticulously calibrated for every vehicle and weight loadings.
The idea of adding vehicle's parameters into the KF states, was also tried, hoping they would converge on-the-fly to the real vehicle parameters, however, this approach also did not produce good results.

8.4. Vehicle-Free Inertial Model

Favoring a vehicle-free inertial model, the filter proposed in this chapter, is based on integrating the rotation rate. It is imperative to note that the attitude measurement from vision, is measured w.r.t. the ground plane, however, the rate measurements from the IMU are w.r.t. to the inertial frame. Nevertheless, the attitude computed from these two approaches could be comparable, if the ground plane does not change, which is often the case whitin short time intervals, allowing the two methods to be successfully fused in a KF.
In the proposed filter, the attitude is represented via a unitary Quaternion variable q, and the time derivative $\dot{q}$ is defined as:

$$\dot{q}(t) = \lim_{\Delta t \to 0} \frac{q(t + \Delta t) - q(t)}{\Delta t} \tag{47}$$

The golden idea behind this filter design is that for any change in the attitude to be valid, it needs to be measured by both of the methods, making the $\dot{q}$ the intended value to be filtered, and thus the KF state vector is comprised of both the q and $\dot{q}$. Following are the state space model of the proposed KF:

$$\begin{cases} X(t+1) = FX(t) + \omega_t \\ Z(t) = X(t) + \upsilon_t \\ \omega_t \sim \mathcal{N}(0, Q) \\ \upsilon_t \sim \mathcal{N}(0, R) \end{cases} \tag{48}$$

$$X = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \end{pmatrix} \tag{49}$$

$$F = \begin{pmatrix} 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{50}$$

$$Z = \begin{pmatrix} q_{m_1} \\ q_{m_2} \\ q_{m_3} \\ q_{m_4} \\ \dot{q_{m_1}} \\ \dot{q_{m_2}} \\ \dot{q_{m_3}} \\ \dot{q_{m_4}} \end{pmatrix} \tag{51}$$

The measurement matrix Z is comprised of both q_m and $\dot{q_m}$. q_m is the direct measurement, of relative attitude of the vehicle's Center of Gravity (CG) w.r.t. the ground plane, extracted using vision through the following equations:

$$\begin{cases} \phi = pitch(t) - pitch^{CG}_{camera} \\ \theta = roll(t) - roll^{CG}_{camera} \\ \psi = 0 \end{cases} \tag{52}$$

In the above equations $roll(t)$ and $pitch(t)$ are the extracted attitude of camera w.r.t. the ground plane from the vision. Converting these measurements from camera to CG, ϕ and θ are the estimated roll and pitch from the vision for CG of the vehicle, w.r.t. the ground plane. ψ is the yaw of the vehicle w.r.t. the ground plane and is always set to zero $\psi = 0$ in this application. q_m is computed by converting above attitude from Euler Angles representation to the Quaternion representation using following equations:

$$\begin{cases} q_{m_1} = \cos(\frac{\psi}{2})\cos(\frac{\theta}{2})\cos(\frac{\phi}{2}) + \sin(\frac{\psi}{2})\sin(\frac{\theta}{2})\sin(\frac{\phi}{2}) \\ q_{m_2} = \cos(\frac{\psi}{2})\cos(\frac{\theta}{2})\sin(\frac{\phi}{2}) - \sin(\frac{\psi}{2})\sin(\frac{\theta}{2})\cos(\frac{\phi}{2}) \\ q_{m_3} = \cos(\frac{\psi}{2})\sin(\frac{\theta}{2})\cos(\frac{\phi}{2}) + \sin(\frac{\psi}{2})\cos(\frac{\theta}{2})\sin(\frac{\phi}{2}) \\ q_{m_4} = \sin(\frac{\psi}{2})\cos(\frac{\theta}{2})\cos(\frac{\phi}{2}) - \cos(\frac{\psi}{2})\sin(\frac{\theta}{2})\sin(\frac{\phi}{2}) \end{cases} \tag{53}$$

$\dot{q_m}$ is the rotation rate measurement from a MEMS 3D gyroscope. As discussed above, within a short time intervals $\dot{q} \sim \dot{q_m}$ holds. Allowing $\dot{q_m}$ to be used as a direct measurement for $\dot{q}$ in the short term, and in the long term, $\dot{q}$ will be corrected by vision. To obtain $\dot{q_m}$ from rate $\omega = [\omega_x, \omega_y, \omega_z]^T$ measured by the IMU, first we define Δq which is the instantaneous small rotation of vehicle's body frame w.r.t. inertial frame, within a very short time interval Δt, represented in the instantaneous body frame. Then $q(t + \Delta t)$ is obtained using equation 55.

$$\Delta \mathbf{q} = \cos\frac{\|\boldsymbol{\omega}\|\Delta t}{2} + \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}\sin\frac{\|\boldsymbol{\omega}\|\Delta t}{2} \tag{54}$$

$$q(t + \Delta t) = q(t)\Delta q \tag{55}$$

By plugging above equations into equation 47 and propagating through, $\dot{q_m}$ is derived:

$$\dot{q_m} = \begin{pmatrix} \dot{q_{m_1}} \\ \dot{q_{m_2}} \\ \dot{q_{m_3}} \\ \dot{q_{m_4}} \end{pmatrix} = 0.5 \begin{pmatrix} q_1 & -q_2 & -q_3 & -q_4 \\ q_2 & q_1 & -q_4 & q_3 \\ q_3 & q_4 & q_1 & -q_2 \\ q_4 & -q_3 & q_2 & q_1 \end{pmatrix} \begin{pmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{56}$$

In the proposed filter, the value of $\dot{q}$ is decided by measurements from vision and gyro as well as its previous values, by properly tuning the process noise Q and measurement noise R considering the accuracy of IMU and reliability of the line detection pipeline, it is possible to obtain very good filtering results.

8.5. Gross Error Detection

Using KF it is possible to perform a gross error detection to increase the robustness of the system. Ideally, the error between the measurement and prediction should obey a normal distribution with a zero mean value $Z_k - H\hat{X}_{k|k-1}H^T \sim \mathcal{N}(0, HP_{\hat{X}_{k|k-1}}H^T + R_k)$.
According to (ref) the anomaly in the $\Delta\chi_k^2$ is a good indication of a gross error not belonging to the distribution. Considering the observation model $H = I_{8\times 8}$:

$$\Delta\chi_k^2 = (Z_k - \hat{X}_{k|k-1})^T (P_{\hat{X}_{k|k-1}} + R_k)^{-1} (Z_k - \hat{X}_{k|k-1}) \tag{57}$$

Where, $(Z_k - \hat{X}_{k|k-1})$ is the error between prediction and measurement. $P_{\hat{X}_{k|k-1}}$ and R are the predicted estimate covariance and observation noise covariance, respectively. Figure (number) shows an example of anomaly in the $\Delta\chi_k^2$, in which case the particular frame is skipped and the KF state and the estimate covariance P are restored, to remove the effect of gross error.

9. Evaluation

Figure 17 shows an example of the BEV image and the extracted world points using the original IPM and adaptive IPM. Although, simply by looking at the BEV images it becomes obvious that the adaptive IPM produces much more parallel road lines, nevertheless, for scientific reasons we need to perform a comprehensive evaluation of the proposed methods. The evaluation of the proposed attitude estimation pipeline is done using our dataset (see section 2.2), via two different methods:
The first evaluation method, feeds the proposed attitude estimation pipeline with a set of rendered images, from the viewpoint of a virtual camera, following the same trajectory as the vehicle and with an adjustable oscillating roll and pitch angles, to see whether the estimated attitude correctly follows the ground truth oscillation of the virtual camera.
The second evaluation method works by comparing the extracted world points, from the images of the vehicle's front view camera, against the ground truth of the world points, provided by the dataset, and computes a set of metrics to evaluate whether the adaptive camera calibration improves the accuracy of inverse perspective mapping.

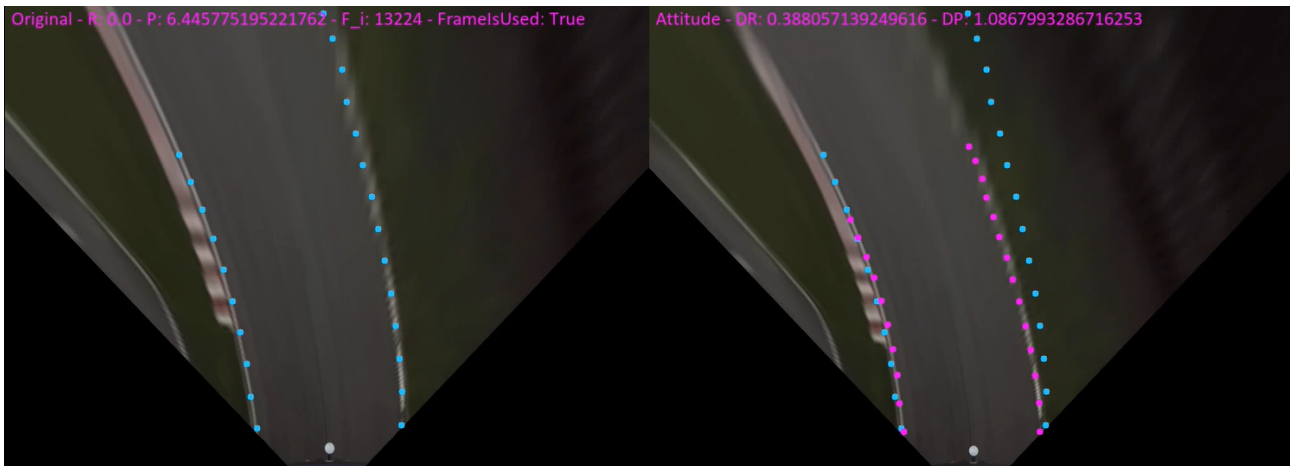


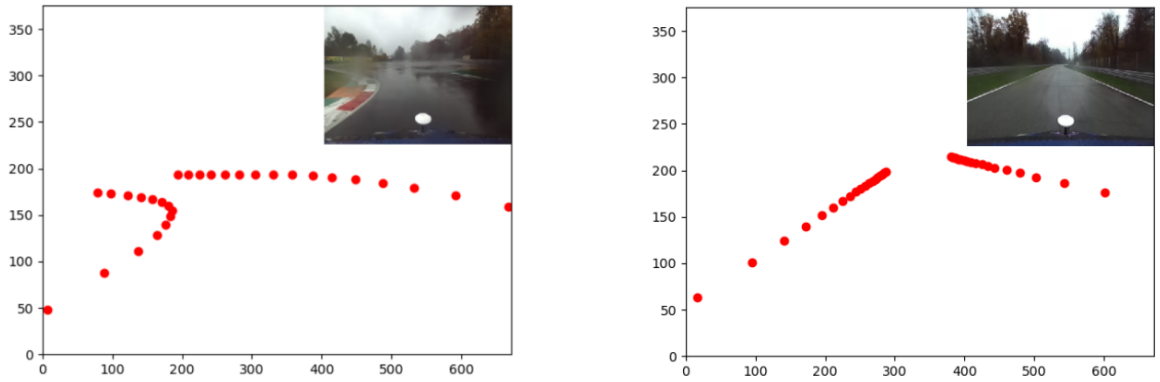
Figure 17: The BEV images and the extracted world points, in the left, using the constant pre-calibrated IPM and the right using the adaptive IPM. As written on top of the corrected BEV image, for this particular frame in the dataset, the correction of the relative attitude, from pre-calibrated value, are roughly 0.39 and 1.09 degrees and for the roll and pitch, respectively. For comparison purposes only, the world points using the pre-calibrated IPM are also plotted, in blue, on the corrected BEV image.

9.1. Virtual Camera

The Virtual Camera is view matrix of a camera with a calibration similar to the actual camera on the vehicle, except for roll and pitch angles, and following the exact same trajectory as the vehicle, thanks to Centimeter-accurate RTK-GPS measurements provided by the dataset.
Using the virtual camera and ground truth of the lines for the circuits, also provided by the dataset, it is possible to project the ground truth line points to the viewpoint of the virtual camera, and create images of the road lines as they would have been seen from the viewpoint of an actual camera with the same calibration.
A virtual camera could be set to have any arbitrarily desired calibration, including a sinusoidal attitude with a desired amplitude and frequency, enabling us to create a dataset of images with a desired roll and pitch angles. (FIGURE)

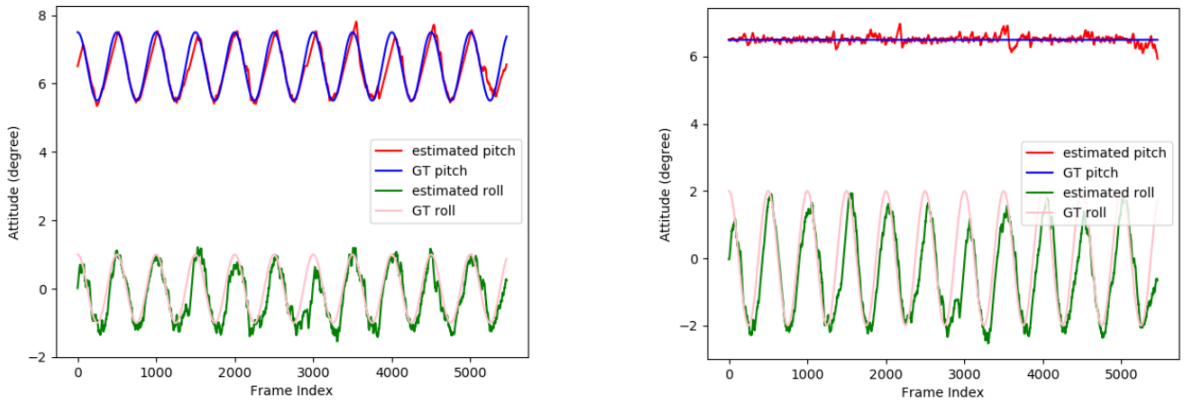
9.2. Evaluation 1

The goal of the first evaluation method is to understand and confirm the performance of the algorithm using ideally detected lines. In this evaluation method, images are created using an oscillating virtual camera, whit desired attitude, and the idea is to evaluate the ability of the proposed pipeline in extracting the true attitude of the virtual camera by examining the images only.
It is important to note that the ground truth provided by this dataset is far from perfect, in fact, it is also subject to errors in the measurement, and thus, similar to real-life scenarios a post-processing filter is needed. In this setup however, since we have no IMU data for the virtual camera, for the filtering purposes, the proposed Kalman Filter could not be used and therefore a Moving Average is used, which reduces the response bandwidth of the system hindering its ability to follow fast oscillations.
As shown in Figure 19 the algorithm can effectively track the oscillations of the virtual camera, albeit with some delay due to MA, and as expected the pitch estimation is more robust and smooth than roll estimation.



(a) (b)

Figure 18: Two sample frames that are rendered using the virtual camera and ground truth line points. The virtual camera calibration is exactly like the actual camera except for a slightly modified roll of about 10 degrees, and thus the rendered images look very similar to actual images from the dataset.



(a) (b)

Figure 19: The estimated attitude from the algorithm is closely following the actual ground truth attitude of the virtual camera with some noise. Compared to pitch, estimating roll is much more difficult and noisy.

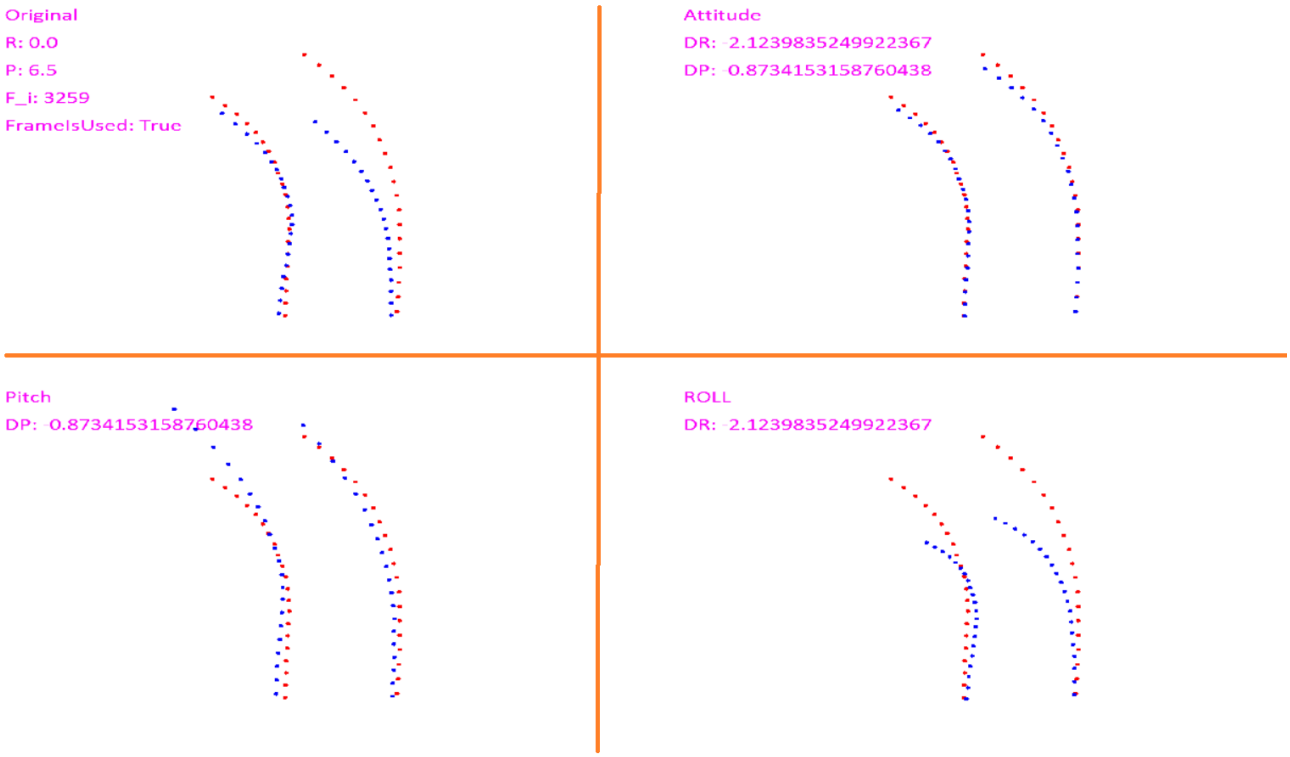
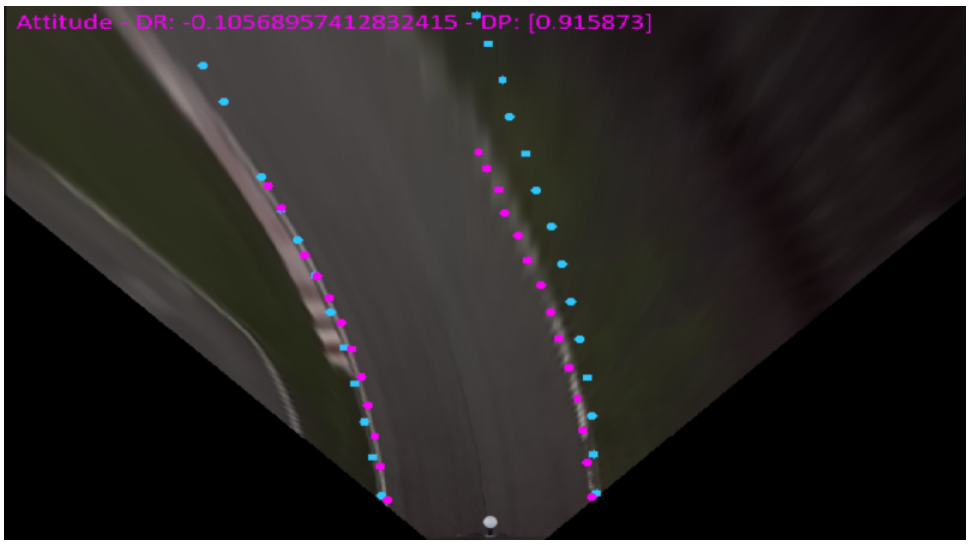


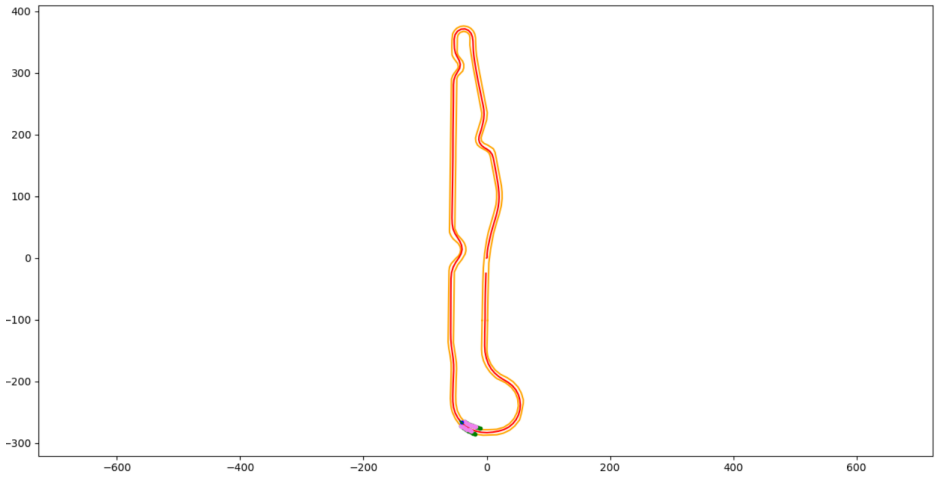
Figure 20: An instance of the BEV images created using the rendered images from the virtual camera. The world points plotted in red are projected to BEV using the ground truth IPM of the virtual camera which is oscillating. On the top row, the world points plotted in blue, are projected to BEV using the constant pre-calibrated IPM (top left) and the adaptive IPM (top right) and on the bottom row the blue points are projected using only pitch and only roll corrected IPMs. Comparing to the the ground truth IPM of the virtual, obviously, the adaptive IPM creates the most accurate world points among all IPMs.

9.3. Evaluation 2

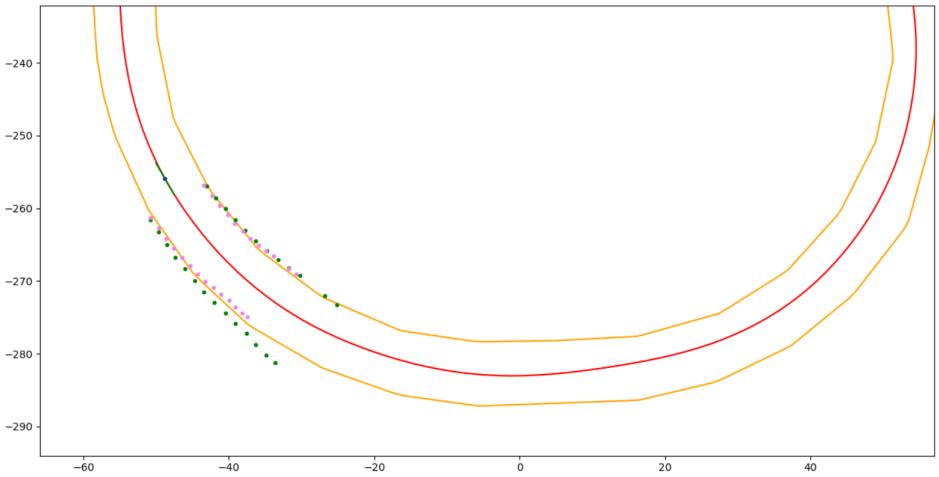
The goal of the second evaluation method is to understand the performance of the algorithm in real-life scenarios where the world lines are extracted using the line detection pipeline from real images having a considerable noise or sometimes a gross error.
As shown in Figure 21, the second evaluation method uses the GPS pose of the vehicle and a linear interpolating, to smoothly project the detected world points from the vehicle's local world frame (see section 2.1) to the earth frame of reference (i.e. a map), where the world points could be compared against the ground truth lines to measure the error.



(a) The extracted world points, using both of the IPMs, are plotted on the corrected BEV image.



(b) Same world points from previous BEV image projected on the map of Arese circuit.



(c) Zoomed in version of the previous plot. Perceptibly the world points using the adaptive IPM (pink points) are noticeably closer to ground truth of the lines, compared to the green points.

Figure 21: Using the precise GPS trajectory of the vehicle, plotted in red, the extracted world pints are projected on the map to be compared against the ground truth of the lines that are plotted in orange

In order to measure the distance of the world pints from the ground truth lines, the initial step is to fit the ground line point with a first order spline and extract the knots of the spline. Then distance of a single points w.r.t. the spline could be computed using following equations:

$$\overrightarrow{v_i} = p - k_i \tag{58}$$

$$\overrightarrow{e_i} = k_{i+1} - k_i \tag{59}$$

$$j = \text{argmin}(|v_i|) \tag{60}$$

$$e_m = \begin{cases} e_j, & \text{if } \arccos\left(\frac{\overrightarrow{v_j}.\overrightarrow{e_j}}{|v_j||e_j|}\right) < \arccos\left(\frac{\overrightarrow{(v_j)}.\overrightarrow{(-e_{j-1})}}{|v_j||e_{j-1}|}\right) \\ -e_{j-1}, & \text{otherwise} \end{cases} \tag{61}$$

$$distance = |v_j| \times \sqrt{1 - \left(\frac{\overrightarrow{v_j}.\overrightarrow{e_m}}{|v_j||e_m|}\right)^2} \tag{62}$$

As shown in Figure 22, in the above equations p is the desired point and k_i are the knots of the spline.

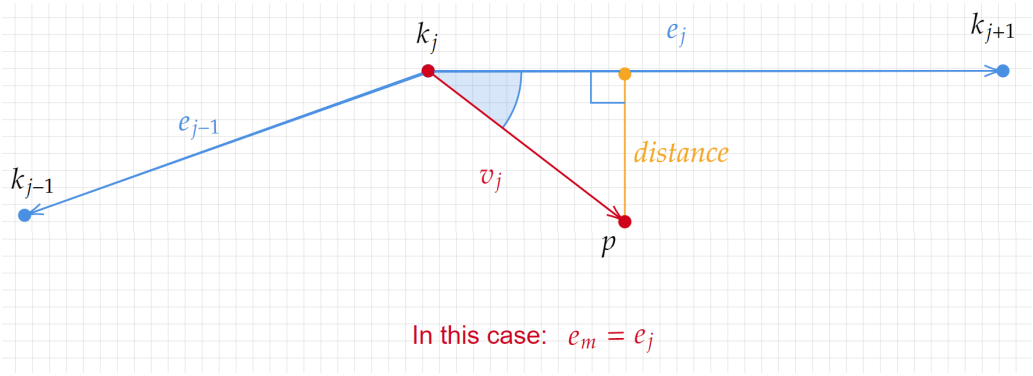


Figure 22: Computing the distance of a point w.r.t a first order spline using simple vector operation.

By computing the distance for all of the world points w.r.t. the spline in all of the frames, the mean, standard deviation, confidence interval and the quartiles are calculated for the inner and outer lines separately. After noticing that the error is most significant on parts of the track where the vehicle is bending or cornering, for a better evaluation we decided to compute all of the above metrics separately for the straight and curved sections. Tables 1 show the evaluation results of detected world points with and without the adaptive IPM.
Even in the presence of large magnitude errors due to line detection, as shown in Figure 23, which increase the mean error and could not be mitigated by adjusting the IPM, the mean error drops compared to original IPM, especially, on curved road section. In some cases the mean error may increase slightly nonetheless the overall performance is acceptable.
The other important thing to notice is the minimum error of about 0.5 meters in all the corrected cases. The reasons behind this error are:
The error in heading direction of the vehicle extracted form GPS data, which makes the projection of world points to the map, slightly inaccurate.
The second source of this error is due to rather large width of the road lines, and placing the centroid of the points, for world points and the ground truth points, on different side of the line.
The third reason is simply the oscillations and errors in attitude estimation. After all there is nothing perfect in this world.

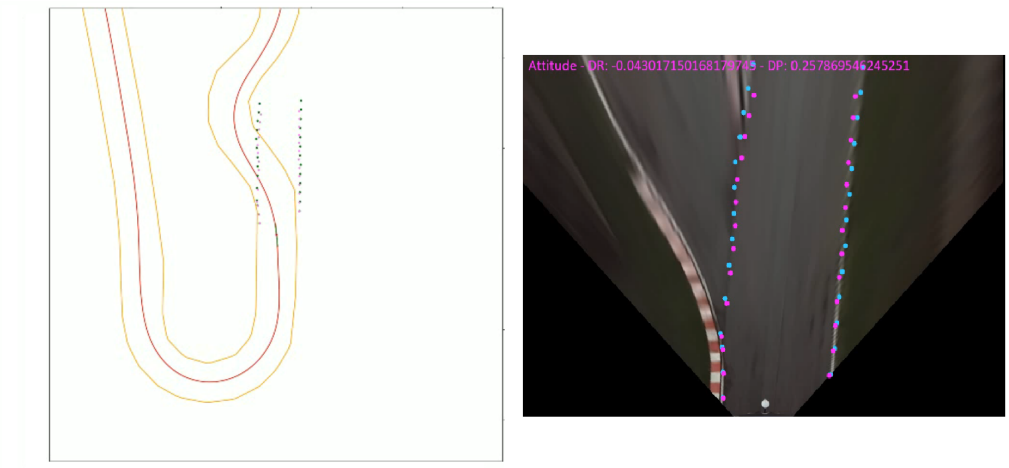


Figure 23: While using the full track test in the evaluation 2, there are many instances where the extracted world points belong to different road than which the vehicle is moving on, and thus, the world points are compared against a different ground truth which results in large value errors that dominate the mean error. In this particular frame, the vehicle, and consequently, the ground truth of the lines, are turning left in the road split, but the extracted world points belong to the straight road.

Evaluation on Arese track

	Mean	STD	W2STD	CI1	CI2	Q1	Median	Q3
OCO	1.53	0.95	94.14	1.283	1.417	0.6991	1.307	1.667
OCC	0.58	0.50	93.89	0.5453	0.6156	0.2359	0.4126	0.678
OSO	0.507	0.4738	94.96	0.4582	0.5558	0.0871	0.4096	0.789
OSC	0.3771	0.2586	96.02	0.3505	0.4037	0.1648	0.2999	0.555
ICO	0.538	0.6479	99.15	0.4927	0.5832	0.2991	0.4231	0.5175
ICC	0.4192	0.5785	99.39	0.3788	0.4596	0.2261	0.311	0.4371
ISO	0.518	1.176	97.26	0.3949	0.6412	0.2068	0.2618	0.3317
ISC	0.6281	1.097	97.26	0.5133	0.7429	0.2533	0.4488	0.5723

Table 1: This table provides error distribution characteristic and comparison for original and corrected IPMs. The naming schema is for the rows of the table is the short of the following format: [Outer/Inner][Curved/Straight][Original/Corrected]. For instance the row including error distribution data for Inner Curved Original line points are represented by ICO for short.

9.4. Time Performance

The time performance of the proposed algorithm is very good, allowing it to be utilized in real-time systems. Since the optimization step performs only six iterations in total, the computation time could be less than 5 milliseconds per frame, on an average CPU, which could be further improved by employing multi-processing techniques to perform optimization on six separate processes running in parallel.