

VISION BASED ROAD SLOPE ESTIMATION



by
Eser Üstünel

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Electrical and Electronics Engineering

Yeditepe University

2021

VISION BASED ROAD SLOPE ESTIMATION

APPROVED BY:

Assoc. Prof. Dr. Engin Maşazade
(Thesis Supervisor)
(Marmara University)

.....

Prof. Dr. Duygun Erol Barkana
(Yeditepe University)

.....

Prof. Dr. Emin Erkan Korkmaz
(Yeditepe University)

.....

Prof. Dr. Hazım Kemal Ekenel
(Istanbul Technical University)

.....

Assoc. Prof. Dr. Arzucan Özgür
(Bogazici University)

.....

DATE OF APPROVAL: /.... /2021

I hereby declare that this thesis is my own work and that all information in this thesis has been obtained and presented in accordance with academic rules and ethical conduct. I have fully cited and referenced all material and results as required by these rules and conduct, and this thesis study does not contain any plagiarism. If any material used in the thesis requires copyright, the necessary permissions have been obtained. No material from this thesis has been used for the award of another degree.

I accept all kinds of legal liability that may arise in case contrary to these situations.

Name, Last name

Signature

ACKNOWLEDGEMENTS

I would like to thank my supervisor Assoc. Prof. Dr. Engin Maşazade for all the support and education he has given me during my PhD study. I would like to thank Prof. Dr. Emin Erkan Korkmaz and Assoc. Prof. Dr. Arzucan Özgür for being a committee member and their support throughout my study at Yeditepe University. I would like to thank Prof. Dr. Duygun Erol Barkana and Prof. Dr. Hazım Kemal Ekenel for being a committee member. I would like to thank my wife Dr. Bilgen Ekim Üstünel for her endless love and understanding. I would like to thank my children Alya (7-year old) and Mete (3-year-old) for their endless noise and obstruction.

ABSTRACT

VISION BASED ROAD SLOPE ESTIMATION

Lateral and longitudinal control of a vehicle are the basis of autonomous driving. Road slopes (parameters) are vital information that are used to control a vehicle laterally in lane keeping assistance and longitudinally in adaptive cruise control respectively. We present four methods estimating road slopes or 3D road parameters from instant road images using a front monocular camera. We start with a simple algorithm Geometry Based (GB) method that extracts invariant properties on 2D image. The second solution Covariance Based (CB) method, which uses covariance matrix properties of 2D road lines as features, is similar to supervised learning techniques. We present a systematic road slope estimation called as Iterative Least Squares with Optimization (ILSO) in our third solution. The algorithm is smartly divided into two processes as Least Squares (LS) and optimization depending on their estimation capabilities. LS estimates road parameters and optimization step estimates longitudinal distances. It is an iterative estimation that for each iteration LS and optimization run one time. It is an easy technique due to its systematic character that it does not require complex geometric calculations and match between left and right line. In our final method Local Features Based (LFB), we use the SIFT local features between two consecutive image pairs to estimate road slopes. GB, CB and ILSO methods rely on road lines and provide in advance estimations that enables proactive driving, whereas LFB uses the entire image and estimates road slopes when the road slope actually changes at that moment. All our methods are analytical and give explicit results except CB, which predicts a parameter by choosing a minimum metric. We use two different road models to characterize 3D road parameters for the algorithms. Multi-Layer Perceptron (MLP) learning method is used as a benchmark in our tests. We compare the performances and system requirements of our methods under controlled and uncontrolled test environments.

ÖZET

KAMERA GÖRÜNTÜLERİ İLE YOL EĞİMLERİNİN BULUNMASI

Bir aracın yanal ve uzunlamasına kontrolü otomatik sürüşün temelidir. Yol eğimleri (ya da parametreleri) şerit tutma yardımıyla yanal olarak ve uyarlamalı hız sabitleyicide boylamasına olarak bir aracı kontrol etmek için kullanılan hayati bilgilerdir. Ön tek-gözlü kamera kullanılarak elde edilen anlık yol görüntülerinden yol eğimlerini veya 3 boyutlu (3D) yol parametrelerini kestiren dört yöntem öneriyoruz. İlk metod Geometri Tabanlı (GB) yöntem 2D görüntüdeki değişmez özellikleri kullanarak yol parametrelerini bulur. İkinci bir çözüm olan Kovaryans Tabanlı (CB) yöntemde 2D yol çizgilerini algoritmamızı eğitmek için kullanıyoruz. Bu metod çizgilerin kovaryans matrisi özelliklerini kullanır. Üçüncü çözümümüzde Yinelemeli En Küçük Kareler (EKK) ve Optimizasyon (ILSO) olarak adlandırılan sistematik bir yol eğimi kestirmesini öneriyoruz. Bu algoritma tahmin yeteneklerine bağlı olarak akıllıca En Küçük Kareler (Least Squares) ve Optimizasyon olmak üzere iki basamağa ayrılmıştır. EKK yol parametrelerini bulur ve optimizasyon adımı uzaklıkları bulur. Her yineleme EKK ve optimizasyonun bir kez çalışmasıdır. Karmaşık geometrik hesaplamaları, sol-sağ çizgileri arasında eşleşmeyi gerektirmemesi ve sistematik karakteri nedeniyle ILSO kolay bir tekniktir. Yerel Özellik Tabanlı (LFB) son yöntemimizde yol eğimlerini bulmak için iki ardışık görüntü arasındaki SIFT yerel özelliklerini kullanıyoruz. GB, CB ve ILSO yöntemleri yol çizgilerine dayanır ve proaktif sürüşü mümkün kılan önceden kestirmeler sağlarken, LFB tüm görüntüyü kullanır ve yol eğimi o anda gerçekten değiştiğinde yol eğimlerini bulur. Tüm yöntemlerimiz analitiktir ve bir metriği en küçükleyerek yol parametresini bulan CB dışında hepsi açık sonuçlar verir. Algoritmalar için 3D yol parametrelerini karakterize etmek için iki farklı yol modeli kullanılır. Çok Katmanlı Algılayıcı (MLP) öğrenme yöntemi testlerimizde bir kıyaslama olarak kullanılmaktadır. Önerilen yöntemlerin kestirme performansını, sistem gereksinimlerini ve uygulanabilir koşullarını kontrollü ve kontrolsüz test ortamları altında karşılaştırılır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iv
ABSTRACT	v
ÖZET	vi
LIST OF FIGURES	ix
LIST OF TABLES	xii
LIST OF SYMBOLS/ABBREVIATIONS	xiii
1. INTRODUCTION	1
1.1. LITERATURE SURVEY	2
1.2. LIST OF CONTRIBUTIONS	6
1.3. THESIS ORGANISATION	13
2. PRELIMINARIES	15
2.1. TRIGONOMETRIC EQUATIONS	15
2.1.1. Linearity of Sine Function Around 0 Degree	15
2.1.2. Solution for Addition of Cosine and Sine Function	15
2.2. MATRIX THEORY	16
2.2.1. Covariance Matrix and Its Properties	16
2.2.2. Column Space (CS) and Null Space (NS) of a Matrix	17
2.2.3. Eigenvalues, Eigenvectors and Principal Component Analysis (PCA)	17
2.2.4. Approximate Null Space (ANS)	18
2.3. SCALE-INVARIANT FEATURE TRANSFORM (SIFT)	18
2.4. MULTILAYER PERCEPTRON (MLP)	19
2.5. SIGNAL-TO-NOISE RATIO (SNR)	21
3. SYSTEM MODEL	23
3.1. BASIC ROAD MODEL (BRM)	24
3.1.1. Basic Road Model for Straight Roads	24
3.1.2. Basic Road Model for Circular Roads	26
3.2. CLOTHOID ROAD MODEL (CRM)	28
3.3. BASIC RM (BRM) vs CLOTHOID RM (CRM)	29
3.4. PINHOLE CAMERA (PHC) MODEL	30
3.5. EQUIVALENT FOCAL LENGTH CALCULATION	31

4. GEOMETRY BASED (GB) SLOPE ESTIMATION	33
4.1. GB LATERAL SLOPE ESTIMATION ON STRAIGHT BANKED ROADS ...	33
4.2. GB LATERAL SLOPE ESTIMATION ON CURVES	35
4.3. GB LONGITUDINAL SLOPE ESTIMATION ON STRAIGHT ROADS	38
4.4. GB LONGITUDINAL SLOPE ESTIMATION ON CURVES	40
5. COVARIANCE BASED (CB) SLOPE ESTIMATION	41
5.1. UPDATED COVARIANCE BASED METHOD	45
6. ITERATIVE LEAST SQUARES (LS) ESTIMATION AND OPTIMIZATION (ILSO)	48
6.1. ITERATIVE LEAST SQUARES (LS) ESTIMATION	49
6.2. OPTIMIZATION FOR DISTANCE ESTIMATION	52
6.3. ITERATION STOP CRITERIA	56
7. LOCAL FEATURES BASED (LFB) SLOPE ESTIMATION	57
7.1. LFB: LATERAL ANGLE ESTIMATION	59
7.2. LFB: LONGITUDINAL ANGLE ESTIMATION	60
7.3. LFB: YAW ANGLE ESTIMATION	61
8. NUMERICAL RESULTS AND DISCUSSIONS	63
8.1. RESULTS USING BASIC RM	63
8.1.1. Geometry Based Results	63
8.1.2. Local Features Based Results	67
8.1.3. Covariance Based Results	68
8.1.4. Comparison of Methods	69
8.2. RESULTS USING CLOTHOID RM	74
8.2.1. Curvature Requirements on Highways	76
8.2.2. System Internal Dynamics	77
8.2.3. Comparison of Methods in Simulations	78
8.2.4. Virtual and Real Images Results	83
9. DISCUSSIONS	88
10. CONCLUSION	91
REFERENCES	95

LIST OF FIGURES

Figure 1.1.	Annotation of road. (a) Berkeley, (b) Cityscapes datasets.	6
Figure 1.2.	Annotation of road. (a) KITTI, (b) Caltech, (c) VPGNet datasets.	7
Figure 2.1.	Sin() and cos() about 0°	16
Figure 2.2.	MLP with 4 layers	20
Figure 3.1.	Front camera installed on a car.....	23
Figure 3.2.	3D angles. (a) Roll, pitch, yaw angles, (b) Angles for a vehicle.	25
Figure 3.3.	Straight banked road	26
Figure 3.4.	CarMaker image and basic road model with $R=1000m$	27
Figure 3.5.	Road model as a bird's eye view	28
Figure 3.6.	3D road profile. (a) Horizontal profile, (b) Vertical profile.....	29
Figure 3.7.	3D road profile. (a) Lateral slope (α), (b) Projected 2D view.	30
Figure 3.8.	Pinhole camera (PHC) model.....	31
Figure 3.9.	Similar triangles to find equivalent focal length (f).....	32
Figure 3.10.	A straight road with left and right lines	32
Figure 4.1.	An uphill road starts after plain (side view)	39
Figure 4.2.	A downhill road starts after plain (side view)	40
Figure 5.1.	Generation of Y_N vector for lateral and longitudinal slope	42
Figure 5.2.	V-shape of distance values for CB method at 50dB	46
Figure 6.1.	Iterative estimation processes.....	49

Figure 6.2.	Square error at $Z=10\text{m}$ and $Z=20\text{m}$ for v_l (linear scale)	53
Figure 6.3.	Square error at $Z=10\text{m}$ and $Z=20\text{m}$ for v_l (logarithmic scale).....	53
Figure 7.1.	SIFT matches between 2 correlated images	58
Figure 7.2.	Longitudinal slope estimation by SIFT	61
Figure 7.3.	Yaw angle estimation by SIFT	62
Figure 8.1.	Lateral slope estimations (GB,CarMaker)	64
Figure 8.2.	Lateral system dynamics (GB,BRM). Effect of m_l and d_l errors.	65
Figure 8.3.	Lateral system dynamics (GB,BRM). Effect of R for $\alpha = 5^\circ$	65
Figure 8.4.	Longitudinal slope estimation errors for uphill (GB,CarMaker)	66
Figure 8.5.	System dynamics depending on Z_b for β uphill (GB,CarMaker)	66
Figure 8.6.	Longitudinal slope estimations for downhill (GB,CarMaker).....	67
Figure 8.7.	Downhill system dynamics depending on Z_{min} (GB, CarMaker)	67
Figure 8.8.	Slope estimations (LFB, CarMaker)	68
Figure 8.9.	Comparison of CB and MLP by using BRM. Roll angle (α) in degrees .	69
Figure 8.10.	Comparison of CB and MLP by using BRM. Pitch angle (β) in degrees	70
Figure 8.11.	Lateral slope estimations on a curve ($R=1000\text{m}$) with noise over BRM .	71
Figure 8.12.	Comparison of methods for lateral slope (All Methods, CarMaker)	72
Figure 8.13.	Comparison of methods for longitudinal slope (All Methods, CarMaker)	72
Figure 8.14.	Before the longitudinal slope	73
Figure 8.15.	On the longitudinal slope	74
Figure 8.16.	Road lines with $f = 1$ and different noise variances (30dB, 40dB, 50dB)	75
Figure 8.17.	Distance (Z_l, Z_r) and position (d_l, d_r) estimations over SNR	78

Figure 8.18.	Mean number of iterations for different number of samples over SNR ..	79
Figure 8.19.	Distance to the left line (d_l) estimation	80
Figure 8.20.	Lateral angle (α) estimation.....	81
Figure 8.21.	Horizontal parameters (a_h, b_h) estimation.....	81
Figure 8.22.	Vertical parameters (a_v, b_v) estimation.....	82
Figure 8.23.	Longitudinal slope estimation for 5 percent (2.862°) after 30m	84
Figure 8.24.	Lateral slope estimation for 5 percent (2.862°) in the curve	84
Figure 8.25.	A straight road A656 from Mannheim to Heidelberg.....	85
Figure 8.26.	Straight left lane on uphill	86
Figure 8.27.	Right highway exit on uphill	86
Figure 8.28.	Left banked curve on uphill	86

LIST OF TABLES

Table 1.1.	Summary of the contributions	13
Table 2.1.	Summary of activation energy and loss function	21
Table 4.1.	Actual values for (4.10)	38
Table 8.1.	Longitudinal estimations for real road in Figure 8.14	73
Table 8.2.	Estimated parameters for synthetic images	84
Table 8.3.	Calibration parameters of Figure 8.25.....	85
Table 8.4.	Estimated parameters for real test images.....	87
Table 9.1.	Summary for the methods	90
Table 9.2.	Comparison of complexities (in time)	90

LIST OF SYMBOLS/ABBREVIATIONS

α	Lateral slope angle
β	Longitudinal slope angle in BRM
θ	Curvature angle in BRM
Δ	Threshold for exiting iterations
A	Matrix
$A[n, m]$	An element in a matrix
a	Vector
$a[n]$	An element in a vector
a_h	First order horizontal parameter
a_v	First order vertical parameter
b_h	Second order horizontal parameter
b_v	Second order vertical parameter
c_U	Left and right line corelation constant
d	Lane width
d_l	Distance to left line
d_r	Distance to right line
$E[.]$	Expectation operation
f	Focal Length
fps	Frame per second
N_l	Number of samples on left line
N_r	Number of samples on right line
$.k$	iteration index
R	Radius of a curve
κ	Curvature of a curve
U	Uniform distribution
ANN	Artificial neural network
ANS	Approximate null space

AWGN	Additive white Gaussian noise
BRM	Basic road model
CNN	Convolutional neural network
CRM	Clothoid road model
CCE	Categorical cross entropy
CS	Column space
DL	Deep learning
LS	Least squares
MAE	Mean absolute error
ML	Machine learning
MLP	Multilayer perceptron
MSE	Mean square error
PHC	Pinhole camera
RM	Road model
SIFT	Scale invariant feature transform
SNR	Signal to noise ratio

1. INTRODUCTION

Advanced Driver Assistance Systems (ADAS) in a vehicle has found great applications and implementations in last three decades due to improved and cheaper computer technology. Nowadays, sophisticated algorithms can run on different Electronic Control Units (ECU) where each ECU has its own functionality such as motor, powertrain, braking, airbag, etc. As the expectations from the vehicle increase, camera and computer vision-based methods have also found big Research and Development (R&D) area for both researchers and Original Equipment Manufacturers (OEMs). Therefore, the Camera ECU (CECU) has become a standard hardware within mid-class or above car models. The CECU captures video frames which are then used for various functions such as lane departure warning and Lane Keeping Assistance (LKA) by implementing road and road line detection. Collision warning/avoidance and Adaptive Cruise Control (ACC) functions are realized by implementing object, vehicle, pedestrian detection. Therefore image processing and computer vision algorithms should be accurate and robust since CECU is also an Active Safety (AS) tool in a vehicle. Furthermore, the signals obtained from camera can be also forwarded to be used in other ECUs or a central ECU which may be responsible of fusion of radars, lidars and other camera signals.

Front camera has already been deployed as perception sensor. Now it is time to use the camera for another purpose as a software solution without any additional hardware or harness need. In this work, we use the monocular camera solely for the purpose of road slope estimation. Road slope (lateral slope, curvature and vertical angles) estimation or in other words 3D road parameters (horizontal and vertical profiles) estimation is useful in many aspects. Lateral slope (roll angle) refers to the angle between the line on the road width and the earth's horizontal plane. Lateral slope estimation can be used in a LKA system for a better control of car location in the lane and for a proper speed in order to take the curve securely. Even on a straight road, there is some lateral slope in order to disperse the rain. On the other hand, longitudinal slope (pitch angle) or vertical profile indicates an uphill or a downhill on the road. Longitudinal slope estimation can be helpful for ACC, where detecting the uphill or downhill slope in advance improves the performance of ACC.

Similarly, a downhill slope estimation can prevent rapid acceleration of a car, which has been fast enough on a straight road.

A vision-based slope (3D road parameters) estimation can give the slope information earlier than the vehicle enters this road section, which has different slope. This gives significant amount of time to be proactive for a vehicle. As an example, there can be a case that a driver feels that the vehicle mistakenly shifts up to higher gear before coming to the uphill. Whenever it reaches to the uphill with small speed and with high gear, powertrain may have difficulty to push the vehicle. That would not be a problem, if the car could have estimated the uphill in advance. Moreover, the lifetime of the gearbox might be extended, since its ECU would choose the proper gear value according to the road and it would not change the gear frequently.

1.1. LITERATURE SURVEY

We divide our survey into groups in terms of types of sensors and their usage.

Mechanical Sensors: In the literature, there exists non-camera based research for slope estimation or classification which generally consider longitudinal slope. In [1] longitudinal slope was estimated by using GPS, pressure and motor torque sensors. In [2, 3] GPS and onboard sensors such as engine control sensor, transmission control sensor, wheel speed sensor, accelerometer and vehicle's longitudinal model were integrated in a probabilistic data association filter. In [4, 5] accelerometer sensor and vehicle longitudinal dynamic model were used together in an Extended Kalman Filter for the pitch angle as for a longitudinal slope estimation. Ultrasonic sensors were used slope estimation in [6]. Roll angle estimation of 2-wheel vehicles such as motorbikes was presented by using gyroscope and speed sensors in [7]. Vertical slope and vehicle position were estimated by using on board sensors and GPS in [8]. On the other hand, as shown in [1], several types of noise can affect the performance of GPS and mechanical sensors. As an example, a GPS sensor can make an error of 4 meter on a 2D map and around 15m vertically. Inertial Measurement Unit (IMU) sensor was used for 3D accelerations and 3D angular velocities in [7, 2, 3]. However cheaper IMUs have fewer dimensions for accelerations and velocities. IMU signals have

thermal noise, whose variance could depend on temperature, which may further require sophisticated noise suppression filters. Sampling on the sensor in IMU has also drift due to oscillators i.e. there may be drift on the signal over time. Moreover, an extra harness is needed for IMU ECU.

Lidars and Radars: Lidars were used for road or obstacle detection in [9, 10] and road curb detection in [11]. A fusion system between lidar and camera was used for drivable region detection [12] or object tracking [13]. Spline fit model was used road boundary detection in [14].

A recent work [15] used Lidars and deep learning network to find out height of grid cells on the road. That can be useful to find out drivable road detection in city roads, however on highway scenario abundance of road point clouds should be converted to road model parameters to give succinct and meaningful information to the control systems of a vehicle. Generally, lidars are still expensive comparing to the cameras and they need complex algorithm to detect road surface comparing to detecting road lines in camera. In addition, lidars cannot be used in road line detection therefore they don't give any information about vehicle's position in the lane and curvatures ahead.

General Camera Usage: Object (vehicle, pedestrian, obstacle, sign) and lane (line) detections have always been important and known tasks for the Camera ECU. Recent information and research directions were given for vehicle detection in [16, 17], pedestrian detection in [18], obstacle detection in [19], sign detection in [20], and lane detection in [21, 22]. Maximum a posterior estimation as probabilistic approach was used for road detection in [23]. Extracted vertical profile is combined with edge detection in a stereo camera to find urban lane markings [24]. A different camera type, which is fisheye camera creating omnidirectional images, is used to detect straight lines in [25].

In image based research, feature selection has been the fundamental step for object recognition [26]. Classical approaches using Local Features such as SIFT [27], Gabor [28], BRISK [29], SURF [30] descriptors require less computational resources, hence suitable to implement on a real-time embedded system. Some of the features such as SIFT, Gabor

and BRISK provide invariance against some level of rotation, translation, scaling, and illumination change. Recently, for recognition and classification problems, advanced Deep Learning (DL) algorithms have been developed which are able to find their own features from vast data [31].

Depth Estimation using Cameras: Depth, which can be recalled as distance or range as well, can be found by using monocular cameras. Depending on detection or segmentation, depth was found in [32], which clustered the image based on edge, hue and saturation, and dark channel prior used saturation to estimate depth. Semantic information of segmentation was used for range estimation in [33]. A deep learning solution for range estimation was given in [34] using the continuity of range along objects or subregions and maximizing maximum a posteriori (MAP) of energy function. Another mono vision method [35] used unsupervised learning in Convolutional Neural Network (CNN). They claimed that unsupervised learning has key benefits since supervised networks require vast amount of ground truth image data. A very important depth solution from Mobileye, which is a famous computer vision company in automotive industry, was given as patent [36]. By finding image width of a vehicle from its back view and its approximate actual width, it was possible to find distance to the vehicle. Another monocular patent [37] converted RGB colors to YCbCr and minimizes an energy function on small patches to find depth. Stereovision was naturally used for distance estimation in [38, 39] and 3D reconstruction in [40]. Similar to stereo cameras, multiviews of monocular camera were used for depth and velocity estimation on a planar surface in robotics [41] by using Lyapunov-based stability analysis. Location of a monocular camera on a planar surface was found in [42] by using 2-point pose estimation between multi views.

Monocular Camera in Slope Estimations: There are monocular vision solutions for road slopes and parameters estimations in the literature. Slope classification as uphill and downhill was handled [43] by using both front and back monocular cameras. Since it was a binary classification, road slopes and parameters were not estimated. Road lines are important features for road slope estimation. Some important works for line detection were given in [44, 45, 46], where monocular monochromatic images were used. Under strict assumptions in [44], which were constant pitch, flat surface, null roll angle and with

very small horizontal curvature, 3D road structure could be found. Using consecutive frames and degenerative Kalman Filters, [45, 46] found only two parameters, which were distance of host vehicle to left line and host vehicle's orientation angle with respect to the lane. However, both works did not provide ground truth for two parameters and the estimated parameters fluctuate much between consecutive frames. K-Nearest Neighbors (KNN) algorithm was used to suppress outliers in road line detection and Extended Kalman Filter was used to track lines in [47]. However the method gave the result for host vehicle's position in the lane that was not proved against a ground truth. Another monocular camera solution for extracting 3D road structure with using the host car's kinematic information was given in [48]. A CNN method, called as 3D-LaneNet, was introduced for line detection in [49] using synthetic images for training and testing. However, it gave only position of the host car without using any mathematical model for the road.

Stereo Camera in Slope Estimations: Generally 3D road structure including road slopes can be found in stereovision solutions that may use complex disparity map calculations between matched points of two stereo images. Nedevchi et al [50, 51] used disparity maps and road lines for road structure estimation. A full geometric solution was given in [52] using disparity maps without requiring road lines. Instead of disparity maps [53] offered covariance matrix between road lines of two images for coherency. All mentioned stereo solutions used small angle assumptions and some road parameters depended on finding other road parameters first. There existed an interdependence between parameters. Special geometric conditions should have been derived to find road parameters.

Road height information was obtained using row-wise image registration using SURF features in two-view geometry in [54], similarly SURF based matched v-disparity was used for elevation estimation in [55]. Based on road height information drivable road can be segmented from the image [54, 55]. Potholes, which are small 3D holes on the roads, were found in [56] by using subpixel disparity. However, potholes don't give any clue about road parameters. Roll angle was estimated by RANSAC algorithm in stereo vision using disparity maps in [57], where they created artificial disparity maps for testing.

Virtual Data and Camera Datasets: Since real data or image acquisition is expensive

and accurate manual or automatic labeling (annotation) is difficult, using synthetic images becomes prevalent in computer vision [58]. Another work in [59] argued in favor of virtual sensor simulations and gave comparison between real sensors and virtual sensors. Moreover, sensor realistic simulations, which are provided by automotive suppliers [60, 61], are used in Hardware In Loop (HIL) tests in current automotive industry for perception sensors e.g. camera, lidar, radar.

There is a big vision database including 100k image frames annotated in **Berkeley dataset** [62]. 20k images for left image were annotated for a stereo camera in **Cityscapes dataset** [63]. The **KITTI dataset** in [64, 65] provided both lidar and camera annotated data frames. Annotated road lines and lanes were given in **Caltech dataset** [66] and another database considering different weather conditions was given in **VPGNet dataset** [67]. Annotations of road lines, lanes and drivable area are given in Figure 1.1 and in Figure 1.2 corresponding databases. However, all ground truth information are only pixel level annotations. That means; the labeled information given by databases are pixel locations of road lines. They don't give required 3D structure of the road nor the parameters we use in our work.



Figure 1.1. Annotation of road. (a) Berkeley, (b) Cityscapes datasets.

1.2. LIST OF CONTRIBUTIONS

There is neither enough work in monocular camera nor all road parameters are estimated as mentioned in Section 1.1. Therefore, we develop road slope estimation algorithms using monocular camera with the following motivations:

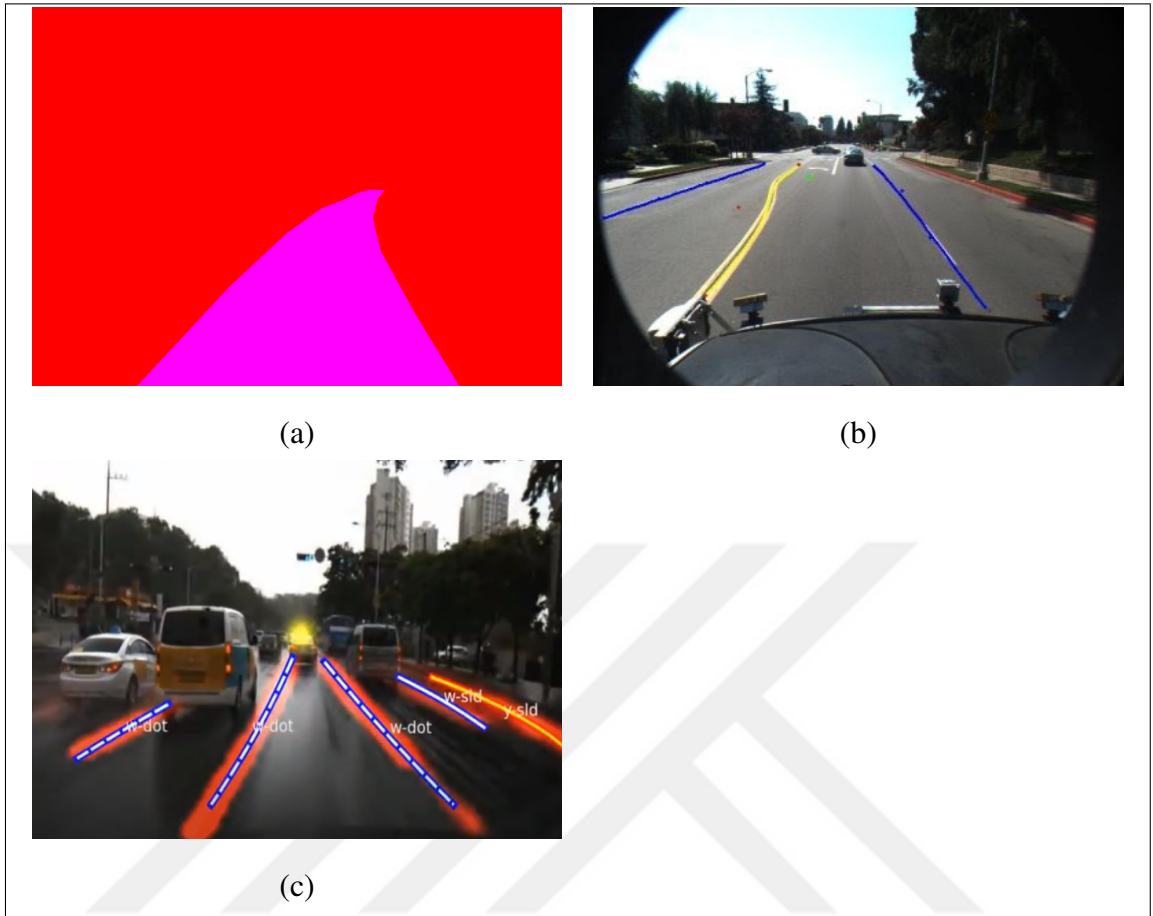


Figure 1.2. Annotation of road. (a) KITTI, (b) Caltech, (c) VPGNet datasets.

- Mechanical sensors have different kinds of disturbances such as thermal noise, timing drifts due to oscillators [1]. Moreover, mechanical sensors don't provide in advance estimations. However, cameras have noise only on pixel level, give in advance solutions, cheap to use and don't require additional cable harnesses.
- Stereovision solutions suffer from complexity in camera calibration and finding correspondence for disparity. In addition, regular texture or no-texture in images, which is high likely on road surfaces, makes it difficult to find correspondence between stereo images. However, calibration and algorithm development are easy in monocular cameras. In addition, current automotive projects predominantly use monocular cameras.
- Best of our knowledge our study is the first work, which estimates the road slope (parameters) directly from instant road images. However, monocular solutions [44, 45, 46] give estimations by using consecutive images.

In this study, we use two mathematical road models, which create road lines parametrically for characterizing 3D road structure. Road slopes or road parameters terms are interchangeably used throughout the thesis. A good 3D road model should be able to describe actual road lines in 3D with enough number of parameters. Simple road model (less number of parameters) would be easier for road parameter estimations, however approximation to real road line may deteriorate. Similarly, complex road models (more number of parameters) are difficult to estimate all parameters but it may approach better to road. Hence, we start with a Basic Road Model (BRM), which uses 5 parameters to create straight road line and circular road models. It proves that our methods on monocular camera can estimate 3D road structure. Then, we use Clothoid Road Model (CRM) given in [50, 53], which uses 7 parameters, which are position of vehicle in the lane (distances to left and right line), lateral bank angle, horizontal and vertical curvature parameters. Lateral slope or angle is about the inclination between left and right line due to different heights of lines. Horizontal parameters are about left and right turn in bird eye view. CRM characterizes road better than BRM because of using more number of parameters, and horizontal and vertical curvatures are expressed as polynomial of longitudinal distance. Note longitudinal distance, range and depth are synonyms throughout the thesis.

We utilize two different features which are road lines and the image Local Features (LF). For the first feature, since our scope is 3D road parameters estimation and there are enormous number of researches in road line detection, we assume road lines are given. The second feature LF can appear on the whole image. SIFT, SURF and BRISK are important local features. Positions of the local features change between image frames that can be used in getting road structure. We choose SIFT descriptors as features and estimations are done with respect to Basic RM using matched SIFT points. Note that relations of algorithms between road models and used features are given in Table 1.1.

Algorithm specific contributions of the thesis are further stated in detail below:

1. **Geometry Based (GB):** We develop geometric invariant properties (invariances) that they cancel out several unknown parameters for the benefit of finding the desired parameters. This method uses road lines as features. It uses instant images (1 frame at a time) and basic road model for estimations. Rather than estimating the road slopes iteratively between consecutive frames in [44, 45, 46, 47] or using the host car's kinematic information in [48], we utilize the basic relationship between the road lines and the road's lateral and longitudinal slopes on a single frame.
2. **Covariance Based (CB):** We convert a classification algorithm [68, 69, 70], which obtains covariance matrix of training vectors, into a parameter estimation problem. The method first applies Principal Component Analysis (PCA) on covariance matrix for dimension reduction of the matrix by suppressing noisy components. Then it finds Approximate Null Space (ANS). The algorithm takes the benefit of being perpendicular of column space and null space of covariance matrix. If a test vector is in column space, it is perpendicular to its class covariance matrix.

CB method does not require mapping between pinhole camera model and road model as in [44, 45, 46, 47]. Therefore it is straightforward and less complex since it only uses basic covariance of road lines and extracts classes afterwards.

It uses road lines as features and both Basic Road Model (BRM) and Clothoid Road Model (CRM) can be used within the method. As an improvement, we use more generic model CRM, can estimate all parameters of CRM and improve estimation by

linearization between class values comparing to Section 8.1.

- 3. Iterative Least Squares and Optimization (ILSO):** In our current study, we exploit 3D continuity of road lines based on CRM and their low pass characteristics for a challenging problem, which is depth or distance estimation on monocular vision. We propose an iterative solution that smartly divides the parameters estimation problem into two groups: In the first group, road parameters, which are used in clothoid road model, are estimated by Least Squares (LS) method. We show that LS is an easy and succinct method, which gives optimum solutions for parameters given the longitudinal distances of pixels on road lines. The road parameters can be lateral slope, curvature and uphill/downhill variables. In the second group, by using simple convex optimization considering the road model we can get the longitudinal distance values of close pixels on road lines to the vehicle. As having continuous, slowly changing, and strictly increasing (in distance) nature of road lines on highways, convex optimization can find longitudinal distances. The distances for distant pixels on road lines are updated in optimization step. All updated distance values are fed into LS block for a better road parameter estimations in the next iteration phase. As a loop, LS finds better road parameters by using updated distance values. The proposed method has good convergence to the actual distance values of pixels within 5 iterations. We name our method Iterative Least Squares and Optimization (ILSO).

One important benefit of proposed algorithm is that both blocks in iterative solution is analytic. It is not data driven solution neither heuristic solution. Moreover, there exists closed form results for both blocks. All those available mathematical derivations make the algorithm fast in each iteration loop. Another important benefit is that algorithm is systematic. It does not require any sequential conditions between parameters or intermediate steps as in stereo works [50, 51, 53]. All road parameters are found in one go and each parameter estimation is independent of each other. Note that only two stereo vision solutions [50, 53] gave all parameters. In addition, our algorithm does not require that left and right road pixels should be matched in order to have same distance to the camera compared to [46, 50, 51, 53]. Different number of pixels on each line and on different distances are welcome in our algorithm comparing to [46, 50, 51, 53], which need equal number of pixels on both road lines. That

increases simplicity and robustness of our algorithm against occlusion on road lines due to other vehicles. Additionally, we don't assume small angle simplifications for cosine or sine terms in lateral slope. Having the solutions on monocular camera makes real time implementation possible comparing to complex stereo solutions.

- 4. Local Features Based (LFB):** We use local features (SIFT descriptors) to find out current road parameters, which vehicle experiences at that moment. Therefore it is not in advance solution. This is the only method that does not require road lines. To the extent of our knowledge, our LFB method is the first work which compares solely two consecutive images to obtain an analytical solution for the road slope estimation problem. Note that LFB is the only method that utilizes only two successive frames, while all others estimate the parameters on a single frame as seen in Table 1.1. It uses basic road model.

In addition to our contributions above, we use an Artificial Neural Network (ANN) method, which is MultiLayer Perceptron (MLP) as a benchmark to compare with our proposed methods. Since we solve very specific problem in the thesis, annotated real camera frames for road parameters don't exist at the time of our work. Therefore virtual images, which is produced by a dedicated software [60, 61], can be used to train Convolutional Neural Network (CNN). However, that brings expensive licence fee issues for the software. If we assume that road lines are the only image cues about road parameters, then we can claim that our MLP usage will be the limit for CNN usage.

After stating short descriptions of the proposed methods, we bring together the significant contributions of all methods with respect to the literature below:

- We estimate all parameters, which can model the 3D structure of a road and vehicle position in the lane. However, monocular solutions in [45, 46] find only two parameters, which are distance to the left line and host vehicle's orientation angle within the lane. Both works don't provide annotations for comparison and the estimated parameters change abruptly during consecutive frames. Another monocular solution [44] works only under strict assumptions such as constant pitch, flat surface, null roll angle and with very small lateral curvature. A deep learning method finds

only position of the host vehicle using monocular camera in [49]. A stereo vision work estimates only roll angle in [57]. Generally, disparity solutions in stereo work [54, 55] and lidar work in [15] can create information about only height of the road that is about the vertical parameters. Another vision type, fisheye camera, is used to detect straight lines in [25].

- Range (distance) information disappears in the time of image formation. GB, CB and LFB methods are independent of range while [44, 45, 46, 50, 53] are dependent on range. Our approaches eliminate complexity of distance estimations in a camera or fusion complexity between different sensors [12, 48].
- We characterize 3D road shape with enough number of parameters (7 parameters in Clothoid Road Model). That information more helpful for ADAS functions such as Lane Keeping Assistance (LKA) and Adaptive Cruise Control (ACC). However, solutions in [15, 54, 55] create big data for the height of the road. Big data means here that hundreds or thousand of lidar points per frame should be analyzed possibly on the road surface. Therefore, the ADAS functions should have a complex preprocessing, which is road detection and mapping the big data to have succinct number road parameters for the purpose of the function.
- We analyze the important databases as seen in Figure 1.1 and conclude that they only give image segmentation (road lines, drivable area in terms of pixels) and classification (types of objects) ground truth. However, they don't annotate the parameters such as distance to left or right line, lateral slope, horizontal and vertical parameters, which are required in this work. Therefore, we record our own images and find plausible results that comply with visual verification. We also offer a process to create annotations (ground truth) for real images in Chapter 10 as a future work.

We show performance of algorithms under different noise variances that can represent different nonperfect situations such as road line detection errors, calibration errors and different weather and illumination conditions. We demonstrate performances under 100k independent simulated frames to have significant results in testing time. Straight lines for large scale vertical axis in result figures shows that enough testing cases are created in Monte Carlo simulations. Our numerical results show that ILSO is the best method withing all methods given that range values, which can be found iteratively.

For our above monocular camera based methods, an informative summary can be found in Table 1.1. The second column (*In Advance*) determines if the method finds road parameters before vehicle comes on this region defined by the road parameters. Only LFB method is not an in advance method. It basically finds current parameters. LFB requires 2 consecutive frames, however the others can find slope on instant images. In our recent development ILSO, we improve road model to clothoid one that is more generic than basic one. In addition, in ILSO method we also estimate distance (range) values in addition to road parameters. In column *Published*, CB and MLP appears in both road models and ILSO appears as current development to be appear in thesis. CB is the best method in Basic RM Section 8.1 and its improved estimation capability is given in the thesis as well for a comparison with our new method ILSO.

Table 1.1. Summary of the contributions

Methods	<i>In Advance</i>	<i>Road Models (RM)</i>	<i>Used Features</i>	<i>Number of Frames</i>	<i>Published</i>
GB	Yes	Basic RM	Road lines	1	in [71]
CB	Yes	both	Road lines	1	in both
ILSO	Yes	Clothoid RM	Road lines	1	in Thesis
LFB	No	Basic RM	SIFT	2	in [71]
<i>MLP</i>	Yes	both	Road lines	1	in both

As a short summary of our recent work [71], we show that a monocular camera can be used for estimating the parameters of Basic RM. We propose three analytical methods and a MLP method for benchmarking to estimate lateral and longitudinal slopes using single image frames. Detected road lines and local features e.g. SIFT are used as features in four algorithms.

1.3. THESIS ORGANISATION

The organisation of the thesis is given as follows.

In Chapter 2, we give succinct mathematical background, which can help to read the thesis.

In Chapter 3, we give two road models, comparison of road models and pinhole camera model, which is the projection model from 3D road model to image view in 2D.

In Chapter 4, we give basic geometric estimation based on basic road model (BRM). It extract invariances that can help slope estimation.

In Chapter 5, we converted a classification algorithm into a parameter estimation method. In the training time, information is gathered as covariance matrix and its Null Space matrix is found to be used in testing time. Principal Component Analysis (PCA) and Null Space analysis is explained in this section.

In Chapter 6, we developed an iterative Least Squares (LS) method with Optimization step. It divided all unknown parameters including road parameters and distance values into two groups. LS find road parameters and Optimization step finds distance (depth) values of pixels on road lines. This is our only method that gives also distance estimates.

In Chapter 7, we give local features based method, which is the only one that does not use any road model (RM). We show how to use any local features, which is SIFT in this case, to find out road slopes.

In Chapter 8, we compare our results on their defined environment such as road models or whole image. We give MLP results for bench-marking to our algorithms.

In Chapter 10, we summarize our thesis for developed algorithms, explain working conditions of each and limitations. We finally give an outlook on possible research directions and key suggestions for improvements on future applications.

2. PRELIMINARIES

In this chapter, we want to give a summary of mathematical background used in our derivations and algorithms. Comprehensive information can be found in given references.

2.1. TRIGONOMETRIC EQUATIONS

Since our work is related to geometry and angles, we give here reasoning and derivations for our trigonometric solutions.

2.1.1. Linearity of Sine Function Around 0 Degree

For the estimation of small angles around 0° , if there exists 2 options by using either $\sin()$ or $\cos()$ function or their inverses, we prefer to use $\sin()$, since it is more linear about 0° than $\cos()$ as seen in Figure 2.1. That means, $\cos()$ is not good responsive or sensitive to the angle changes close to zero. Moreover, estimations and equation derivations can have risk using $\cos()$, since it may have higher values than 1.0 under noise. That would result in a complex number for $\arccos()$ function, which is the inverse of cosine function().

2.1.2. Solution for Addition of Cosine and Sine Function

For the type of equation $C = A \cos(\alpha) + B \sin(-\alpha)$, we start deriving for α

$$\begin{aligned}
 C &= A \cos(\alpha) + B \sin(-\alpha) \\
 &= \sqrt{A^2 + B^2} \left(\frac{A}{\sqrt{A^2 + B^2}} \cos(\alpha) + \frac{B}{\sqrt{A^2 + B^2}} \sin(-\alpha) \right) \\
 &= \sqrt{A^2 + B^2} (\cos(b) \cos(\alpha) - \sin(b) \sin(\alpha)) \\
 &= \sqrt{A^2 + B^2} \cos(\alpha + b)
 \end{aligned} \tag{2.1}$$

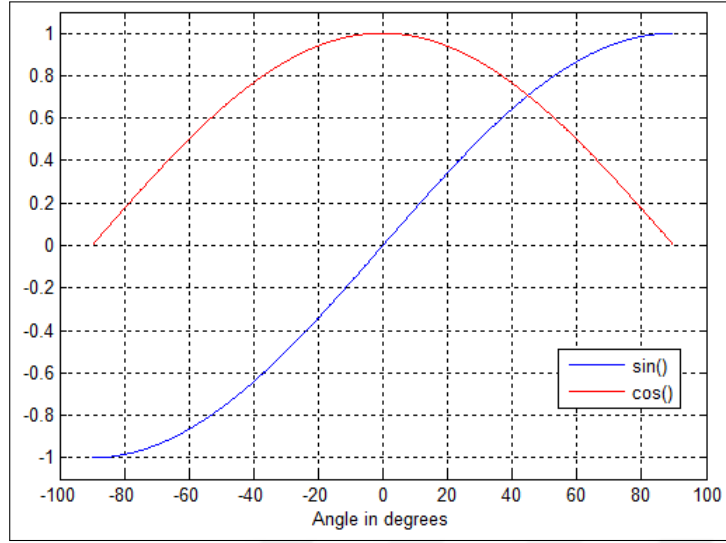


Figure 2.1. Sin() and cos() about 0°

, where the angles are in terms of radians and $b = \arctan(B/A)$. Then, we can find α as

$$\hat{\alpha} = \arccos\left(\frac{C}{\sqrt{A^2 + B^2}}\right) - \arctan\left(\frac{B}{A}\right) \quad (2.2)$$

2.2. MATRIX THEORY

Covariance matrices derived from road lines are informative structures about 3D road structure. To extract this information we use some properties of the covariance matrix [72].

2.2.1. Covariance Matrix and Its Properties

Let $\mathbf{x}_{N \times 1}$ be a random column vector. Its covariance matrix ($\Sigma_{N \times N}$) is defined as

$$\Sigma = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x}^T - E[\mathbf{x}^T])] \quad (2.3)$$

, where $E[.]$ is expectation operator in random processes and $.^T$ is transpose of a matrix. Some related properties of covariance matrix are itemized below:

- Covariance matrix (Σ) is symmetric.
- Covariance matrix is semi-positive definite such that $\mathbf{x}^T \Sigma \mathbf{x} \geq 0$.
- Its eigenvalues are greater or equal to 0.

2.2.2. Column Space (CS) and Null Space (NS) of a Matrix

Let's consider again a covariance matrix, whose size is $N \times N$. Column space is the span of all columns in the matrix. Column space can be named as range in matrix terminology. Column space is spanned in other words linear combination of n independent columns. Let's say size of CS is $n = \dim(\mathbf{C}_s) \leq N$, where n is the number of columns in CS. n is less than or equal to total number columns, which is $N = n + m$.

The remaining columns constitute $(N \times m)$ Null Space ($\mathbf{N}_s$) such that a test vector ($\mathbf{x}_t$) in CS is perpendicular to NS

$$\mathbf{x}_t^T \mathbf{N}_s = 0 \quad (2.4)$$

2.2.3. Eigenvalues, Eigenvectors and Principal Component Analysis (PCA)

There are N eigenvalues and N corresponding eigenvectors of covariance matrix. PCA is to take important eigenvectors based on choosing greatest eigenvalues. Eigenvalues are sorted in descending order such that the first eigenvalue is the greatest λ_{max} and the last eigenvalue is the least λ_{min} . If we set a threshold $\gamma_0 = 10^9$ to choose the first L greatest eigenvalues

$$\lambda_i > \lambda_{max} / \gamma_0 \quad (2.5)$$

, the corresponding eigenvectors $\mathbf{v}_i$ construct $(N \times L)$ $\mathbf{W}_{PCA}$ matrix. PCA uses to remove noise components from actual informative data and lets the size of data or co variance matrix

reduces to L ($N > L$) after projecting on PCA domain.

2.2.4. Approximate Null Space (ANS)

Eigenvectors of covariance matrix can be used to approximate Null Space, which is called Approximate Null Space (ANS) in the thesis. Eigenvalues are sorted in ascending order such that the first eigenvalue is the smallest λ_{min} and the last one is the greatest eigenvalue λ_{max} . If we set a threshold $\gamma_1 = 10^5$ to choose the first m smallest eigenvalues

$$\lambda_i < \lambda_{max}/\gamma_1 \quad (2.6)$$

, the corresponding eigenvectors $\mathbf{v}_i$ construct $(N \times m)$ $\mathbf{W}_{ANS}$ matrix. Then, ANS can be used to test if a test vector ($\mathbf{x}_t$) belongs to this covariance class.

2.3. SCALE-INVARIANT FEATURE TRANSFORM (SIFT)

Local Features (LF) is an important topic, which tries to find significant interest points (keypoints) and their properties (descriptors) around the keypoints, in computer vision. Keypoints are basically pixel locations in an image and descriptor is vector that describes the corresponding keypoint distinctively. Therefore, distinctive property of descriptor is used for finding almost same keypoint in other image. This is called matching between two keypoints. SIFT [27] is one of the most important LF and it is currently referenced by 26469 times [73].

SIFT is a complex algorithm, which is composed of 4 steps:

1. **Scale-space extrema detection:** Keypoint candidates are found in different scales of the same image. Low pass Gaussian kernels, which are at different scales, are convolved with original image. Resulting images are called scale-space images. Detect maxima and minima in differences (Difference of Gaussians) between successive scale-space images. For integer multiple of scales, scale-space images are downsampled to pass on other octave.

2. **Keypoint localization:** Fit to 3D quadratic function to have better maxima and minima. Reject keypoints on strong edges and on flats using Hessian matrix.
3. **Orientation assignment:** Create histogram of local gradient directions at selected scale for each keypoint. Choose the highest bin as effective orientation angle for the keypoint. This is the step to achieve invariance to image rotation.
4. **Keypoint description:** Up to this point, each keypoint has its own pixel location, scale and orientation. Now create a descriptor, which is basically a vector, that is distinctive and invariant also to view point and illumination. Around 16x16 pixel neighborhood of keypoint is divided to 16 4x4 subregions. A histogram of gradients with 8 bins is found in each subregion. Then, the length of the descriptor vector makes $16 \times 8 = 128$. The vector is normalized to 1 to get illumination invariance.

SIFT provides invariant keypoints against image location, scale and rotation and invariant descriptor vector against illumination, 3D viewpoint. Hence, totally SIFT is highly invariant for possible changes including pixel noise in image domain. SIFT descriptors are efficiently distinctive that provides correct matching between keypoints. Hence, SIFT can be used for object recognition, matching between different images, between images of stereo camera, image reconstruction, image registration, motion tracking.

In our thesis, we use SIFT to find out matching points between consecutive frames. During 2 consecutive frames a vehicle goes a small distance that changes the positions of matched keypoints. That can be used for road parameters estimation.

2.4. MULTILAYER PERCEPTRON (MLP)

Multilayer Perceptron (MLP) is a supervised learning technique in Artificial Neural Network (ANN) [74, 75]. It is a class of feedforward artificial neural network. MLP utilizes backpropagation technique, which is Least Mean Squares (LMS) estimation for weights [76], for training. Perceptron and neuron terms can be used interchangeably. MLP can be easily used for classification and parameter estimation for different types of signals such as image signals [71] or underwater signals [77]. Estimation is called as regression analysis in statistics as well.

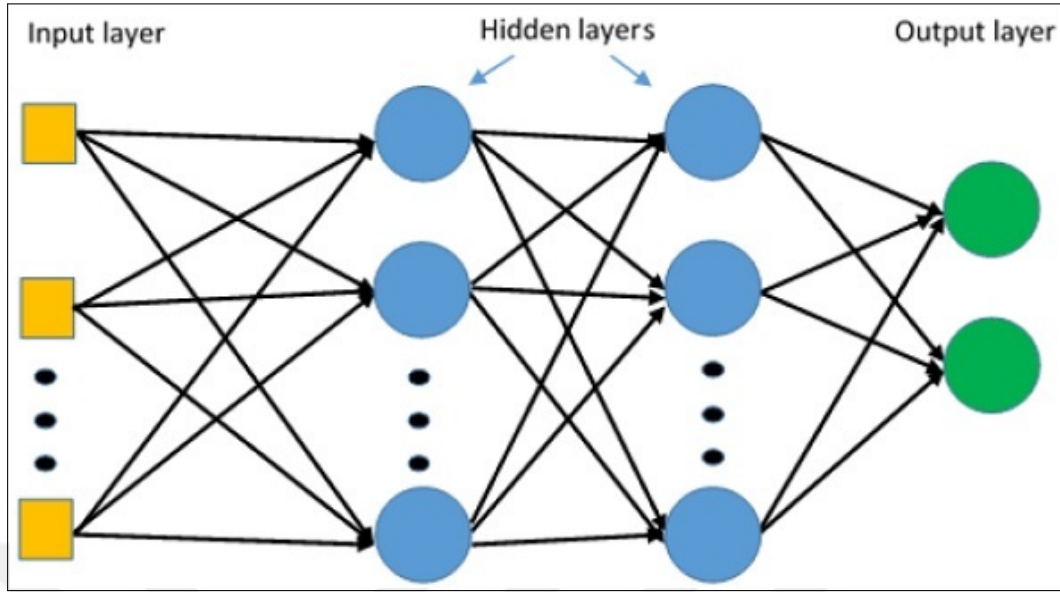


Figure 2.2. MLP with 4 layers

An example network structure for MultiLayer Perceptron (MLP) can be found in Figure 2.2 [78]. The network is composed of four fully connected layers in the figure, which has an input layer and 2 hidden layer and a final layer as output layer. Input layer should have the same size of or more than the size of input vector, which MLP operates on. Output layer should have the size of number of parameters to be estimated. Briefly the structure in the figure can be given as $\text{MLP}(L1, L2, L3, L4=2)$, where $L1$, $L2$, $L3$, and $L4$ represent number of neurons in 4 layers.

Lateral (u -axis) and vertical (v -axis) location of pixels on both road lines are concatenated to be training and test input vectors. MLP estimation is realized with TensorFlow 2.1.0 and Keras 2.2.4 libraries in Python3.7. In the hidden layers *Rectified Linear Unit (ReLU)* activation is used

$$f_i(s) = \begin{cases} s, & s \geq 0 \\ 0, & s < 0 \end{cases} \quad (2.7)$$

, for i th neuron in layer l . s is summation of multiplication of weights $w_{j,i}^{l-1,l}$ and neurons' outputs f_j in previous layer $l - 1$. Therefore, $f_i(x)$ is the current output of i th neuron in layer l . In the output layer *Linear* activation is used for estimations. Activation function can be called as transfer function as well.

We give a summary in Table 2.1 for different type of problems and their necessary parameters in MLP. Loss functions, which are Categorical Cross Entropy (CCE) and Mean Square Error (MSE), are used in back propagation. Classification results of road slopes is given in Section 8.1 and estimation results of road parameters is given in Section 8.2, which is improvement of classification of discretized parameter values.

Table 2.1. Summary of activation energy and loss function

Problem Type	Output Type	Output Activation Function	Loss Function
Classification	Multiple classes	Softmax	Categorical Cross Entropy
Estimation	Real number	Linear	Mean Square Error

There are some other definitions in training time: An *epoch* describes the number of times the algorithm sees the entire data set. Basically one epoch defines one forward pass and one backward pass of all the training examples. *Batch size* is about grouping the train vectors and it defines number of vectors in a batch. A forward and backward pass of one batch makes one *iteration*.

2.5. SIGNAL-TO-NOISE RATIO (SNR)

SIGNAL-TO-NOISE RATIO (SNR) is important metric to show the behaviour of any system to different noise levels. The metric is the ratio of signal power to noise power. Road lines in an image is our signal ($\mathbf{x}$), which is a $(N \times 1)$ vector concatenated of u-axis (lateral) and v-axis (vertical) pixel locations of left and right lines. If we denote the corresponding noise samples as $\mathbf{w}$

$$SNR = \frac{\sum_{n=1}^N \mathbf{x}^2[n]}{\sum_{n=1}^N \mathbf{w}^2[n]} \quad (2.8)$$

The SNR in decibel (dB) is defined as

$$SNR_{dB} = 10 \log_{10}(SNR) \quad (2.9)$$

It is logical to assume $\mathbf{w}$ as Additive White (Independent) Gaussian Noise (AWGN). Then power of noise becomes sample variance of the noise

$$\sigma^2 = \frac{\sum_{n=1}^N \mathbf{w}^2[n]}{N} \quad (2.10)$$

, which makes easy in simulation to add noise on signal.

3. SYSTEM MODEL

We define our problem here and give the basic definitions for our model. A host vehicle, which can be a car, truck or bus, has a camera installed in the middle of lateral length of windshield as seen in red box in Figure 3.1. The camera position is counted as our origin $O_c(X = 0, Y = 0, Z = 0)$ as Camera Reference Frame (CRF) and camera's view is parallel to the ground and its optical axis is parallel to the driving direction. O_c is the aperture of the camera. Our Road Model (RM), which is a mathematical representation of 3D road, is constructed according to the CRF. X is lateral axis with right side positive direction, Y is vertical axis with up-side positive direction and Z is longitudinal axis with front side positive direction, which is the same direction of the motion of host car. The measures on the Z-axis can be called as Range or (Longitudinal) Distance.



Figure 3.1. Front camera installed on a car

3.1. BASIC ROAD MODEL (BRM)

On any road we have 3 different angles as shown in Figure 3.2. The lateral angle (or slope) α measures how much the road is banked horizontally with respect to the earth surface. In other words it is a *Roll* measure of the road. The longitudinal angle β measures how much the road is uphill (or downhill). In other words it is a *Pitch* measure of the road. Finally, the *Yaw* angle θ is the measure of curvature of the curves on the road. Basically, θ is the angle can be seen on a map for the curves. Totally there are 5 unknown road parameters

$$\mathbf{p} = [d_l, d_r, \alpha, \theta, \beta] \quad (3.1)$$

, where d_l, d_r are the distances to the left and right lines respectively. Lane width is defined as $d = d_l + d_r$. In basic road model, we study estimating the lateral angle α and longitudinal angle β .

Range (or depth or distance) is defined as the radial distance from the sensor to the objects. Range is a very important parameter to be used in slope estimations. The radar and lidar sensors are used to find the exact range information. On the other hand, since radars are comparably expensive sensors, only camera based solutions for range information and object detection became popular [79].

3.1.1. Basic Road Model for Straight Roads

For a straight road we can give

$$\begin{aligned} P_{l1} &= (X_{l1}, Y_{l1}, Z_{l1}) = [-d_l \cos(\alpha), -d_l \sin(\alpha) - Y_c, Z_{l1}] \\ P_{l2} &= (X_{l2}, Y_{l2}, Z_{l2}) = [-d_l \cos(\alpha), -d_l \sin(\alpha) - Y_c, Z_{l2}] \\ P_{r1} &= (X_{r1}, Y_{r1}, Z_{r1}) = [d_r \cos(\alpha), d_r \sin(\alpha) - Y_c, Z_{r1}] \\ P_{r2} &= (X_{r2}, Y_{r2}, Z_{r2}) = [d_r \cos(\alpha), d_r \sin(\alpha) - Y_c, Z_{r2}] \end{aligned} \quad (3.2)$$

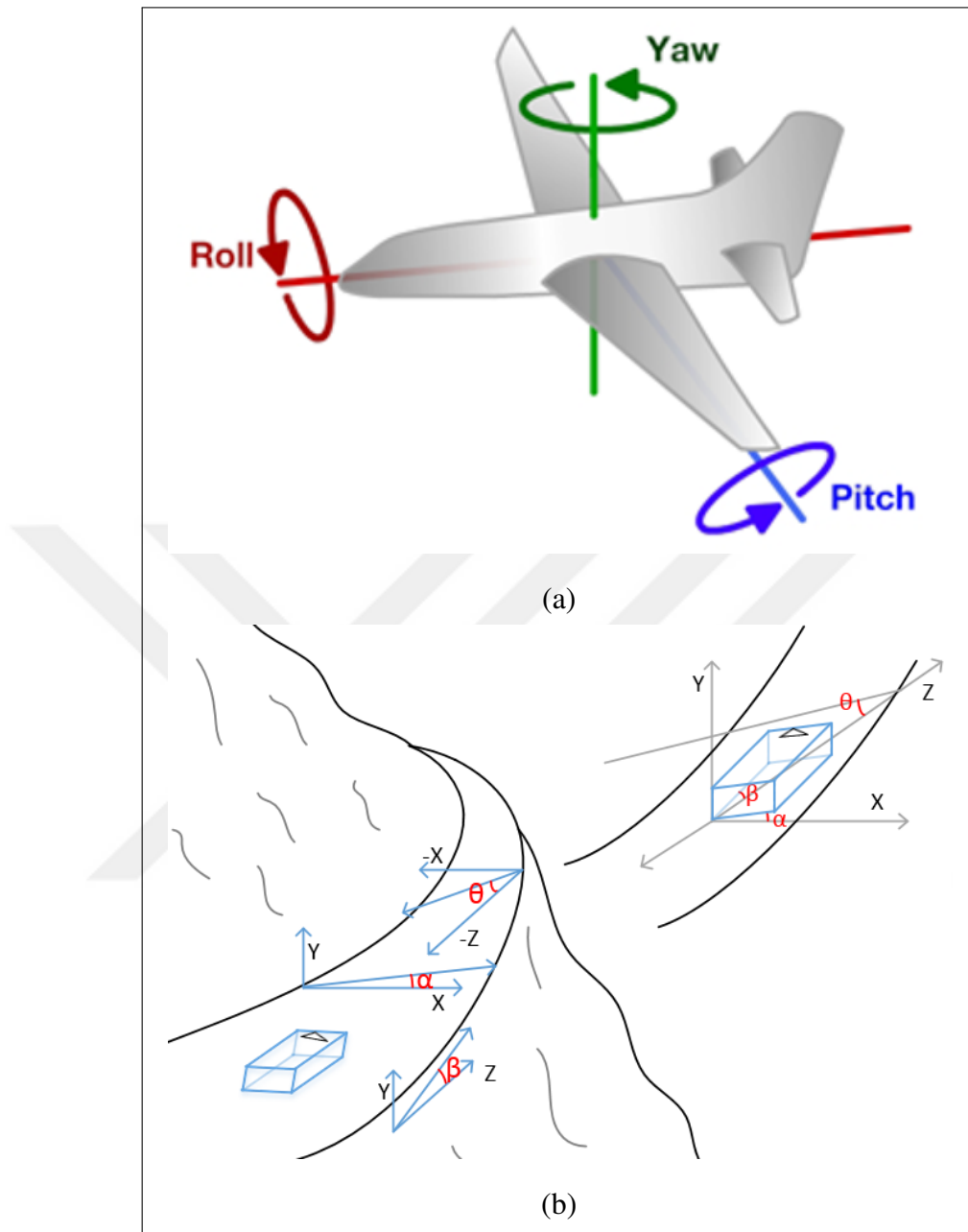


Figure 3.2. 3D angles. (a) Roll, pitch, yaw angles, (b) Angles for a vehicle.

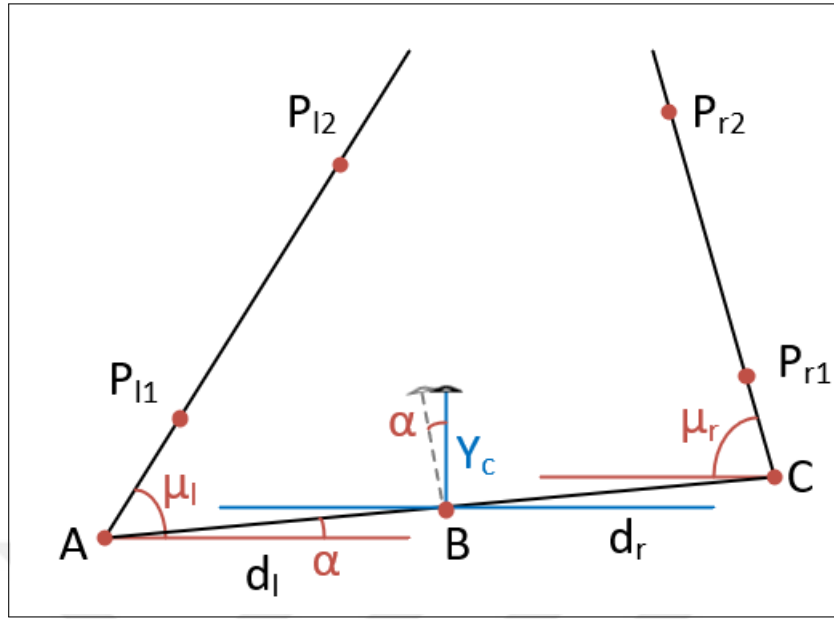


Figure 3.3. Straight banked road

, where $|AB| = d_l$ is the distance to left line and $|BC| = d_r$ to the right line from the camera's projection on the road. The distance between lines is called lane width (d), which is 3.5m on German highways. The representations of the equations in (3.2) can be found in Figure 3.3, where the camera is located on point B.

3.1.2. Basic Road Model for Circular Roads

Without losing the generality we assume that the road is turning left as a circular curve (with radius R), which is banked with a slope α . In Figure 3.4, we have a typical CarMaker¹ [60] image for $\alpha = 0^\circ$ and its corresponding road model is depicted as blue lines. The red lines are for $\alpha = 10^\circ$ from our road model. The bird's eye view is depicted in Figure 3.5 to derive the parametric equations for our road model. Let's assume we have 2 points in 3D as $P_l = (X_l, Y_l, Z_l)$ and $P_r = (X_r, Y_r, Z_r)$. θ is the yaw angle between horizontal axis and tangential passing through P_l . And θ^c is the complementary angle such that $\theta = \pi/2 - \theta^c$. The angle θ , which runs from 90° to 0° , is to define a circular curve in polar coordinates. If the road is straight, in other words the steering wheel is at its default position, yaw angle is accepted as 90° in our system. The elements of P_l and P_r are

¹E. Ustunel was previously working at Adam Opel AG, Germany. CarMaker images used in this study were collected when E. Ustunel was working there.

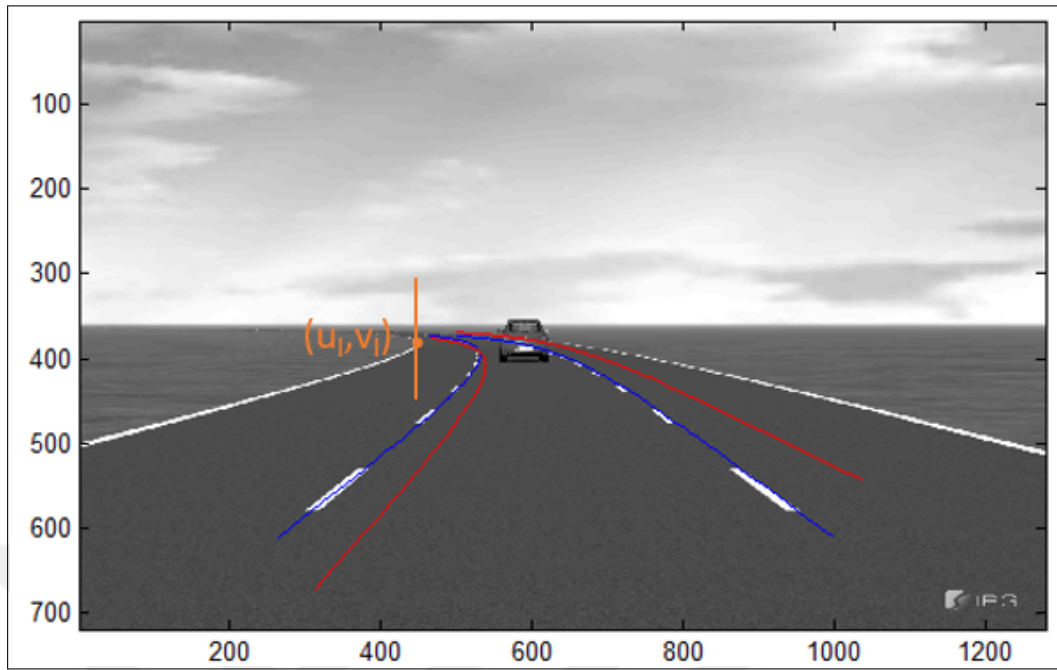


Figure 3.4. CarMaker image and basic road model with $R=1000m$

$$X_l = -d_l \cos(\alpha) - (R - R \sin(\theta))$$

$$X_r = X_l + d \cos(\alpha) \sin(\theta)$$

$$Y_l = -Y_c - d_l \sin(\alpha)$$

(3.3)

$$Y_r = -Y_c + d_r \sin(\alpha)$$

$$Z_l = R \cos(\theta)$$

$$Z_r = Z_l + d \cos(\alpha) \cos(\theta)$$

This road model complies with IPG's CarMaker and it should be known that it is possible to create slightly different road models based on reference point as long as all the pixel on the lines have same relation.

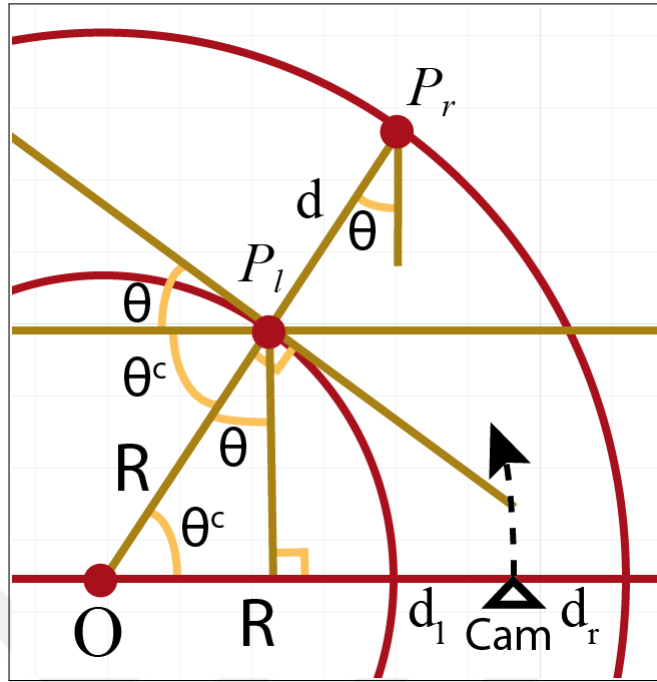


Figure 3.5. Road model as a bird's eye view

3.2. CLOTHOID ROAD MODEL (CRM)

Clothoid Road Model (CRM), which is given in [50, 53], is more generic than Basic RM. CRM reflects real road structure into a mathematical model successfully. Therefore, it is our preferred model. CRM describes the road lines in 3D

$$X_l = -d_l \cos \alpha + a_h Z_l + b_h Z_l^2$$

$$Y_l = d_l \sin \alpha - Y_c + a_v Z_l + b_v Z_l^2$$

(3.4)

$$X_r = d_r \cos \alpha + a_h Z_r + b_h Z_r^2$$

$$Y_r = -d_r \sin \alpha - Y_c + a_v Z_r + b_v Z_r^2$$

, where Y_c is the known height of camera from road surface and it is accepted 1.2m in the thesis. Figure 3.6 explains above equations in (3.4) visually. A vehicle takes a left turn. Left and right lines are shown as blue and red respectively. Figure 3.6a is a bird eye view

on Z-X plane and Figure 3.6b is side view on Z-Y plane. d_l and d_r are the distances to left (l) line and right (r) line respectively as shown in Figure 3.7a. The lane width d is found as $d = d_l + d_r$. Z is unknown longitudinal distance between the pixel on road line and the camera. Distance to left and right line are $d_l = d_r = 2m$ and lateral angles is $\alpha = -2^\circ$ in Figure 3.7a. Moreover, $a_h = -0.001, b_h = -0.001$ are horizontal parameters and similarly $a_v = 0.001, b_v = 0.001$ are vertical parameters. a_h includes ego vehicle's orientation with respect to the road as well. Totally there are 7 unknown road parameters

$$\mathbf{p} = [d_l, d_r, \alpha, a_h, b_h, a_v, b_v] \quad (3.5)$$

The lateral angle (or slope) α measures how much the road is banked horizontally with respect to the earth surface as shown in Figure 3.7a. For a left turn, lateral angle (α) is accepted as negative and similarly for a right turn, lateral angle (α) is accepted as positive. Since it is a left turn, right (red) line is higher than left (blue) line in Figure 3.6b.

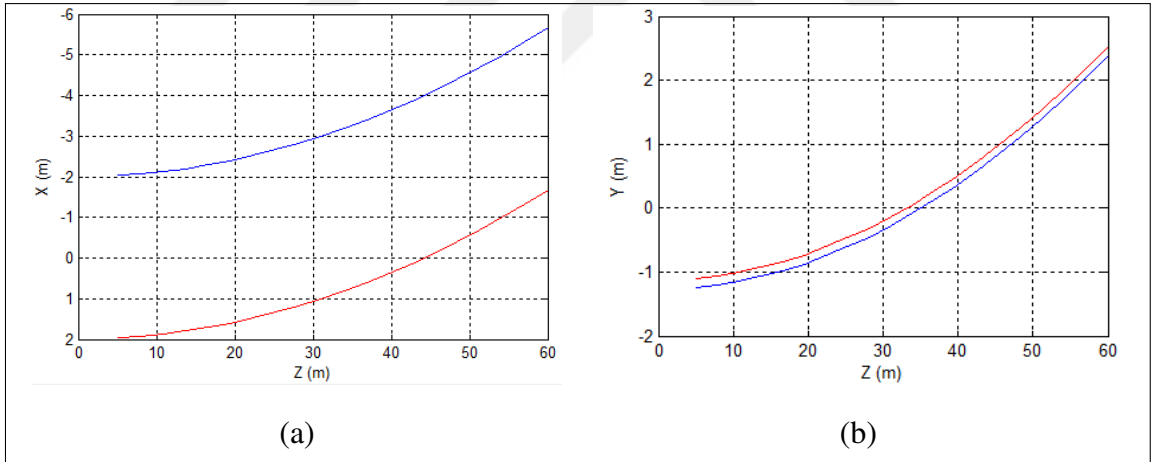


Figure 3.6. 3D road profile. (a) Horizontal profile, (b) Vertical profile.

3.3. BASIC RM (BRM) VS CLOTHOID RM (CRM)

BRM and CRM has 3 common parameters, which are d_l, d_r and lateral slope α . Yaw angle (θ) in BRM corresponds to horizontal parameters in CRM those are a_h, b_h . Similarly, longitudinal angle (β) in BRM corresponds to vertical parameters in CRM those are a_v, b_v . Note that yaw angle or horizontal parameters measures curvature of a road on the earth surface. As seen from the equations, CRM is extended version of BRM such that CRM

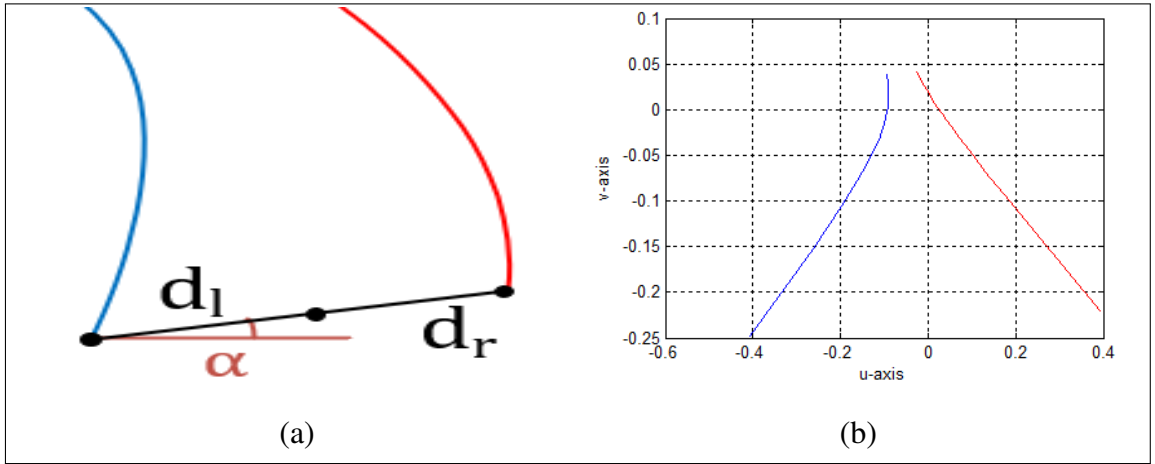


Figure 3.7. 3D road profile. (a) Lateral slope (α), (b) Projected 2D view.

covers more shapes in 3D. In fact, BRM is only valid for completely straight roads or circular roads.

3.4. PINHOLE CAMERA (PHC) MODEL

Pinhole Camera (PHC) model is applicable to find range information, when the object is far away from the camera lens [80]. In this case, we don't need to use basic lens equations, instead we can only use basic PHC equations. An image is projected from 3D environment and the image axes are called u -axis for horizontal axis and v -axis for vertical axis as a regular 2D Cartesian coordinates in Figure 3.8. O_c is the optical center, which is the origin of camera's coordinate system. R is the principal point, which is the origin of image, corresponding to O_c .

A point in 3D environment can be given as $P_i = (X_i, Y_i, Z_i)$ for i -th point. The corresponding 2D point in the image is defined as $p_i = (u_i, v_i)$, which is found by projection from P_i by using PHC model as also similarly mentioned in [46]

$$u_i = fX_i/Z_i$$

(3.6)

$$v_i = fY_i/Z_i$$

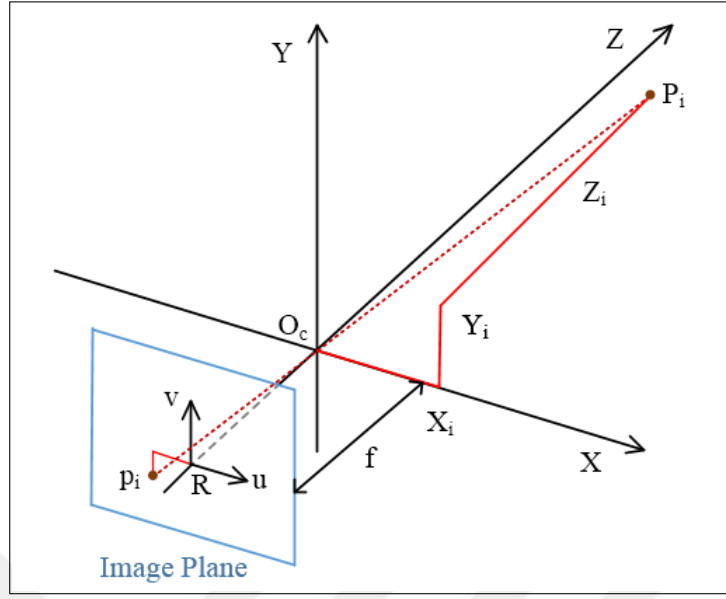


Figure 3.8. Pinhole camera (PHC) model

where f is the equivalent focal length including conversion from meter to pixel width. We assume that camera is calibrated, distortion is removed, and image center (principal point) is assumed to be in the center of image i.e. principal point offset is zero. Then, p_i is the pixel location in an image, and u_i is the point on u-axis (horizontal) and v_i is the point on v-axis (vertical). Projected image of 3D road can be seen in Figure 3.7b.

3.5. EQUIVALENT FOCAL LENGTH CALCULATION

Equivalent focal length (f) calculation is our simple calibration method using longitudinal distance (Z) and lane widths both in 3D and on 2D image. Distance calculation from host vehicle to a point on the road line is necessary for slope estimations. Distance and focal length are correlated variables and coexist in (3.6). To find the equivalent focal length (f), which is in terms of pixels, we use similar triangles in Figure 3.9 with respect to aperture of the camera (O_c):

$$\frac{|P_l P_r|}{|p_l p_r|} = \frac{Z}{f} \quad (3.7)$$

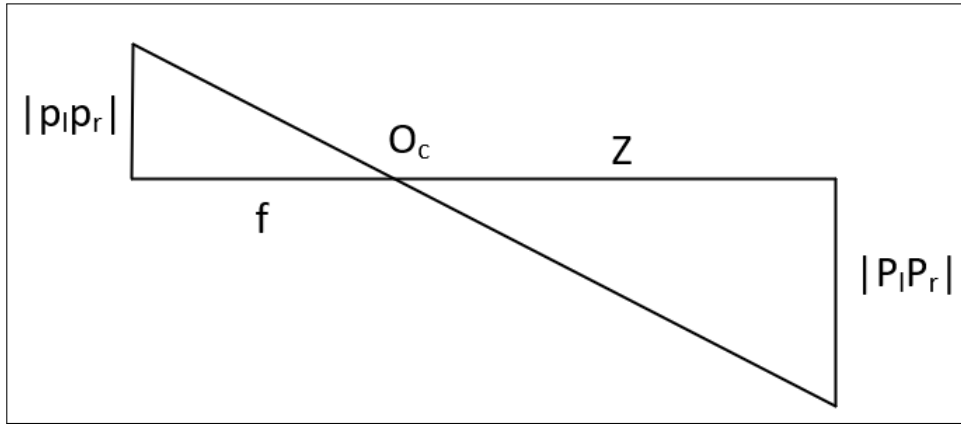


Figure 3.9. Similar triangles to find equivalent focal length (f)

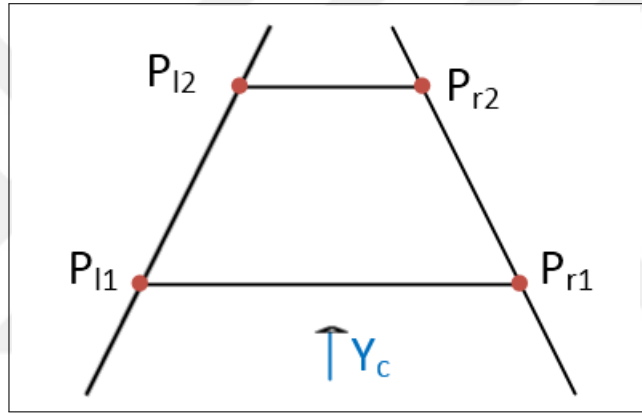


Figure 3.10. A straight road with left and right lines

, where a flat road is used. $d = |P_l P_r|$ is the width of a lane in 3D in Figure 3.9 and Figure 3.10. It is a dynamic parameter, whose value is 3.5m in Germany, and can be arranged for other countries. $|p_l p_r|$ is the corresponding distance on 2D image in terms of pixels in Figure 3.9. Z is the longitudinal distance to the lateral line $|P_l P_r|$.

The equivalent focal length is initially found once by using known distance information in training step and kept fixed as long as camera and camera position are not changed. This fixed focal length f then can be used for range information Z . Equivalent (simplified) focal length f is the output of camera calibration (3.7). The found f values are 1980 pixel for CarMaker images and 950 pixel for Samsung Galaxy S6 images at 1280x720 HD resolution.

4. GEOMETRY BASED (GB) SLOPE ESTIMATION

Geometric Based (GB) solutions use Basic RM (BRM) that assumes exact straight roads and exact circles on the road. Since GB uses simplicity of BRM, it can derive closed form expressions. Therefore, GB method does not work under complex road structure of Clothoid RM (CRM).

GB slope estimation finds intrinsic properties of the road image those are inherited from the actual 3D topography of the road. These intrinsic properties should reveal about the slopes and they may have some invariant properties, which can be derivatives, ratios or widths possibly being independent of distance, focal length or yaw angle. We extract these properties for slope estimations without going through complex algorithms. Geometry based solutions used Basic RM (BRM).

4.1. GB LATERAL SLOPE ESTIMATION ON STRAIGHT BANKED ROADS

Without losing the generality of the problem, the road is first assumed as straight longitudinally. That means, there isn't any curve on the road. This information can easily be extracted from GPS module. The lateral slope can be named as banked road in other references. The road is assumed to have a slope as being inclined from left to right. That is, right hand side of the road is higher than the left hand side. The similar logic also applies to the vice versa situation.

We get 2 points on the left line of the road and 2 on the right line in Figure 3.3. There is clear difference between Figure 3.10 and Figure 3.3 that implies nonbanked road and banked road are separable due to the roll slope. This kind of slope estimation tries to benefit from how road lines appear in image and how the slope embedded in the road lines. As shown in Figure 3.3, the road is inclined by α , and the host vehicle is assumed just before the inclined road. The horizontal and vertical blue line shows host car's current orientation that is assumed parallel to earth surface. After that point it will experience slope change and camera rotates α as counter clock wise seen as dotted-faded black line. As a good remark to

observe, the 2D view of the road will be changed from Figure 3.3 to Figure 3.10 as passing through the slope. Since we want to find slope changes in other words differential values, any initial car orientation assumption is logical.

The corresponding projections points in the image are given as

$$\begin{aligned}
 p_{l1} &= (u_{l1}, v_{l1}) = [fX_l/Z_{l1}, fY_l/Z_{l1}] \\
 p_{l2} &= (u_{l2}, v_{l2}) = [fX_l/Z_{l2}, fY_l/Z_{l2}] \\
 p_{r1} &= (u_{r1}, v_{r1}) = [fX_r/Z_{r1}, fY_r/Z_{r1}] \\
 p_{r2} &= (u_{r2}, v_{r2}) = [fX_r/Z_{r2}, fY_r/Z_{r2}]
 \end{aligned} \tag{4.1}$$

Then, if we take the tangent of left line:

$$\begin{aligned}
 T_l &= \tan(\mu_l) = \frac{v_{l2} - v_{l1}}{u_{l2} - u_{l1}} = \frac{Y_l}{X_l} \\
 &= \frac{Y_c + d_l \sin(\alpha)}{d_l \cos(\alpha)}
 \end{aligned} \tag{4.2}$$

By applying same operations on the right line, we find

$$\begin{aligned}
 T_r &= \tan(\mu_r) = \frac{v_{r2} - v_{r1}}{u_{r2} - u_{r1}} = \frac{Y_r}{X_r} \\
 &= \frac{Y_c - d_r \sin(\alpha)}{d_r \cos(\alpha)}
 \end{aligned} \tag{4.3}$$

where μ_l and μ_r are between $0^\circ \leq \mu_r \leq 90^\circ$. The only requirement for α slope estimation is to know d_l and d_r , which is not a difficult task since closest line pixels to the car can be used. As seen, focal length (f), and range (Z) parameters cancel out, that brings out invariances. That is very important and convenience to get slope information of a road without requiring the focal length and longitudinal distance.

We solve for α using (4.2) and (4.3)

$$\hat{\alpha} = \arcsin \left(\frac{Y_c(T_l d_l - T_r d_r)}{(T_r + T_l) d_l d_r} \right) \quad (4.4)$$

If $T_l < T_r$, then $\alpha < 0$ that is left hand side of the road is lower than right. If $T_l > T_r$, then $\alpha > 0$ that is left hand side of the road is higher than right.

4.2. GB LATERAL SLOPE ESTIMATION ON CURVES

3D points on a curve are given in (3.3). Their corresponding 2D projections are

$$\begin{aligned} p_l &= (u_l, v_l) = [fX_l/Z_l, fY_l/Z_l] \\ p_r &= (u_r, v_r) = [fX_r/Z_r, fY_r/Z_r] \end{aligned} \quad (4.5)$$

We try to find the slope between corresponding two points on left and right lines on the image to find information about α

$$\begin{aligned} \tan(\tilde{\alpha}) &= \frac{v_r - v_l}{u_r - u_l} = \frac{fY_r/Z_r - fY_l/Z_l}{fX_r/Z_r - fX_l/Z_l} \\ &= \frac{R \sin(\alpha) + Y_c \cos(\alpha) + \sin(\alpha) \cos(\alpha)/2}{R \cos \alpha + d_l \cos^2(\alpha)} \\ &\cong \frac{R \sin(\alpha)}{R \cos(\alpha)} = \tan(\alpha) \end{aligned} \quad (4.6)$$

where $\tilde{\alpha}$ is the corresponding angle on an image of actual lateral slope α . One of the important results of (4.6) is that it doesn't depend on θ . Essentially, if R is big enough, we don't need all other terms, since they cancel out in (4.6). That clearly says that lateral slope on image is almost equal to the lateral slope in the real 3D environment. The only

remaining challenge in this modeling is finding the corresponding point p_r , which is on the right line, of the point p_l on the left line.

Assuming the longitudinal distance of P_l to the camera is known, we can find the location of P_r , since the pixel distance between P_l and P_r is unique as an invariant. It is shown in Appendix that, pixel distance between pairs is invariant to lateral slope and a 1-1 function over longitudinal distance. By drawing an arc, which is centered on p_l , we can find corresponding p_r , where arc and right line intersects. Although this approach doesn't require θ , finding the intersection requires numeric computations.

The information about α in (4.6) is clear and easy to understand, but it brings out several challenges as mentioned above. To overcome those, we can extract the information for α on the tangent of left line. In (3.3), the parameter θ is the running variable. Therefore the tangent of the left line can be defined as

$$\begin{aligned} \frac{dv_l}{du_l} &= \frac{dv_l/d\theta}{du_l/d\theta} \\ &= \frac{(Y_c + d_l \sin(\alpha)) \sin(\theta)}{(R + d_l \cos(\alpha)) \sin(\theta) - R} \end{aligned} \tag{4.7}$$

, where θ changes $\theta = \pi/2 : 0$. Although derivative in (4.7) provides invariance for range Z_l , it depends on R and θ , which are unknown and need to be found. However, if we consider the left line where it is vertical i.e. $dv/du \cong \infty$, we can find a relation between R and θ , when denominator goes to zero

$$\sin(\theta) = \frac{R}{R + d_l \cos(\alpha)} \tag{4.8}$$

Let's have a ratio of v_l/u_l for the pixel seen in Figure 3.4 to cancel out focal length f and longitudinal distance Z_l

$$\begin{aligned} m_l &= \frac{v_l}{u_l} = \frac{Y_l}{X_l} = \frac{Y_c + d_l \sin(\alpha)}{d_l \cos(\alpha) + R - R \sin(\theta)} \\ &= \frac{(Y_c + d_l \sin(\alpha))(R + d_l \cos(\alpha))}{d_l \cos(\alpha)(2R + d_l \cos(\alpha))} \end{aligned} \quad (4.9)$$

On the second line of (4.9), we used the equality of $\sin(\theta)$ in (4.8). If we assume the approximation

$$\frac{R + d_l \cos(\alpha)}{2R + d_l \cos(\alpha)} \cong \frac{R}{2R} = 0.5 \quad (4.10)$$

, then (4.9) turns into

$$m_l = \frac{v_l}{u_l} = \frac{Y_l}{X_l} = \frac{Y_c + d_l \sin(\alpha)}{2d_l \cos(\alpha)} \quad (4.11)$$

which doesn't depend on R and θ that makes estimation of slope very easy. The only thing we must find is to find a pixel at left line, at which the slope goes to infinity. Then we take the ratio of ordinates of that pixel as $m_l = v_l/u_l$. If we proceed in the algebra, we get

$$Y_c = 2d_l m_l \cos(\alpha) - d_l \sin(\alpha) \quad (4.12)$$

, and converting it as $C = A \cos(\alpha) + B \sin(-\alpha)$ suggested in Section 2.1.2. Then, we can find α as

$$\hat{\alpha} = \arccos\left(\frac{C}{\sqrt{A^2 + B^2}}\right) - \arctan\left(\frac{B}{A}\right) \quad (4.13)$$

where the angles are in terms of radians. Also, $C = Y_c$, $A = 2d_l m_l$, and $B = d_l$ in the first 2 lines of (4.12). The beauty of the (4.13) is that it requires the location of only one pixel in an image. That pixel is located at the vertical tangent point on left line in (4.9). The slope estimation is found analytically that decreases the complexity very much and it needs detection of only left line.

As for a discussion, for $d = 3.5m$ and $d_l = d/2 = 1.75m$ we get actual values for the approximation in (4.10) in Table 4.1. This shows that the values are very close to 0.5 as a valid approximation, which helps to give good estimation of lateral slope. The worst value as an error is $\varepsilon = 0.00434$ at $R=100m$. Actually, most of the time on highways, radius of a curve is larger than 500m in [81] to allow high cruising speeds.

Table 4.1. Actual values for (4.10)

α	R=100m	R=500m	R=1000m	R=1500m	R=2000m
0°	0.50434	0.50087	0.50044	0.50029	0.50022
10°	0.50427	0.50086	0.50043	0.50029	0.50021

4.3. GB LONGITUDINAL SLOPE ESTIMATION ON STRAIGHT ROADS

We try intuitively to find useful geometric properties of the road that give an information about road slope. We derive our algorithm for an uphill scenario in Figure 4.1. The slope changes around point P_a . We can also find the distances Z_a and Z_b ($Z_{tot} = Z_b + Z_a$) by using the lane width and (3.7).

In the uphill region, the vertical axis on any road line would be

$$v_b = \frac{f(Z_b \tan(\beta) - Y_c)}{Z_{tot}} \quad (4.14)$$

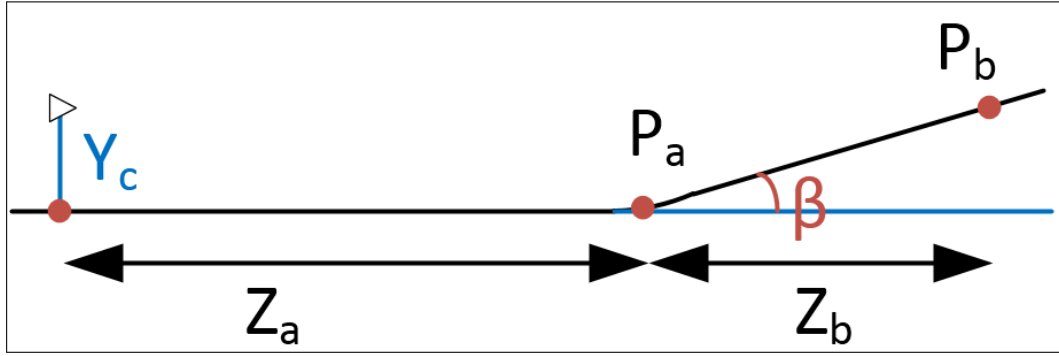


Figure 4.1. An uphill road starts after plain (side view)

, where v_b is measured on the image. The only unknown term is $\tan(\beta)$ and can be found as

$$\tan(\beta) = \frac{v_b Z_{tot}/f + Y_c}{Z_b} \quad (4.15)$$

from which longitudinal angle (β) can be found easily.

We generate a different and basic algorithm for downhill road in Figure 4.2. The idea is that when the foreseeable distance Z is at its minimum, tangent of the downhill slope becomes equal the ratio of camera height (Y_c) by Z_{min} .

$$\tan(\beta) = Y_c/Z_{min} \quad (4.16)$$

We track road lines and estimate the maximum foreseeable longitudinal distance using (3.7), where $|p_l p_r|$ is already updated in the image. We calculate the slope (β) in (4.16) when the maximum longitudinal distance becomes minimum. GPS module can be used here to understand that the road is straight.

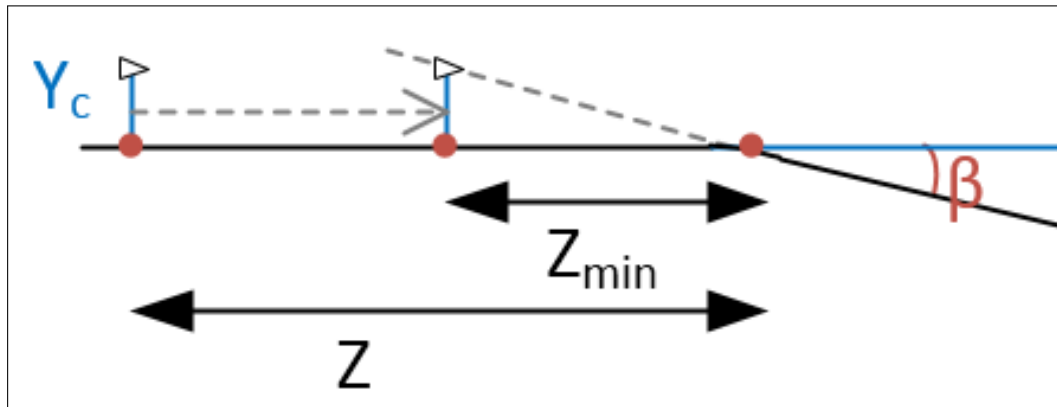


Figure 4.2. A downhill road starts after plain (side view)

4.4. GB LONGITUDINAL SLOPE ESTIMATION ON CURVES

We approximate longitudinal slope estimation on curves by using the same method in previous Section 4.3 . The approximation is using the pixel distance on u -axis on the curve to find lane width. For big radius (R) values, we get satisfactory outputs in Results section without an extra need for geometric calculations.

5. COVARIANCE BASED (CB) SLOPE ESTIMATION

Since Covariance Based (CB) slope estimation does not give closed form, which depends on using mathematical model of road parameters, it can work for both Basic RM (BRM) and Clothoid (CRM).

CB classification is a supervised learning technique, which learns from known road lines to create its training information set. The algorithm is also called as Principal Component Null Space Analysis (PCNSA), which is a powerful tool to classify many objects in a test dataset [68, 69, 70]. In this work, we use this method, which was originally designed for classification problems, for road slope estimation. The idea is that if we use enough number of bins in a range of parameter, which can be $0^\circ \leq \alpha \leq 10^\circ$, we can approximate close enough to the actual slope. We call these bins as classes in our research. The bin size can be 1° , which can have enough resolution for the angle. After getting the road lines on the left and right, we first create a covariance matrix, which has big dimensions. However, Principal Component Analysis (PCA) decreases the dimensions and extract important vectors from the covariance matrix. We train the algorithm with different values of lateral and longitudinal slope as a supervised learning. Covariance Based method estimates road slopes before vehicle enters the slope change as also seen Figure 5.1. Therefore, it has benefit of estimation road slope beforehand.

Covariance matrix is generated by choosing a pixel on e.g. right line, and the finding the correlation of this pixel to all pixels on both left and right line. By this way, we include all the information e.g. slopes, radius (R) of a curve. After training the algorithm, we estimate in testing step how much the query is close to one of the slopes, which are kept in bins. Assuming we have $N/4$ samples on left and right lines in Figure 5.1, where each sample $p = (u, v)$ is composed of horizontal (u) and vertical (v) components, we construct

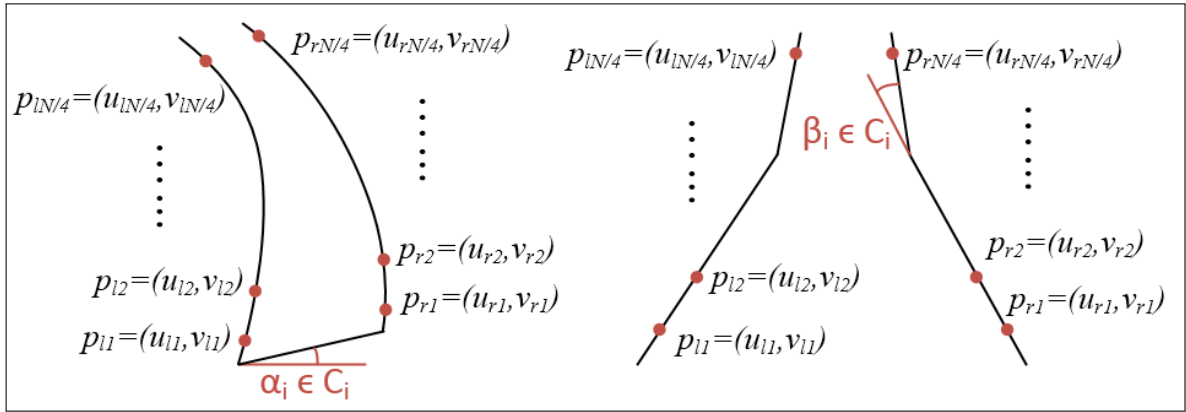


Figure 5.1. Generation of Y_N vector for lateral and longitudinal slope

N-element column vector $\mathbf{x}_{N \times 1}$

$$\begin{aligned} \mathbf{x} &= [p_{l1}, p_{l2}, p_{l3}, \dots, p_{lN/4}, p_{r1}, p_{r2}, p_{r3}, \dots, p_{rN/4}]^T \\ &= [(u_{l1}, v_{l1}), (u_{l2}, v_{l2}), (u_{l3}, v_{l3}), \dots, (u_{rN/4}, v_{rN/4})]^T \end{aligned} \quad (5.1)$$

We have different line types depending on lateral slope, longitudinal slope and curvature of the road. These line types are used in training process. The $N \times N$ covariance matrix for a class (C_i) can be found as

$$\Sigma_i = \sum_{k=1}^K [\mathbf{x}_i(k) - \boldsymbol{\mu}_i][\mathbf{x}_i(k) - \boldsymbol{\mu}_i]^T \quad (5.2)$$

where K is the number of trials in training step for each class C_i . C_i is the i -th bin (or class) after the parameter range is divided by total number of classes (C). That means each parameter in a road model will have C bins namely classes. $\mathbf{x}_i(k)$ is $N \times 1$ vector found in (5.1) for each class C_i and for each trial k in that class. By changing the road parameters, we can have variety, which constructs the K number of different trials. For a lateral slope, we can change the distance to the left line, curve radius (R), left and right lines' arc lengths on the curve. $\boldsymbol{\mu}_i$ is N dimensional column vector as a mean of i -th class that can be found by averaging the trials in that class.

Left curve, right curve, straight, uphill, and downhill as upper classes can be detected by using GPS or another PCNSA algorithm or other techniques as a preprocessing step. Then slope estimation step follows. The importance and usage of PCNSA in this work is its ability to estimate the slope values in an upper class, which is assumed already found in preprocessing step. This estimation is basically to find the closest bin in the range of values of the slope.

Let's assume the total number of varieties is CK , and total covariance matrix can be written as

$$\Sigma = \frac{1}{CK} \sum_{i=1}^C \sum_{k=1}^K [\mathbf{x}_i(k) - \boldsymbol{\mu}][\mathbf{x}_i(k) - \boldsymbol{\mu}]^T \quad (5.3)$$

$\boldsymbol{\mu}$ is average of in-class mean vectors $\boldsymbol{\mu}_i$. Since N is a big number, using Principal Component Analysis (PCA) on $\Sigma_{N \times N}$, we can decrease the dimensions to $L \ll N$. $\mathbf{W}_{PCA}$ is a $N \times L$ projection matrix, whose columns are the L eigenvectors of the matrix $\Sigma_{N \times N}$. Those L eigenvectors correspond to L biggest eigenvalues of the matrix. In our experiments we typically choose $L = 32$. Decreasing dimension of queries and covariance matrix is done by converting those onto the PCA space. PCA also helps us to separate the noisy ones from the informative ones. PCA projection is done as

$$\mathbf{y}_L = \mathbf{W}_{PCA}^T (\mathbf{x}_N - \boldsymbol{\mu}_N)$$

$$\boldsymbol{\mu}_{i,L} = \mathbf{W}_{PCA}^T (\boldsymbol{\mu}_{i,N} - \boldsymbol{\mu}_N) \quad (5.4)$$

$$\Sigma_{i,L \times L} = \mathbf{W}_{PCA}^T \Sigma_{i,N \times N} \mathbf{W}_{PCA}$$

After PCA process, we have covariance matrix for each class $\Sigma_{i,L \times L}$, which is a $L \times L$ matrix for class- i . For each class covariance matrix Σ_i , we find null space and column space of it. Queries, which can also be called as test objects, are in column space. Since column space

and null space are perpendicular to each other, dot product of query with null space should be small enough to say that the query belongs to this class. For a query, we do this operation for each class, then choose the minimum one that gives its class. This class is regarding the magnitude of slope.

After projections, we find Approximate Null Space (ANS) matrix $\mathbf{W}_{ANS,i}$, along which intra-class variances are smallest, for each class. We use again eigenvalue analysis to construct ANS, which has n_i eigenvectors corresponding to the n_i smallest eigenvalues. For this purpose, there are two conditions for two thresholds to construct an efficient ANS:

i) There should be a threshold γ_1 , which is high enough to find smallest eigenvalues' eigenvectors. These eigenvalues satisfy

$$\lambda < \lambda_{max,i}/\gamma_1 \quad (5.5)$$

We construct columns of ANS for each *class-i* that obey the rule above in (5.5). In our experiments, we use $\gamma_1 = 10^5$.

ii) There is another condition on the ANS to satisfy that *class-i* is linearly separable from any other *class-j*. There should be an another threshold such that

$$|(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T \mathbf{e}_i| > \gamma_2 \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\| \quad (5.6)$$

, where $\mathbf{e}_i$ is the *i-th* column in ANS. In our experiments, we use $\gamma_2 = 10^{-2}$.

After constructing an Approximate Null Space for each class, we classify a query by calculating its distance to every class

$$d_i(\mathbf{y}) = \|\mathbf{W}_{ANS,i}^T(\mathbf{y} - \boldsymbol{\mu}_i)\| \quad (5.7)$$

These are the distances to the every class and minimum distance shows that X_L belongs that class. The calculations till (5.7) are in offline process, but (5.7) is an online process that

should be finished in real time for real applications.

For the parameter estimation problem, the maximum estimation error is

$$\varepsilon_{max} = \pm |C_{i+1} - C_i|/2 \quad (5.8)$$

, which is actually half of the bin range in terms of degrees.

5.1. UPDATED COVARIANCE BASED METHOD

Our initial work Principal Component Null Space Analysis, which is called as Covariance Based (CB) estimation, divides each continuous road parameter into discrete class bins linearly. For example, if a parameter is in (a, b) interval and we want to have C number classes, we can sample the interval (a, b) linearly. Each sample is called as a class or bin for the parameter. For each class (bin) of a parameter, CB algorithm finds Approximate Null Space (ANS) of class covariance matrix. If a projected test vector ($\mathbf{y}$) of road lines is in the column space of the class- i covariance matrix, product ($d_i(\mathbf{y})$) of test vector ($\mathbf{y}$) and null space is close to 0. It classifies a query by calculating its distance to every class as in (5.7), where $\mathbf{W}_{ANS,i}$ i -th class ANS and $\boldsymbol{\mu}_i$ is the i -th class mean vector that are found previously in offline process. Minimum of $d_i(\mathbf{y})$ over i running index gives the corresponding class- i .

However, since a road parameter divided into discrete bins, estimations are discrete values (*classVali*). In other words, it is a classification algorithm between bins of possible values of a parameter. There are infinitely many parameter values between class values. To have a soft estimation covering all values, we develop an Algorithm 5.1, where $\mathbf{d}$ is a distance vector in (5.7) for C classes of a parameter. *classVali* (C_i) are the linearly sampled values of a parameter in the training step. d_{sort} is sorted distance vector and its first element has the minimum distance to the most possible class value, which is *classVal1*. If the minimum and second minimum distance values are next to each other, we find the estimation as line 5 in the algorithm by using d_{sort} as weights.

The corresponding plot is given in Figure 5.2, where x-axis is the class values sampled at

Algorithm 5.1. Updated CB algorithm

```

1:  $[d_{sort}, index] = sort(d, "ascend")$ 
2:  $classVal1 = find\_class(index[1])$ 
3:  $classVal2 = find\_class(index[2])$ 
4: if  $abs(index[1] - index[2]) == 1$  then
5:    $est = \frac{classVal1.d_{sort}[2] + classVal2.d_{sort}[1]}{d_{sort}[1] + d_{sort}[2]}$ 
6: else
7:    $est = classVal1$ 
8: end if

```

integers between -5 and 5 of any parameter. Note that class values are not supposed to be integers, which are here given only for explanatory purpose. The tested parameter value is 1.33, which is close to class value 1. By using the distances as weights we interpolate linearly between class value 1 and 2.

Moreover, we improve CB algorithm with clothoid road model and generalize soft estimation method to 7 road parameters.

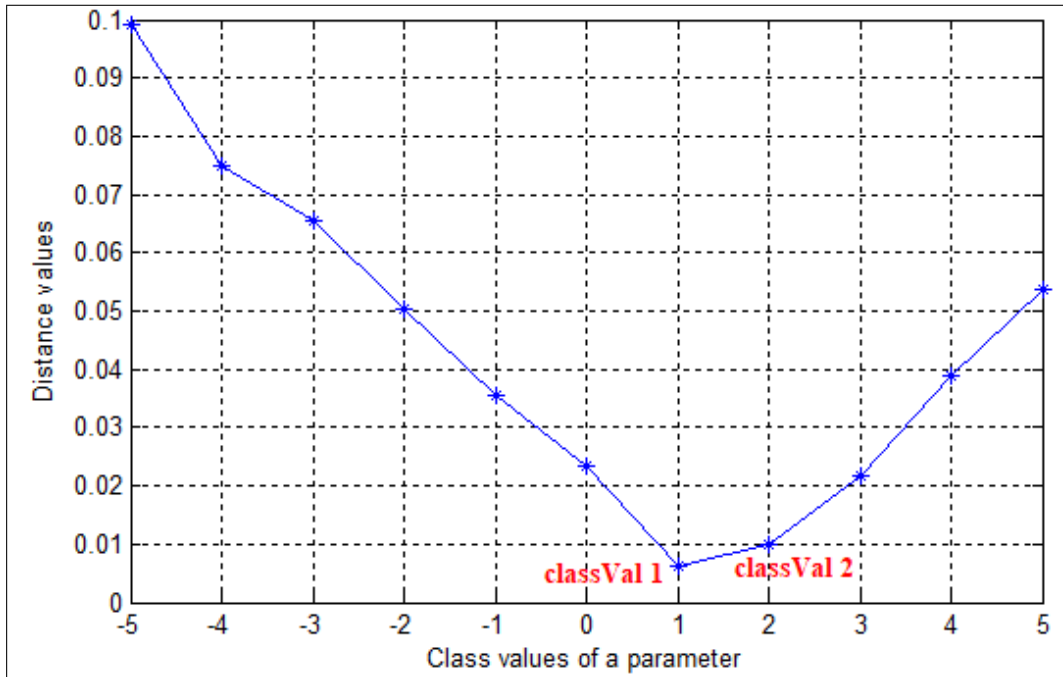


Figure 5.2. V-shape of distance values for CB method at 50dB

As for a discussion about the complexity, we already calculate $\mathbf{W}_{PCA}$, $\mathbf{W}_{ANS}$, and mean vectors $\boldsymbol{\mu}$ -s in offline training mode. In online testing mode, we only need the first line of the equation (5.4). That means, for a real time operation the required number of operations are few since all the eigenvalue operations are done in advance before the test in (5.7).



6. ITERATIVE LEAST SQUARES (LS) ESTIMATION AND OPTIMIZATION (ILSO)

Iterative Least Squares Estimation and Optimization (ILSO) method can derive analytical closed form solutions under Clothoid RM (CRM).

The algorithm is divided genuinely into 2 processes, which are Least Squares (LS) estimation [82, 83] for road parameters, and optimization [84, 85] for Z longitudinal distances, iteratively. First, we define initial distance values $Z_{l,r}^k[n]$ corresponding pixels on left (l) and right (r) lines. k is not an exponent of Z but it is the iteration index and $k = 0$ means initial chosen values. n is for pixel index, which are samples of road lines in an image. n changes between 1 to N_l or N_r for road left and right lines respectively. Using these Z values, LS estimates the road parameters, which are $\mathbf{p} = [d_l, d_r, \alpha, a_h, b_h, a_v, b_v]$. The number of road parameters are 7. There are totally $7 + N_l + N_r$ unknown parameters to be estimated. Then, Optimization step finds remaining $N_l + N_r$ distance values $Z_{l,r}^{k>0}[n]$ for each n and each line by improving distance values at each iteration step k . Overall process can be seen in Figure 6.1 and a summary of the method is given in Algorithm 6.1.

Algorithm 6.1. ILSO method

- 1: *Set max iteration $k_{max} = 10$*
- 2: *Set initial distances for left and right lines $Z[n]^{k=0}$*
- 3: LS finds road parameters $\mathbf{p}$ using distances $Z[n]^k$
- 4: Optimization finds distances $Z[n]^k$ using road parameters $\mathbf{p}$
- 5: **if** Average difference $Z[n]^{k-1} - Z[n]^k < it_{th} = 0.005$ in (6.17) **then**
- 6: Stop iterations and give results
- 7: **else**
- 8: Increase iteration k and turn Step 3:
- 9: **end if**

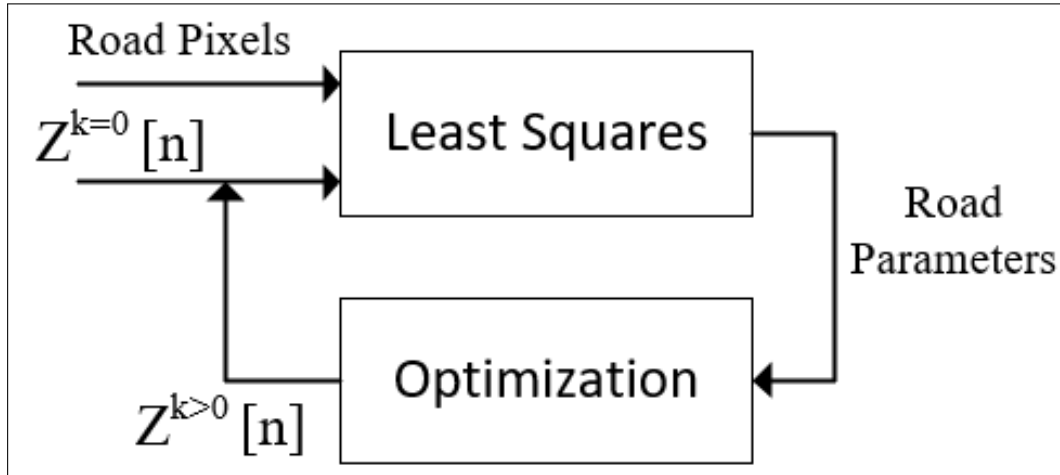


Figure 6.1. Iterative estimation processes

6.1. ITERATIVE LEAST SQUARES (LS) ESTIMATION

Least Squares (LS) is a systematic way to estimate road parameters, in which there are 7 unknown parameters. In each frame LS can be applied on left and right lines separately.

Before applying LS, there must be preprocess to make 2D equations be a polynomial of Z and be independent of focal length. That is, 2D image observations are compensated by multiplying with Z/f

$$u'_l[n] = u_l Z/f = -d_l \cos \alpha + a_h Z_l[n] + b_h Z_l^2[n]$$

$$v'_l[n] = v_l Z/f = d_l \sin \alpha - Y_c + a_v Z_l[n] + b_v Z_l^2[n]$$

(6.1)

$$u'_r[n] = u_r Z/f = d_r \cos \alpha + a_h Z_r[n] + b_h Z_r^2[n]$$

$$v'_r[n] = v_r Z/f = -d_r \sin \alpha - Y_c + a_v Z_r[n] + b_v Z_r^2[n]$$

to have modified u' and v' values. Actually (6.1) is similar to 3D road model in (3.2), except now Z values are to be estimated ones.

There are 4 LS solutions, which are for u and v values of left and right lines. The $N \times 3$

information matrix ($\mathbf{X}_l$) is denoted for LS as

$$\mathbf{X}_l = \begin{bmatrix} Z_l^2[1] & Z_l[1] & 1 \\ Z_l^2[2] & Z_l[2] & 1 \\ \vdots & \vdots & \vdots \\ Z_l^2[N_l] & Z_l[N_l] & 1 \end{bmatrix} \quad (6.2)$$

Similar equation to (6.2) for right line can be derived, however we give only for left line in order to avoid repetition.

The unknown parameter vectors, which are 3x1 column vectors, are

$$\begin{aligned} \mathbf{x}(u'_l) &= \begin{bmatrix} b_h & a_h & -d_l \cos(\alpha) \end{bmatrix}^T \\ \mathbf{x}(v'_l) &= \begin{bmatrix} b_v & a_v & d_l \sin(\alpha) - Y_c \end{bmatrix}^T \\ \mathbf{x}(u'_r) &= \begin{bmatrix} b_h & a_h & d_r \cos(\alpha) \end{bmatrix}^T \\ \mathbf{x}(v'_r) &= \begin{bmatrix} b_v & a_v & -d_r \sin(\alpha) - Y_c \end{bmatrix}^T \end{aligned} \quad (6.3)$$

Each parameter vector in (6.3) is found by LS using modified observations in (6.1)

$$\mathbf{b}_{l1} = \mathbf{x}(u'_l) = (\mathbf{X}_l^T \mathbf{X}_l)^{-1} \mathbf{X}_l^T \mathbf{u}'_l$$

$$\mathbf{b}_{l2} = \mathbf{x}(v'_l) = (\mathbf{X}_l^T \mathbf{X}_l)^{-1} \mathbf{X}_l^T \mathbf{v}'_l$$

(6.4)

$$\mathbf{b}_{r1} = \mathbf{x}(u'_r) = (\mathbf{X}_r^T \mathbf{X}_r)^{-1} \mathbf{X}_r^T \mathbf{u}'_r$$

$$\mathbf{b}_{r2} = \mathbf{x}(v'_r) = (\mathbf{X}_r^T \mathbf{X}_r)^{-1} \mathbf{X}_r^T \mathbf{v}'_r$$

, which are 4 typical linear LS estimates. Note that $\mathbf{b}_{l1}$ and $\mathbf{b}_{l2}$ are found by left line and $\mathbf{b}_{r1}$ and $\mathbf{b}_{r2}$ are found by right line. One important observation is that there are 4 equations and 3 unknowns d_l , d_r and α in the third elements of parameter vectors in (6.3). If we solve for alpha in left line,

$$\hat{\alpha}_l = \arctan \left(-\frac{b_{l2}[3] + Y_c}{b_{l1}[3]} \right) \quad (6.5)$$

Based on $\hat{\alpha}_l$, we can find distance to the left line

$$\hat{d}_l = \frac{b_{l2}[3] + Y_c}{\sin(\hat{\alpha}_l)} \quad (6.6)$$

Similar solutions can be found using right line for alpha

$$\hat{\alpha}_r = \arctan \left(-\frac{b_{r2}[3] + Y_c}{b_{r1}[3]} \right) \quad (6.7)$$

, and then the distance to right line is

$$\hat{d}_r = \frac{b_{r2}[3] + Y_c}{-\sin(\hat{\alpha}_r)} \quad (6.8)$$

To exploit diversity of 2 lines, we can find overall alpha as

$$\hat{\alpha} = \arcsin \left(\frac{b_{l2}[3] - b_{r2}[3]}{\hat{d}_l + \hat{d}_r} \right) \quad (6.9)$$

, which slightly better than average of $\hat{\alpha}_l$ and $\hat{\alpha}_r$ in simulations. Since lateral angle α is around 0° including negative and positive values, $\sin()$ function is used instead of $\cos()$, since $\sin()$ is linear around 0° and $\cos()$ value might be higher than 1 due to noise, although $-1 \leq \cos() \leq 1$.

The other parameters a_h, b_h, a_v, b_v are easy to find since they are the direct output of the solution (6.4). There are 2 estimations for these parameters from left and right line. Each group of estimation is used in optimisation step for left or right line respectively.

6.2. OPTIMIZATION FOR DISTANCE ESTIMATION

Distance estimation is one of the challenging problems in computer vision both for monocular and binocular camera. For each pixel there is an unknown distance value (Z), hence there are as many pixels as the unknown Z values. The pixels on close points of road lines can be a good start for Z estimation, since close points are expected to have less measurement error.

Step.1 Find distance of closest pixel: We can derive equations for left line and similar equations can be written for right line. For the left line, we can use vertical axis v_l , which is in the compensated estimation in (6.1), for Z estimation. The error for left line can be defined between theoretical value and measurement as

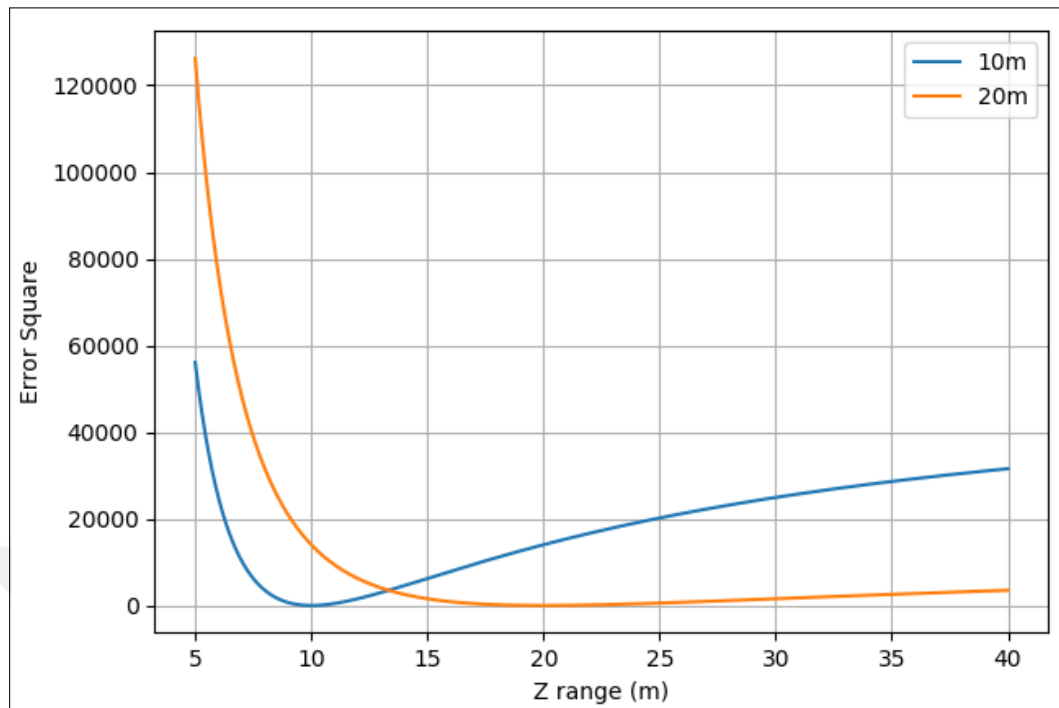


Figure 6.2. Square error at Z=10m and Z=20m for v_l (linear scale)

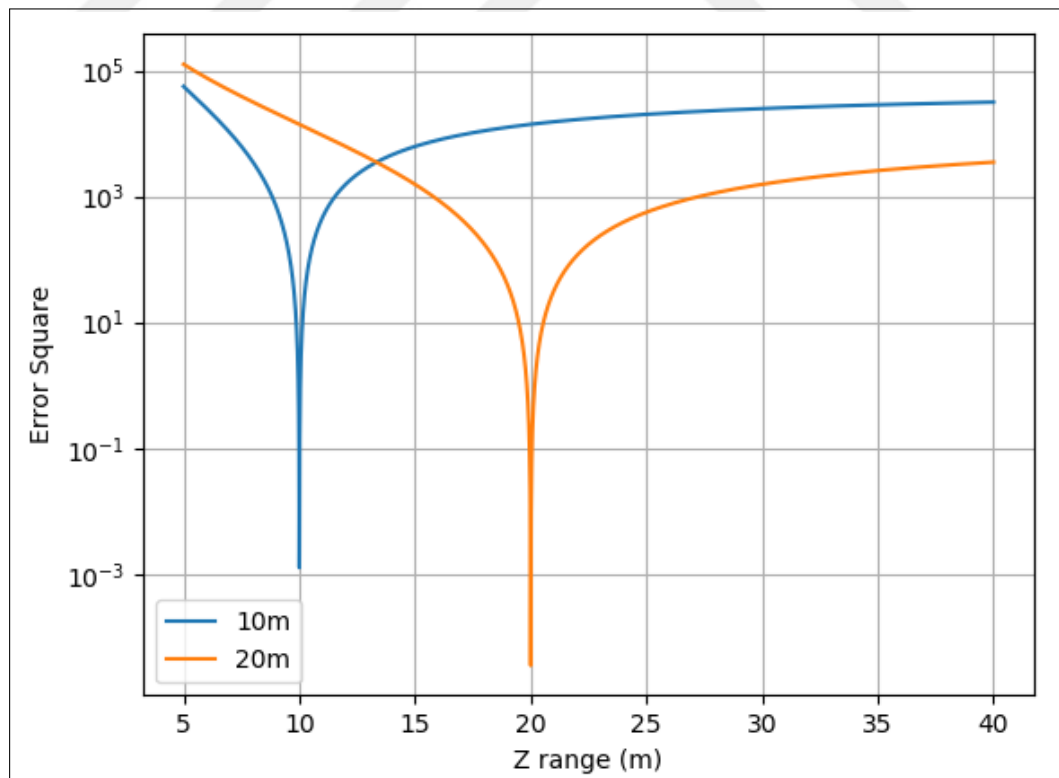


Figure 6.3. Square error at Z=10m and Z=20m for v_l (logarithmic scale)

$$\epsilon = v_l Z_l / f - v'_l \quad (6.10)$$

, where n -th index is dropped for easy notation and v_l is theoretical value and v'_l is modified observations in (6.1). ϵ^2 is a convex function of Z_l as seen in Figure 6.2 and corresponding logarithmic scale on vertical axis is given in Figure 6.3. For simplicity, we drop left subscript l from Z in our equation derivations. However, derivations still apply for both left and right lines. Due the convexity of ϵ^2 as seen in Figure 6.2, we can easily take its derivative to find optimal Z location, which is going to be our Z estimation for a pixel

$$\begin{aligned} \frac{d\epsilon^2}{dZ_l} = & Z_l^0 (-2B_l a_v + 2v_l B_l / f) \\ & + Z_l^1 (2v_l^2 / f^2 + 2a_v^2 - 4B_l b_v - 4v_l a_v / f) \\ & + Z_l^2 (6a_v b_v - 6v_l b_v / f) \\ & + Z_l^3 (4b_v^2) \end{aligned} \quad (6.11)$$

, where $B_l = d_l \sin \alpha - Y_c$. Similar equation can be derived by using v_r and in this case $B_r = d_r \sin \alpha + Y_c$. Equation (6.11) is a 3rd order polynomial and its roots for Z can be found by any builtin function in Python or Matlab. We can accept a root for $\hat{Z}_l^k[n = 1]$ between 0m and 20m for the closest pixel $v_l[n = 1]$. This convex optimization step can be done for closest samples to the ego vehicle since they have less (measurement) noises. We use it for the closest one ($n = 1$) as accepting row index n in image starting from bottom of the image.

Since the optimization step is sensitive to the previous errors from LS step and the errors for big Z values, it not converge to actual parameter values. Therefore, we develop update step

below for remaining samples.

Step.2 Find other distances of pixels: In this step, other remaining pixels' distances are found based on the closest pixel's distance found in Step.1.

For k -th iteration, vertical value $Y_l^k[n = 1]$ for the closest pixel, which is actually the first pixel ($n = 1$), is found as

$$Y_l^k[1] = d_l \sin \alpha - Y_c + a_v \hat{Z}_l^k[1] + b_v \hat{Z}_l^{2,k}[1] \quad (6.12)$$

$Y_l^k[1]$ is the basis for finding other values $Y_l^k[n > 1]$, since for the closest point Z is small and estimations errors of a_v and b_v are negligible.

Then other Y values for other pixels $n = 2, 3, \dots, N_l$ can be found as

$$\begin{aligned} Y^k[n] = & Y^k[1] + a_v(\hat{Z}_l^{k-1}[n] - \hat{Z}_l^k[1]) \\ & + b_v(\hat{Z}_l^{2,k-1}[n] - \hat{Z}_l^{2,k}[1]) \end{aligned} \quad (6.13)$$

, where $k-1$ is previous iterations and k is current iteration. Then we can update Z values for k -th iteration and n -th pixel as

$$\hat{Z}_l^k[n] = Y^k[n]f/v_l[n] \quad (6.14)$$

Similar equations can be derived for right line by starting from

$$Y_r^k[1] = -d_r \sin \alpha - Y_c + a_v \hat{Z}_r^k[1] + b_v \hat{Z}_r^{2,k}[1] \quad (6.15)$$

For bigger values of α in curves and not very accurate Z estimations in real images, lateral

angle estimation can be modified as

$$\hat{\alpha}^k = \arctan \left(\frac{E[Y_l^k] - E[Y_r^k]}{\hat{d}_l + \hat{d}_r} \right) \quad (6.16)$$

, where $E[.]$ is sample mean operator of height values over pixel samples.

6.3. ITERATION STOP CRITERIA

We initially set maximum allowed iteration number as $k_{max} = 10$. We change iteration number for each frame depending on a condition defined in (6.17). As long as Z values change (Δ) between iteration k and k+1 is more than it_{th} as threshold, we continue the ILSO algorithm till k_{max} . If change (Δ) between iterations is less than it_{th} , we stop iterations and claim results are found. We use iteration threshold $it_{th} = 0.005$.

$$\Delta = \frac{\sum_{n=1}^{N_l} |\hat{Z}_l^{k+1}[n] - \hat{Z}_l^k[n]| + \sum_{n=1}^{N_r} |\hat{Z}_r^{k+1}[n] - \hat{Z}_r^k[n]|}{N_l + N_r} \quad (6.17)$$

7. LOCAL FEATURES BASED (LFB) SLOPE ESTIMATION

Similar to the Geometric Based (GB) solutions, Local Features Based (LFB) method uses Basic RM (BRM) that assumes exact straight roads and exact circles on the road. Since LFB uses simplicity of BRM, it can derive closed form expressions. Therefore, LFB method does not work under complex road structure of Clothoid RM (CRM). LFB estimation is our only method that does not use road lines as inputs. Instead it uses the changes of pixel locations of local features considering BRM.

Local Features (LF) are effective and popular tools in Computer Vision for the object detection and classification problem. However, we use them for another novel purpose, which is to estimate road slope. LFB method estimates road slope when the vehicle moves through a changing slope. Thus this is not in advance estimation but it is current estimation of road. This property is a drawback with respect to the other proposed algorithms. However, since LF is used on a large scale in image processing and computer vision for object recognition, adding slope estimation can be an extra benefit to the existing usage. A very popular local features algorithm SIFT [27, 86, 87] can be a good candidate for our purpose, since SIFT features are robust to scaling, to small rotations, partially invariant to illumination and it has good matching properties. Here, other features such as BRISK can also be applied. Summary and comparison of those descriptors including SIFT and BRISK are left to the reader and can be found in [86, 87].

Basically, SIFT has two important components: Keypoints are the pixel locations $p = [uv]^T$, where ^{T} is transpose of a vector. Local features of an image are the vectors which have the histogram information of the gradients around each keypoint. Each keypoint has at least 1 local features. We operate SIFT on 2D gray value images without trying to convert back to 3D. To stay in a low complexity and to have high correct performance, we offer to use the upper keypoints, which are on the upper part of an image. We consider 80 percent of the image from top. That gives the keypoints, which are very far away from the vehicle. Therefore, in a short period of time of driving the distance on Z-axis doesn't change much but a slope can be already changed. These proper keypoint candidates exist at traffic signs,

buildings, on clouds, or at the top of the mountains.

A pair of images for matched SIFT features is shown in Figure 7.1, where on the right image the car moved 30m front assuming all other objects are static. The left figure shows 2.86° longitudinal angle. As expected matching should be between similar parts of the similar objects. Most of the matches are correct, however a careful reader can see few mismatched keypoints in the figure.

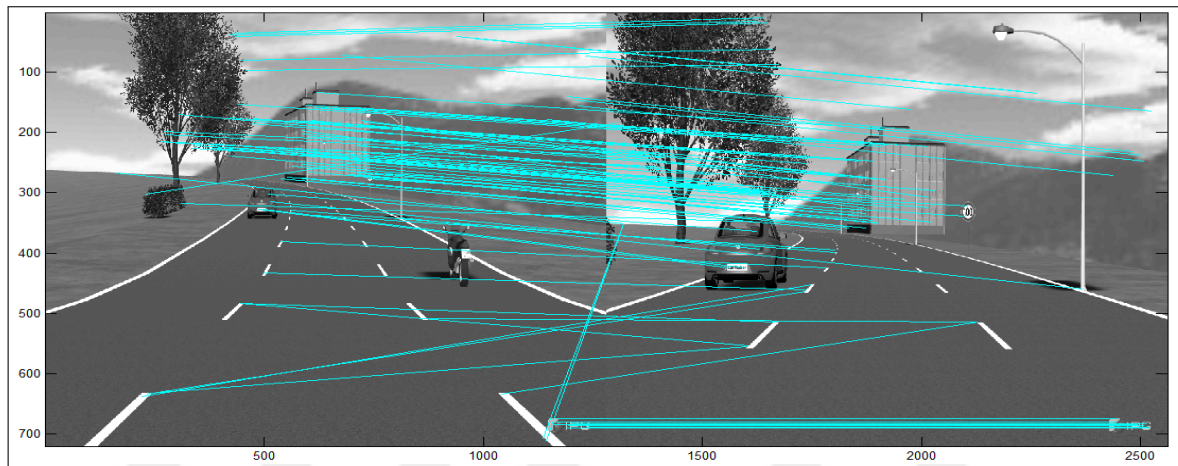


Figure 7.1. SIFT matches between 2 correlated images

The 3 angles, which are roll(lateral), yaw(turning left or right), pitch(longitudinal), can be found by applying 2 different methods in this section. For roll estimation, direction of Z -axis doesn't change, the SIFT keypoints will be rotated in the image as experiencing the slope. For pitch and yaw estimation, because of the change in direction of Z -axis we use another method. We give derivations for pitch and with similar technique yaw can be found.

We assume that traveled distance by the car is negligible between two video frames for our SIFT based slope estimations. If speed of a car is 30m/sec (108kph) and video frame rate is 30fps, then traveled distance is only 1m between 2 frames i.e. that makes the assumption logical.

7.1. LFB: LATERAL ANGLE ESTIMATION

Let's say vehicle has keypoint at p and after a small driving time the corresponding matching SIFT point is at p' , which is an affine transform of the previous point p . Affine transform is composed of rotation and translation in 2D image

$$\mathbf{p}'_i = \mathbf{R}\mathbf{p}_i + \mathbf{t} \quad (7.1)$$

$$\mathbf{p}'_j = \mathbf{R}\mathbf{p}_j + \mathbf{t}$$

, where i and j are any two not identical indices. $\mathbf{p}$ is vector representation of a point p . $\mathbf{t}$ is (2x1) translational vector, and $\mathbf{R}$ is (2x2) rotation matrix

$$\mathbf{R} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \quad (7.2)$$

To cancel the $\mathbf{t}$ term in (7.1) we subtract the second equation from the first one in (7.1)

$$\begin{bmatrix} u'_j - u'_i \\ v'_j - v'_i \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} u_j - u_i \\ v_j - v_i \end{bmatrix} \quad (7.3)$$

This avoids calculation of the translation vector $\mathbf{t}$. After algebraic manipulation using (7.3) to separate for α , we get

$$\begin{bmatrix} \Delta u' \\ \Delta v' \end{bmatrix} = \begin{bmatrix} \Delta u & \Delta v \\ \Delta v & -\Delta u \end{bmatrix} \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix} \quad (7.4)$$

, where we used $\Delta u = u_j - u_i$, $\Delta v = v_j - v_i$, $\Delta u' = u'_j - u'_i$, and $\Delta v' = v'_j - v'_i$. After

matrix inversion, we find cosine and sine terms

$$\cos(\alpha) = \frac{\Delta u \Delta u' + \Delta v \Delta v'}{\Delta u \Delta u + \Delta v \Delta v} \quad (7.5)$$

$$\sin(\alpha) = \frac{\Delta v \Delta u' - \Delta u \Delta v'}{\Delta u \Delta u + \Delta v \Delta v}$$

Inverse functions of both cosine and sine function in theory can be used to find α . However, we prefer to use sine. One reason is that since numerator and denominator are close to each other, a noisy measurement can make the ratio term greater than 1 for cosine that makes α estimate a complex number. Another reason is that around $\alpha = 0$, cosine looks like flat and it is not separable. However around $\alpha = 0$ sine tends to be more linear and separable for arcsine. Then, we estimate lateral slope as,

$$\hat{\alpha} = \arcsin\left(\frac{\Delta v \Delta u' - \Delta u \Delta v'}{\Delta u \Delta u + \Delta v \Delta v}\right) \quad (7.6)$$

If there are N number of key points, the number of the pairs or subtractions as in (7.1) and (7.3) is $N(N-1)/2$. For any pair, we have initial estimations. If the number $N(N-1)/2$ is greater than 200, we use histogram of $N(N-1)/2$ estimations and decide on the $\hat{\alpha}$, which has the highest rate of occurrences. Otherwise for smaller values, mean value of individual estimations gives better solutions.

7.2. LFB: LONGITUDINAL ANGLE ESTIMATION

Pitch Estimation is simpler than roll estimation, since by using only v -axis on the image we can find the β values for the longitudinal slope. We assume translational change, which is the amount of longitudinal driven distance, is small around point A in Figure 7.2. Point P is the symbolization of SIFT key point in 3D. The angle change happens in Y - Z plane in Figure 7.2 and $|PB|$ is the Y -axis before point-A and $|PB'|$ is for the Y' -axis after passing point-A.

Then, for the uphill pitch case, by using the formula in (3.6) the estimated angle is

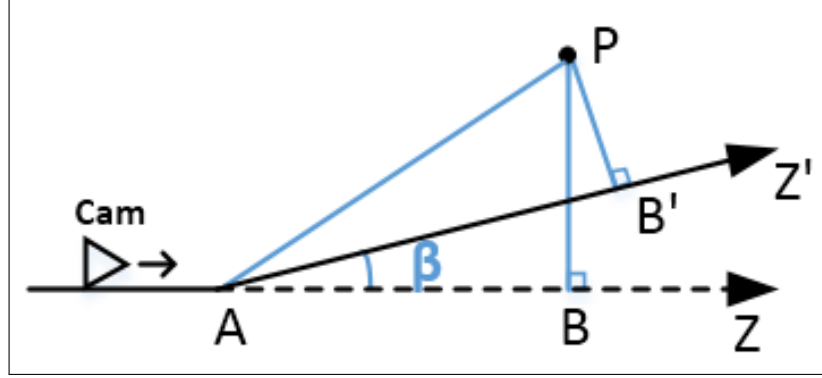


Figure 7.2. Longitudinal slope estimation by SIFT

$$\hat{\beta} = \arctan(v/f) - \arctan(v'/f) \quad (7.7)$$

, where v is the column value of a SIFT keypoint on the image before rotation. v' is the corresponding SIFT keypoint after rotation. The $\hat{\beta}$ value in (7.7) is in term of radians. Actually, (7.7) finds the difference between the angles $\angle PAB$ and $\angle PAB'$.

7.3. LFB: YAW ANGLE ESTIMATION

Yaw angle estimation is the amount of angle change taken by left turn or right turn. The change happens in X - Z plane as seen in Figure 7.3, where $|PB|$ is the X -axis before point-A and $|PB'|$ is for the X' -axis after passing point-A.

This estimation is also simple in complexity. The similar assumption in Section 7.2 still holds that driven distance is small around point A, where angle changes. Then, for the left turn case, by using the formula in (3.6) the estimated angle is

$$\hat{\theta} = \arctan(u/f) - \arctan(u'/f) \quad (7.8)$$

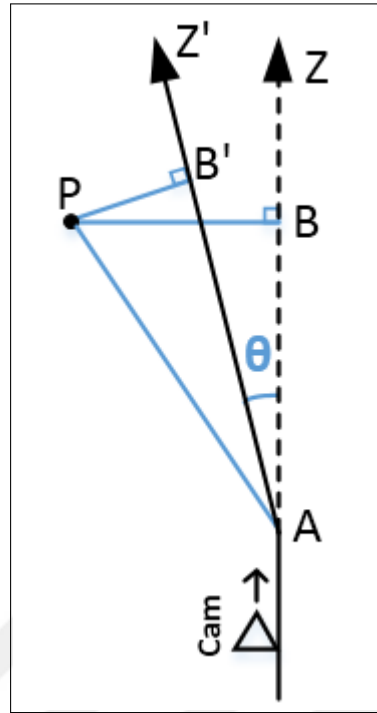


Figure 7.3. Yaw angle estimation by SIFT

, where u is the raw value of a SIFT keypoint on the image before rotation. u' is the corresponding SIFT keypoint after rotation. The $\hat{\theta}$ value in (7.8) is in term of radians. Actually, (7.8) finds the difference between the angles $\angle PAB$ and $\angle PAB'$.

For Yaw and Pitch estimations, mean value of estimations due to matched SIFT points is used. One issue for this technique is that if Roll exists with Yaw or Pitch, the problem can not be solved with this method. However, since u and v are independent, this algorithm works when even Yaw and Pitch coexists. Found values can be accumulated over time to cover all angle change.

8. NUMERICAL RESULTS AND DISCUSSIONS

We divide our results into two parts: First uses Basic Road Model (BRM) and it compares GB, CB, LFB and MLP results. The second part uses Clothoid Road Model (CRM) as an improvement of BRM and it compares CB, ILSO and MLP results. MLP provides additional comparison of our developed methods.

8.1. RESULTS USING BASIC RM

We separate our results into 4 subsections: Geometry Based (GB), Local Feature Based (LFB), Covariance Based (CB), and a comparison and discussion subsection of all three methods. We compare our methods with an additional state-of-art Artificial Neural Network (ANN) technique, which is MultiLayer Perceptron (MLP) using detected road lines as feature. Our testing environment is composed of virtually generated CarMaker (CarMaker) images [60] with resolution of 1280x720 pixels, mathematical road model (BRM) that is defined in (3.2) and (3.3), and real images from Samsung Galaxy S6 mobile phone's back camera(S6), whose resolution is also 1280x720 pixels. CarMaker and BRM are called controlled test environment and S6 is uncontrolled test environment. In figures, we use an abbreviation e.g. (GB:Geometry Based, CarMaker) to give a quick understanding of used estimation method and test environment respectively.

8.1.1. Geometry Based Results

We use CarMaker road images to conduct our experiments. In CarMaker, percentage slope (SlopePer) is used and defined as 100 times tangent of an angle that is $100 \tan(\cdot)\%$. We generated 4 different slopes, whose percentages are 0, 5, 10, 15 and corresponding angles are $SlopeDeg = [0^\circ, 2.8624^\circ, 5.7106^\circ, 8.5308^\circ]$ respectively calculated by using

$$SlopeDeg = 180 \arctan(SlopePer/100)/\pi \quad (8.1)$$

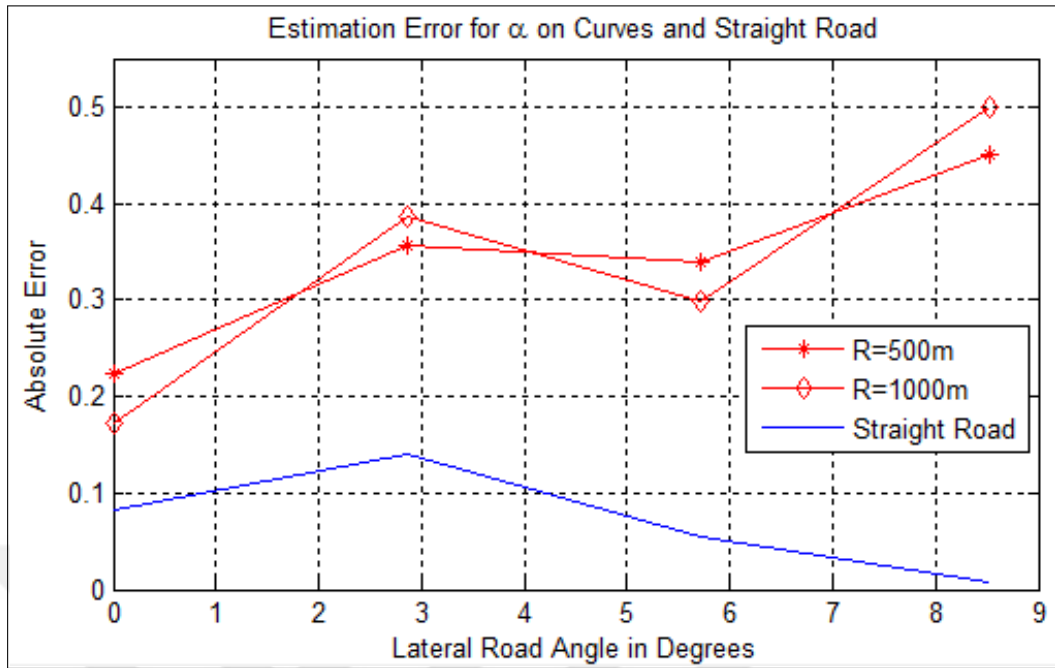


Figure 8.1. Lateral slope estimations (GB,CarMaker)

The results for the lateral angle for straight road and curved roads are depicted in Figure 8.1, where x-axis is actual angle values (*SlopeDeg*) and y-axis is absolute estimation errors of those angles. It is seen that lateral slope estimations for straight road are better than the curved road, since finding exact tangent point for (4.11) on the pixel level is not an easy task. The red lines for curves explain that the algorithm works well different radius lengths.

The error matrix as performance of (4.13) is shown in Figure 8.2, where percentage error (± 5) percent of m_l and d_l are applied for the case of curve $R=1000m$ with $\alpha = 5^\circ$. Maximum error is observed as 4.096° for $0.95m_l$ and $0.95d_l$ and second maximum error is observed on the other corner of $1.05m_l$ and $1.05d_l$ expectedly. Dark blue cells are about 0.01° estimation errors. Effect of R values in Table 4.1 is given in Figure 8.3 for using 0.5 approximation value in (4.10). The maximum error is 0.36° even for $R=100m$ and after $R=500m$ results are better and stable with using the approximation. It is seen that our approximation has less effect than errors in m_l or d_l .

The results for longitudinal slope are presented in Figure 8.4 for uphill. Two cases, which are straight road and curve road with $R = 1000m$, are depicted in the figure. All errors are

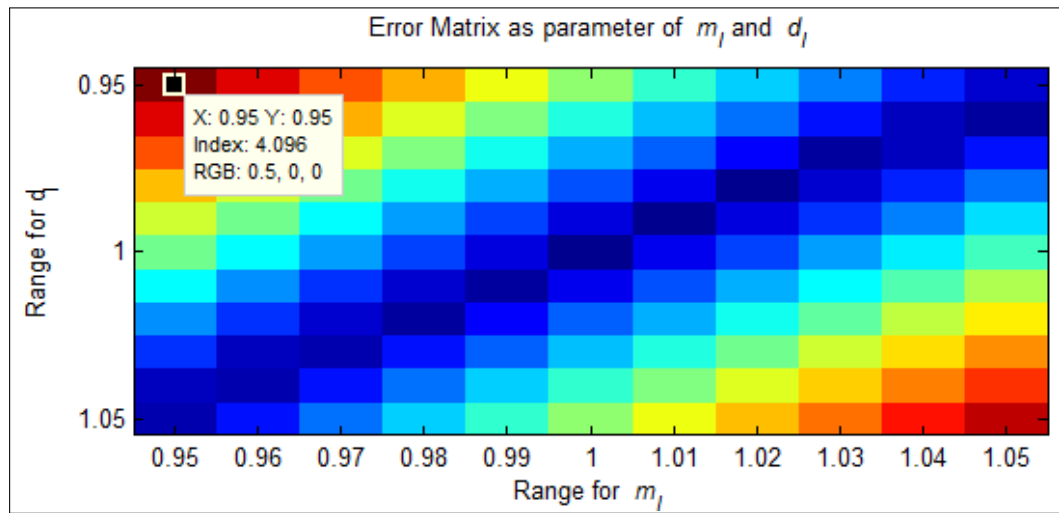


Figure 8.2. Lateral system dynamics (GB,BRM). Effect of m_l and d_l errors.

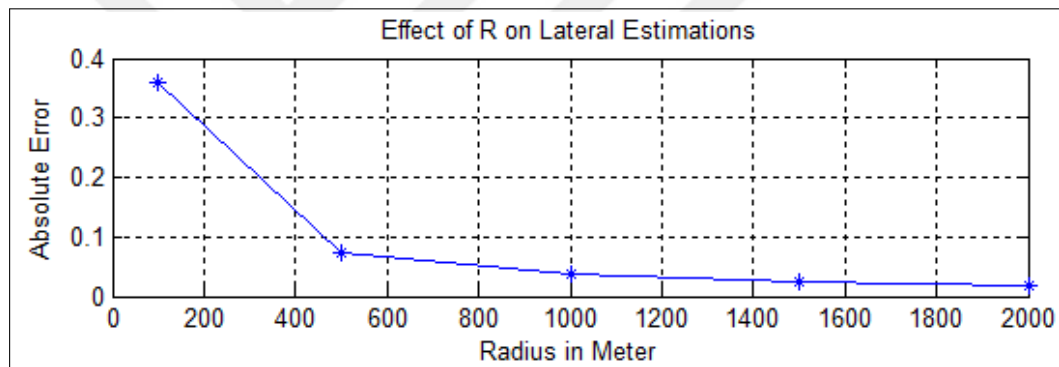


Figure 8.3. Lateral system dynamics (GB,BRM). Effect of R for $\alpha = 5^\circ$.

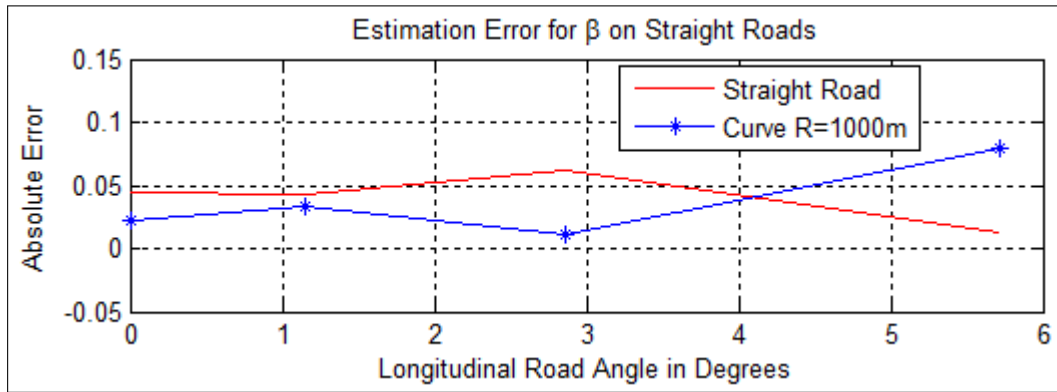


Figure 8.4. Longitudinal slope estimation errors for uphill (GB,CarMaker)

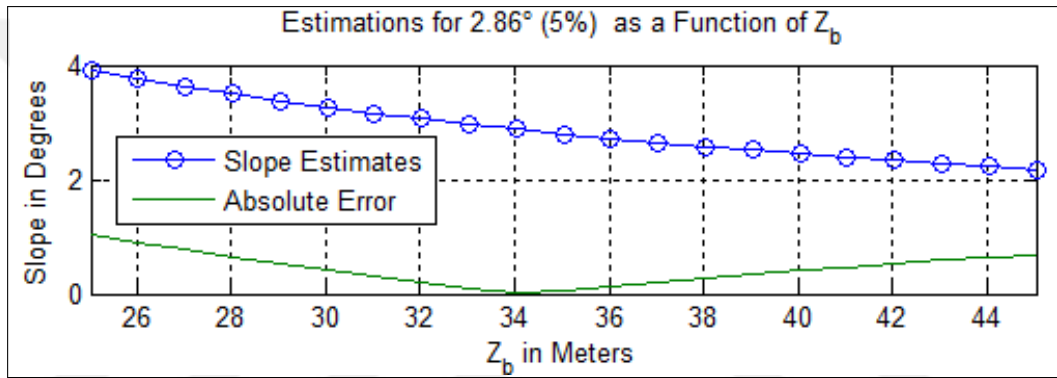


Figure 8.5. System dynamics depending on Z_b for β uphill (GB,CarMaker)

less than 0.1° . We assume that the lane width within the curve is the distance between image points, which have same v-axis value. Also it is assumed that the start point of inclination seen in Figure 4.1 is already found. The Figure 8.5 is to show dynamics of uphill slope estimation. We change Z_b in (4.15), since it is the most effective term in the equation. For 5 percent slope, if Z_b is estimated as just above 34m, the absolute error becomes minimum, which is 0° . For other Z_b values, we would have error. However, it wouldn't be a big error which is mostly less than 1° as shown on green line, since the range of the estimations are about 2° on blue line.

The results for downhill longitudinal slope are next depicted in Figure 8.6, which shows the performance of the algorithm in terms of absolute error. That is less than 0.1° as absolute error. Figure 8.7 is about the dynamics of the algorithm for changing Z_{min} in (4.16). The *Slope Estimation* in blue color in Figure 8.7 shows the range of the estimable values for the

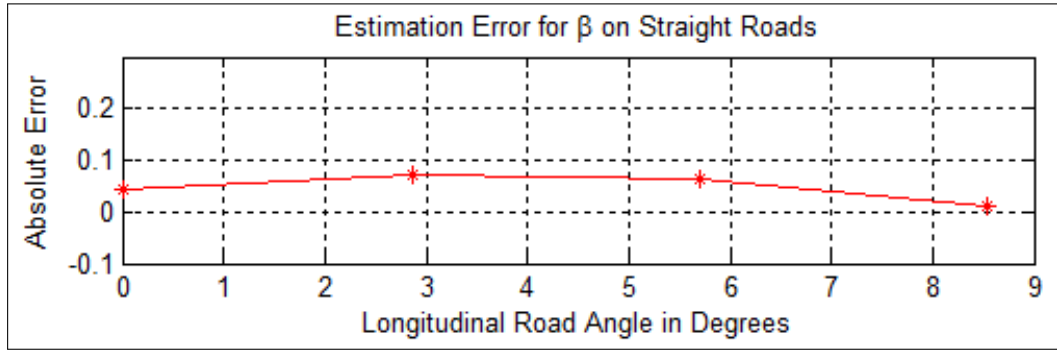


Figure 8.6. Longitudinal slope estimations for downhill (GB,CarMaker)

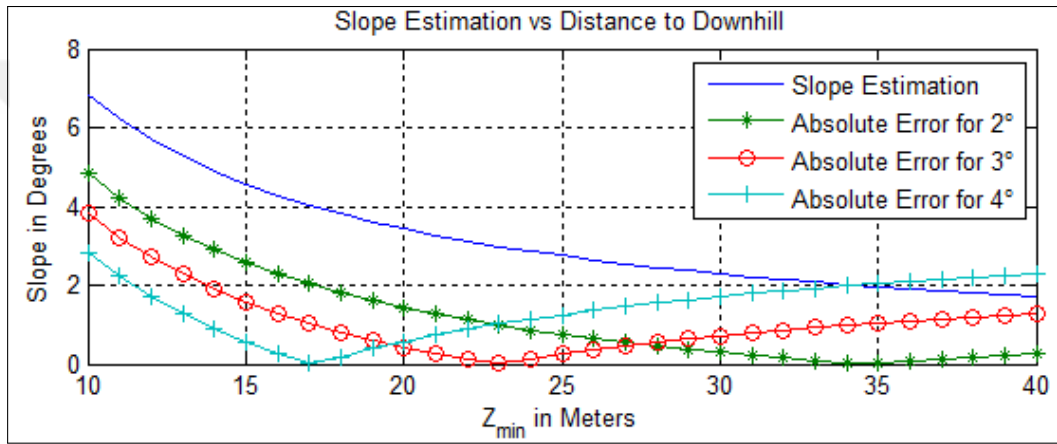


Figure 8.7. Downhill system dynamics depending on Z_{min} (GB, CarMaker)

road angle, that is around between 2 and 7 degrees for $10m < Z_{min} < 40m$. The errors in Z_{min} contribute in the slope estimation errors. It is seen that for lower road slope values, the algorithm is more robust to the errors in Z_{min} .

8.1.2. Local Features Based Results

Absolute error results for the lateral and longitudinal angle are given in Figure 8.8, where the road does not have a curvature. For lateral estimation, there is no pitch angle and initially the road is laterally straight, then 20m later it has a lateral slope in terms of percentage e.g. 0, 5, 10 and 15. At the starting point 0m and after 30m, which is to make it sure that car is on banked road, 2 images are shot and both are used for SIFT features extraction. Longitudinal slope (uphill) estimation process is handled similarly and results are given in Figure 8.8 without curvature and roll angle. Both errors are less than 1° and since different images are

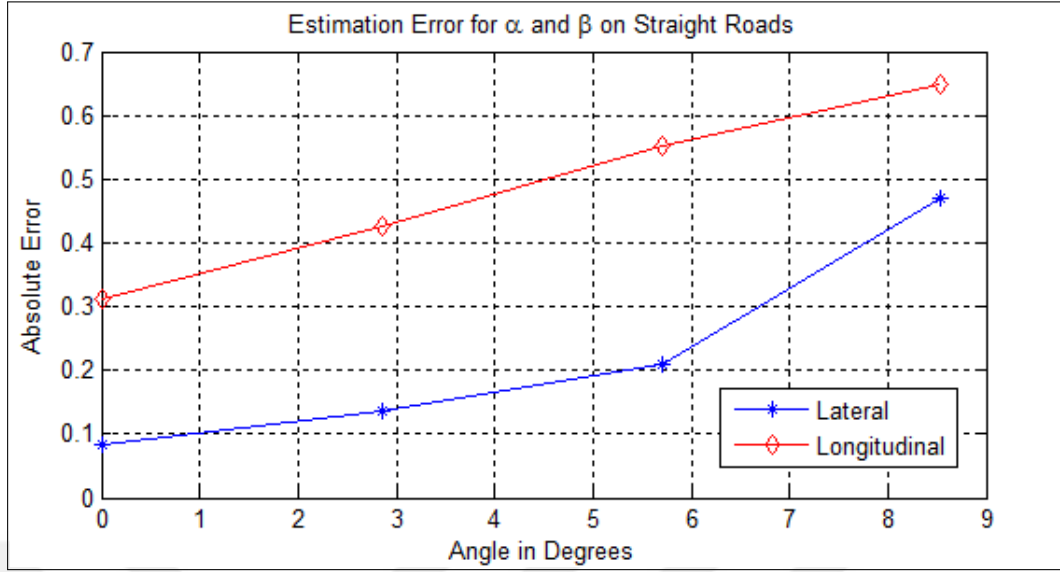


Figure 8.8. Slope estimations (LFB, CarMaker)

used for both estimations, there may be performance differences between α and β due to SIFT internal dynamics.

8.1.3. Covariance Based Results

We first quantize the continuous angle parameter, which is between 0° and 10° , into 11 bins for both lateral slope (α) and longitudinal slope (β) estimation where the bins are called as classes. For each class, we train the algorithm to be used in (5.7) by changing the parameters according to $R = 400 : 200 : 2000m$, $d_l = 1 : 0.25 : 2m$, and $\theta = 89 : -1 : 75^\circ \sim 85^\circ$, where the middle values in all three show the incremental size. θ range starts every time from 89° and ends between in the range $[75^\circ, 85^\circ]$ to create different curves. The total number of trials in training for each bin (class) becomes $K = 495$ and yields totally 5445 trials for all classes. Number of trials is 1010 in testing step. A trial vector is generated as in (5.1) whose length is $N = 128$.

We further compare the CB slope estimation results with another supervised learning algorithm MultiLayer Perceptron (MLP) composed of the following four fully connected layers. Input layer of MLP has 128 neurons because of training and test vectors' size $N = 128$, internal two layers has 64 neurons. Training and test vectors are composed of left

and right road lines. The output layer has 11 neurons for both lateral and longitudinal angles. Briefly the structure can be shown as MLP(128,64,64,11). MLP classification is realized with TensorFlow and Keras libraries in Python3.7. In the first 3 layers, *activation function Rectified Linear Unit (ReLU)* is used and in the last layer *Softmax activation function* is used. In Keras *compile* method, which configures the Keras model for training, *categorical cross entropy* is used for *loss function* and *Adam* is used for *optimizer*. The Keras model is trained with *fit* method with the parameters *epochs* = 10 and *batchsize* = 10. Then, the *predict* method in Keras model is used for estimations. Note that for MLP, we use the same training and test dataset generated above to evaluate the CB method. The results for lateral and longitudinal slopes are depicted in Figure 8.9 and Figure 8.10. Since all the bins have equal size 1° , for lateral slope and longitudinal slope we have 0.5° maximum error by using CB method. The minimum error can be attainable if the actual slope is close to center of bins. CB method is separable for lateral and longitudinal estimations. Although MLP works similarly to CB for lateral estimation, its longitudinal slope estimation performance is worse than CB.

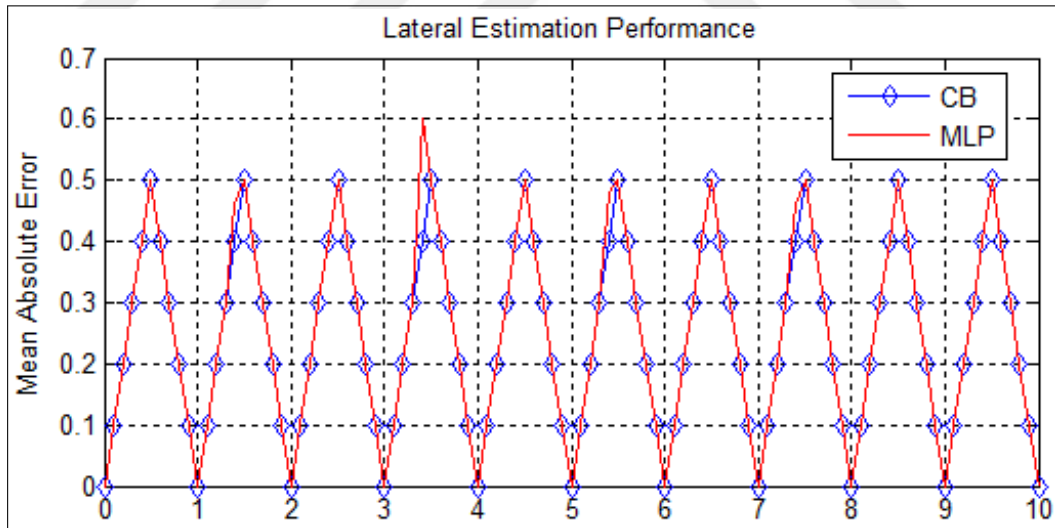


Figure 8.9. Comparison of CB and MLP by using BRM. Roll angle (α) in degrees

8.1.4. Comparison of Methods

GB, CB and MLP based slope estimation requires the road lines. Therefore, we compare the robustness of these algorithms for inexact road lines generated by adding independent

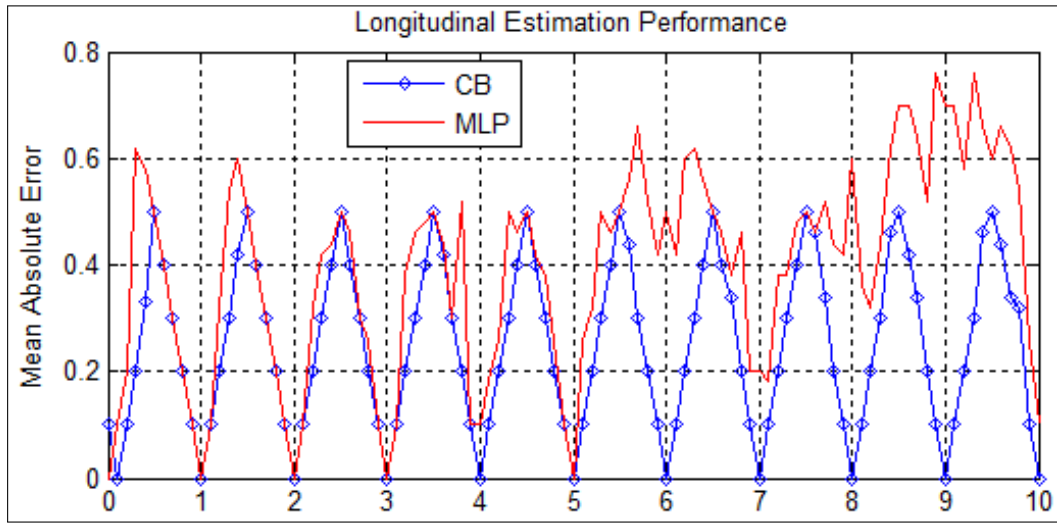


Figure 8.10. Comparison of CB and MLP by using BRM. Pitch angle (β) in degrees

Gaussian noise (AWGN) values on the pixel locations of road lines. A similar noisy road line detections can be seen in Figure 8.16. Additive noise usage for performance test is also used in one of the important computer vision paper [27]. The corresponding performance in terms of Mean Absolute Error, which is the average of all α classes, is given in Figure 8.11. It is seen that proposed algorithm CB method outperforms MLP. Although, CB method is a supervised learning algorithm it can also be regarded as an analytical method as compared to MLP since CB method suppresses noisy components with Principal Component Null Space analysis and it uses the argument of being perpendicular of null space and column space of covariance matrix. The worst performance is obtained by GB since it only uses one pixel value for slope estimation and error on particular pixel makes the method less robust against the noise.

For lateral and longitudinal slope estimations, the performance comparison between our three proposed methods and MLP are given in Figure 8.12 and Figure 8.13 for the virtual environment generated by using CarMaker software. It is a controlled environment with curvature, whose radius is $R=1000\text{m}$. In Figure 8.9 there exists only lateral slope on the curve and in Figure 8.10 there exists only longitudinal slope on the curve. Class differences in CB method and histogram bin sizes in LFB method are both 0.5° . For both α and β cases, LFB method has the worst performance since in this test setup there coexists a curve (θ) and local features performance may degrade in virtual images. Additionally, SIFT based

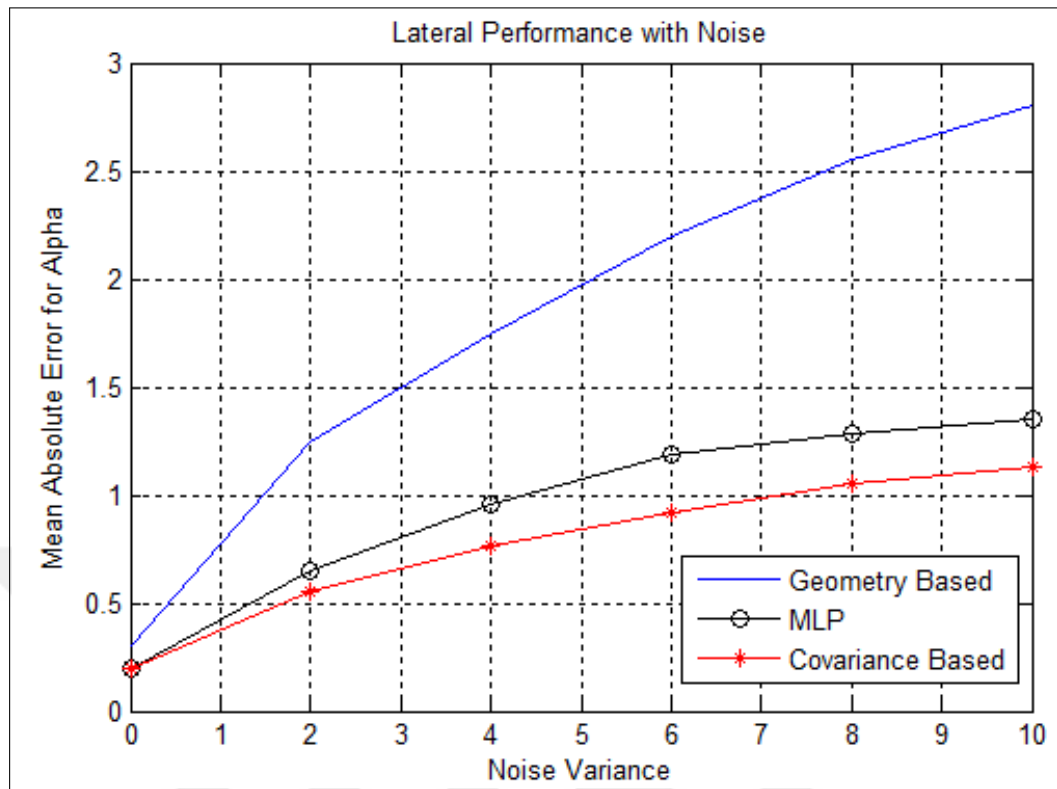


Figure 8.11. Lateral slope estimations on a curve ($R=1000m$) with noise over BRM

method's performance gets worse with increasing road slope performance. CB method has similar performance for both cases, and its maximum estimation error is 0.25° . CB method and MLP have same performance for lateral slope, however CB method outperforms MLP for longitudinal slope. This conforms with Figure 8.9 and Figure 8.10. GB results are better in longitudinal slope than lateral one since finding the correct tangential point is not easy in low resolution image for lateral slope. For longitudinal slope case, GB and CB methods are comparable, but for lateral slope case, the performance of CB method is clearly better than GB method because of the previously mentioned finding correct tangent point problem.

Longitudinal slope is further estimated for the real road as an uncontrolled environment shown in Figure 8.14 and Figure 8.15. The images are captured on on A67 Highway from Lorsch to Mannheim in Germany. In the figures, a sensor for pitch (on the left of sensor) and roll (on the right of sensor) measurement are seen. The camera is located at 1.2m height and the car is driven at a constant speed 100kph on the highway A67 in Germany from Lorsch to Mannheim cities. In the real road test, lateral estimation can not be given because the

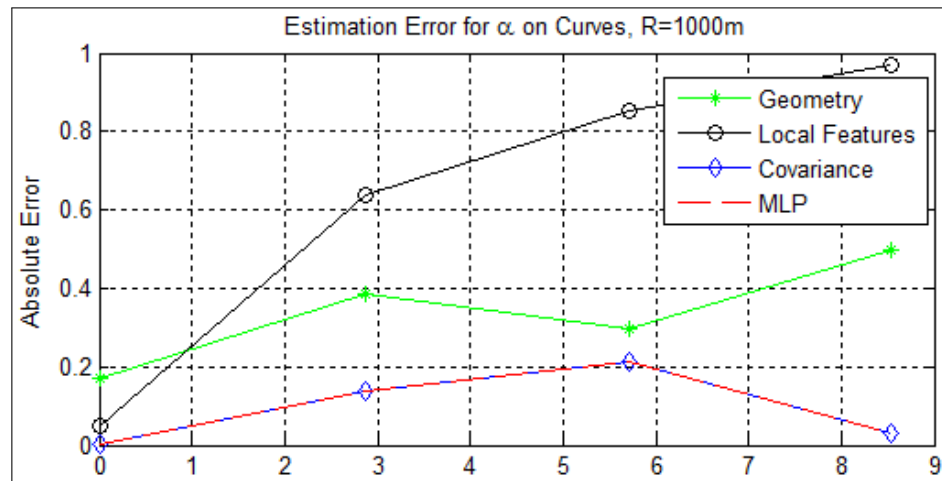


Figure 8.12. Comparison of methods for lateral slope (All Methods, CarMaker)

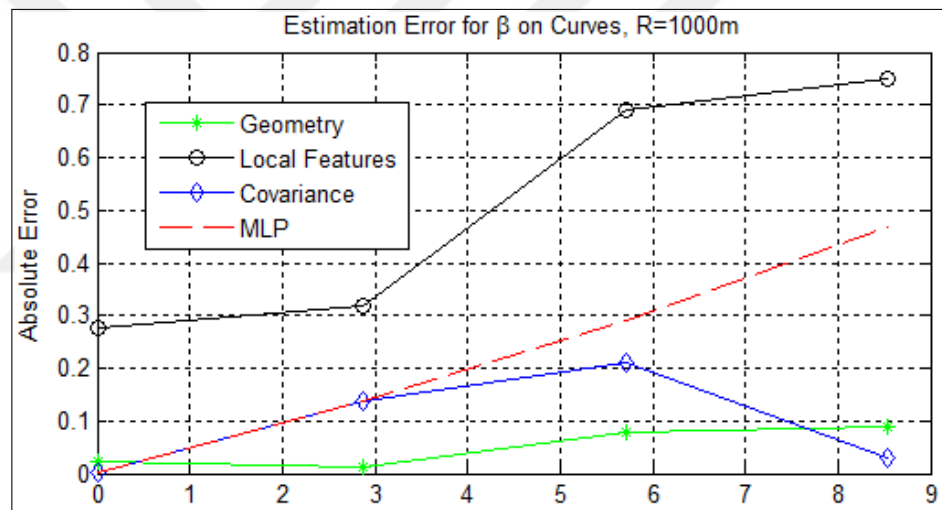


Figure 8.13. Comparison of methods for longitudinal slope (All Methods, CarMaker)



Figure 8.14. Before the longitudinal slope

used sensor is highly affected by centrifugal forces on curves. Estimation results and errors in parenthesis are given in Table 8.1. The real sensor reading 0.86° is used as a benchmark in the first cell of the table. Bin size for CB and MLP is 1° and they yield better results since they find the second bin between $(0.5^\circ, 1.5^\circ)$ and sensor reading 0.86° is close to that bin center. Note that in real road settings LFB method has better performance as compared to the Carmaker images. The reason is that Carmaker provides virtual road image and SIFT points in Carmaker don't occur as good as in real road image. GB and LFB results are comparable and all results are fine since maximum error is about 0.2° . Another important benefit is that CB and MLP can be trained by road model lines and used in testing for real environment.

Table 8.1. Longitudinal estimations for real road in Figure 8.14

Sensor	Geometry	Local Features	Covariance	MLP
0.86°	$0.67^\circ(0.19^\circ)$	$0.66^\circ(0.20^\circ)$	$1^\circ(0.14^\circ)$	$1^\circ(0.14^\circ)$



Figure 8.15. On the longitudinal slope

8.2. RESULTS USING CLOTHOID RM

We assume a calibrated monocular camera and left and road lines are extracted in the image. Additive White Gaussian Noise (AWGN) is added on both dimensions u-axis and v-axis as seen in Figure 8.16, where line detections for 40dB and 50dB are meaningful. Therefore we can call SNR region after 40dB as operating conditions. AWGN can represent calibration and line detection errors [27]. Because of difficulty of having Ground Truth for road parameters, it is very trustworthy to use synthetic images or simulated road lines as done in stereovision [52, 53]. Therefore, our testing environment is simulated 2D road lines based on 3D clothoid road model with addition of AWGN. We give Covariance Based (CB) and Artificial Neural Network (ANN) results for comparisons with ILSO results in Section 8.2.3.

For a better parameter precision, we use close ranges till 60m

$$5m \leq Z_l[1], Z_r[1] \leq 10m \quad (8.2)$$

$$50m \leq Z_l[N_l], Z_r[N_r] \leq Z_{max} = 60m$$

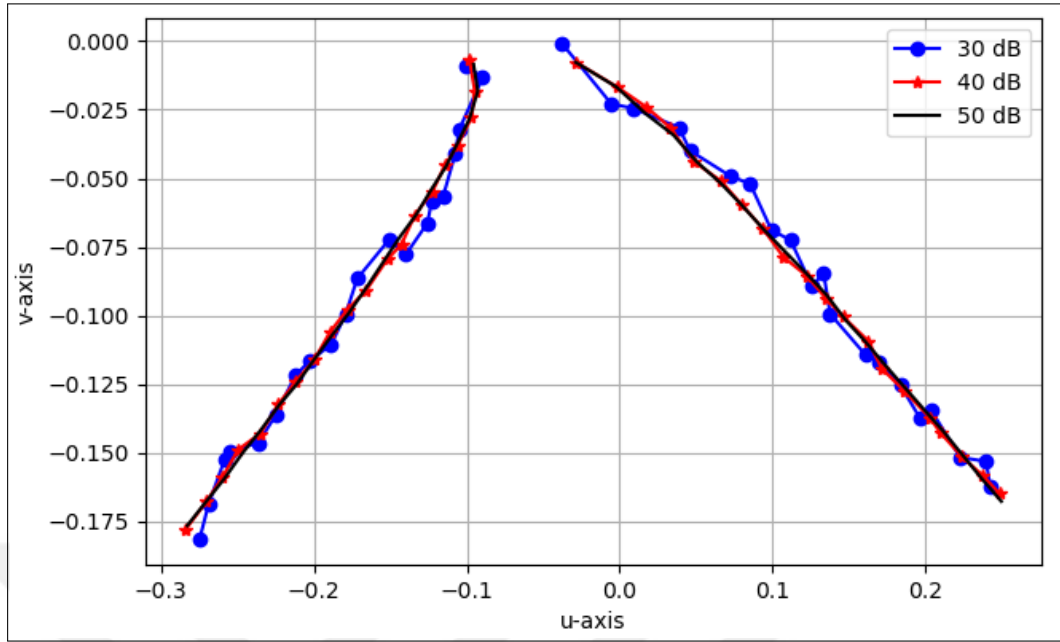


Figure 8.16. Road lines with $f = 1$ and different noise variances (30dB, 40dB, 50dB)

for each image, where road parameters are assumed to be fixed. That does not mean any flat plane assumption, but it is a fixed 3D clothoid surface. For the first chunk of 50m in another image, there can be slightly changes in values, which can be estimated similarly. There can be intersected regions between frames depending on vehicle speed that can help our estimation as an input for the next image.

We calculate distance values (Z) on both lines and Z_l and N_l , Z_r and N_r change parallel. That means, if initial found range is less than 50m, we can increase sample size N_l or N_r to be in (50m,60m). For the opposite case, if initial found range is bigger than 60m, we can reduce sample size to be in (50m,60m). We choose $Z_{max} = 60m$ to compromise between accurate results and high enough range. We use vertical parameter (v) in optimization step, therefore it is a good assumption that error in optimization step does not grow much till Z_{max} under high vertical curvature. On a planar surface under low vertical curvature, high range choice would be possible.

We create $Z_l[n]$ as to sample in equal distance on v -axis. The corresponding right line can

be generated as

$$Z_r[n] = Z_l[n] + c_U Z_l[n]U \quad (8.3)$$

, where U is uniform random variable with distribution between $(-0.5, 0.5)$ and constant is chosen as $c_U = 0.1$. Therefore, we don't assume left and right lines are matched. That creates more generic and realistic situation. With (8.3), we don't assume left and right pixels have same distance, if road lines are sampled for the same v-axis value. Note that the proposed algorithm (ILSO) even does not need the correlation condition in (8.3). ILSO can work under the case where Z_l and Z_r are completely independent.

8.2.1. Curvature Requirements on Highways

According to US Department of Transportation [88], corresponding radius of lateral and vertical curvatures cannot be less than 780m. Therefore we make road parameters comply with this requirement. By considering the plane curves $X(Z)$ and $Y(Z)$ as a function of Z in (3.2), radius of curve can be defined as

$$\begin{aligned} R = 1/\kappa &= \frac{(1 + (Y'(Z))^2)^{1.5}}{|Y''(Z)|} \\ &= \frac{(1 + (a_v + 2b_v Z)^2)^{1.5}}{|2b_v|} \end{aligned} \quad (8.4)$$

, where κ is curvature, which is inverse of radius (R). Similar equation to (8.4) can be derived for $X(Z)$. Both $X(Z)$ and $Y(Z)$ have first and second derivatives of Z . This requirement helps the road model in 3D interpolates different road structures by suppressing higher variation of road lines.

To satisfy the requirement in [88], we set the ranges of parameters by the help of (8.4)

$$\begin{aligned} -0.1 &\leq a_h, a_v \leq 0.1 \\ -0.001 &\leq b_h, b_v \leq 0.001 \end{aligned} \tag{8.5}$$

For other parameters, we set the ranges in (8.6).

$$\begin{aligned} -5^\circ &\leq \alpha \leq 5^\circ \\ 0.1m &\leq d_l, d_r \leq 2.5m \end{aligned} \tag{8.6}$$

All parameters above in eqs. (8.5) and (8.6) get their values from random uniform distribution in their defined intervals.

8.2.2. System Internal Dynamics

Estimation performance is shown in logarithmic scale Figure 8.17 implementing 100k Monte Carlo simulations under additive independent Gaussian noise (AWGN). Each simulation is an independent image frame. Mean Absolute Error (MAE) for distance ($E[Z_{l,r}^e] = E[|Z_{l,r} - \hat{Z}_{l,r}|]$) and MAE for lane position ($E[d_{l,r}^e] = E[|d_{l,r} - \hat{d}_{l,r}|]$) between true values and estimates are depicted in the figure. The parameters are chosen from an uniform distribution in their specific intervals. Number of iterations is varying and defined in (6.17). Logarithmic MAE values are depicted with respect to the different Signal to Noise Ration (SNR) values on *x-axis*. Blue lines represent 20 left and 15 right (#20,#15) and red lines represent 40 left and 30 right samples (#40,#30). As expected, red lines have better performance than blue lines as more noise reduction in LS. Inside each color group, left estimations better than right ones since they have more samples than right ones. As an overall observation higher signal power gives better estimates and at 60dB all estimates are less than 1cm error.

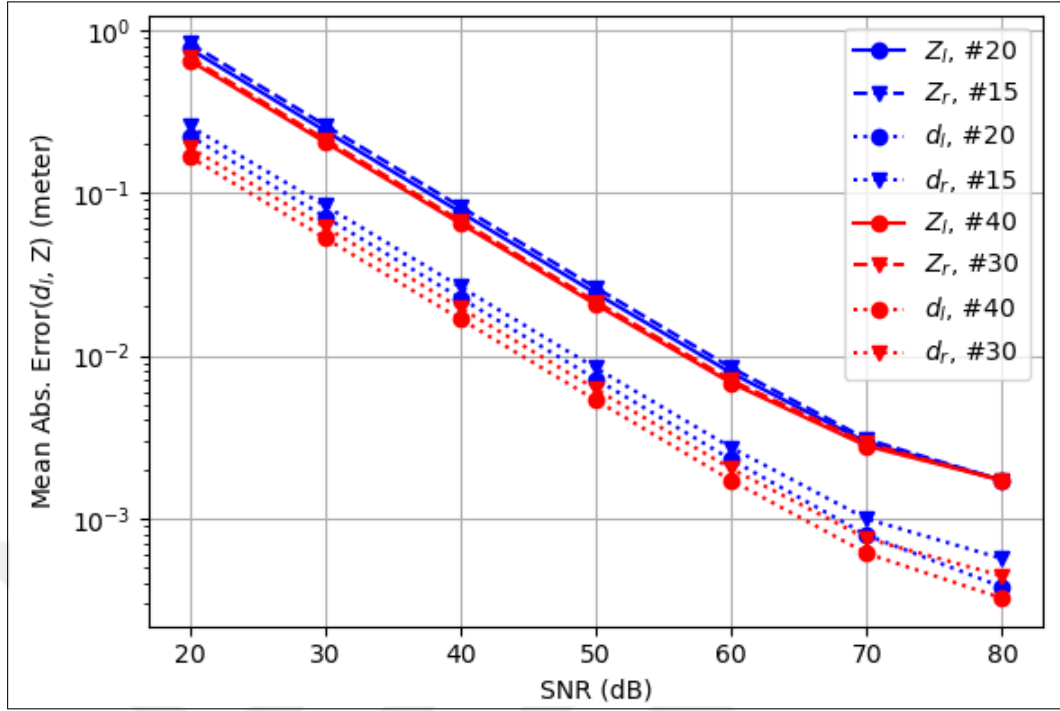


Figure 8.17. Distance (Z_l, Z_r) and position (d_l, d_r) estimations over SNR

Average number of iterations for different sample cases over SNR is seen in Figure 8.18. At 50dB the sample case (#60, #45), in which left line has 60 pixels and right line has 45 pixels, has iteration of 4.4215. Similarly the sample case (#40, #30) has 4.65 and (#20, #15) has 5.07 mean iterations. We can say that all cases have 5 integer number of iterations and higher number of samples may have slightly less iteration values. An important observation is that iteration value does not change much through SNR. Even for low SNRs, algorithms converge to parameters' values, which may be wrong values, with small fluctuation.

8.2.3. Comparison of Methods in Simulations

In this section, we compare our results with other algorithms MultiLayer Perceptron (MLP) in Artificial Neural Network (ANN) and Covariance Based (CB) method developed in Section 5 using our simulations.

For MLP and CB training, all 7 parameters are divided by 10 class bins equally in their defined region in (8.5) and (8.6). Therefore, there are 10^7 (10M) training vectors for both algorithms. Each vector is collected from 20 left pixels and 20 right pixels. Considering

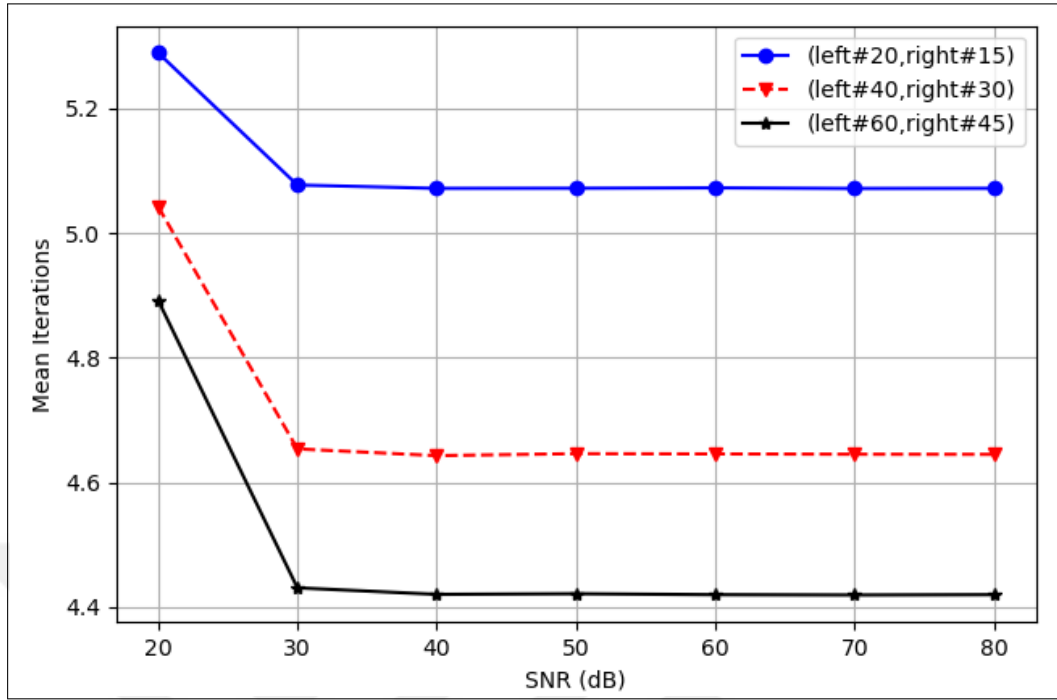


Figure 8.18. Mean number of iterations for different number of samples over SNR

u-axis and v-axis, each vector has length of 80.

For MLP there is one input layer of 128 perceptrons and 2 hidden layers with 64, 64 perceptrons respectively. The output layer has 7 neurons for 7 unknown road parameters. Briefly the structure can be given as MLP(128,64,64,7) as 4 layers. First 3 layers use *Rectified Linear Unit (ReLU)* activation function and output layer uses *Linear* activation function. In Keras *compile* method, *MSE* is used for *loss function* and *Adam* is used for *optimizer*. Then, the Keras model is trained with *fit* method with the parameters *epochs* = 10 and *batchsize* = 10 over 10 Million training vectors.

Testing is done over 10^5 (100k) Monte Carlo simulations for both algorithm. Each simulation generates both road lines as appeared in 2D frames, which are independent frames for each simulation. Parameter values are drawn as uniform distribution in their defined intervals.

In Figure 8.19, we compare distance to the left line (d_l) with MLP and CB. As seen, ILSO is better in all SNR values. Both CB and MLP encounters a saturation in high SNR

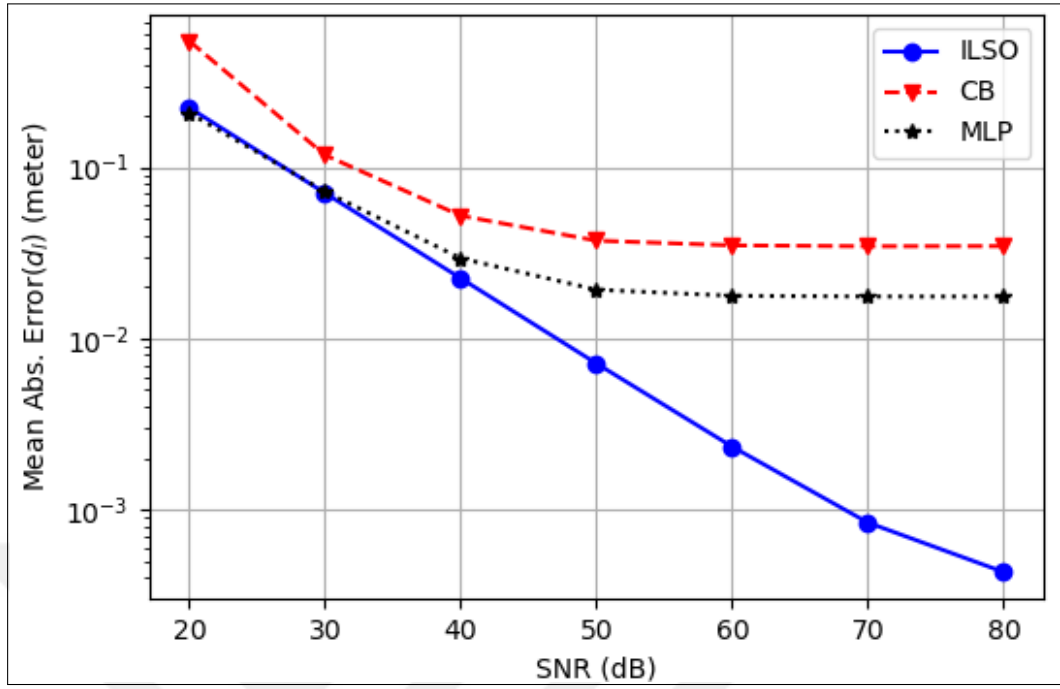


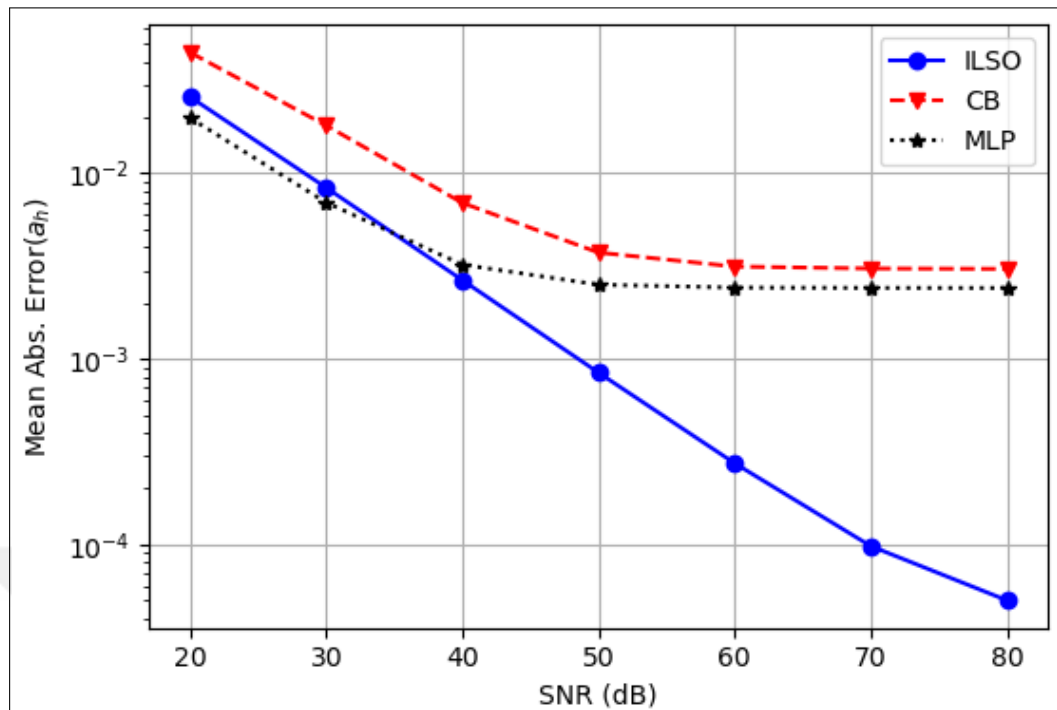
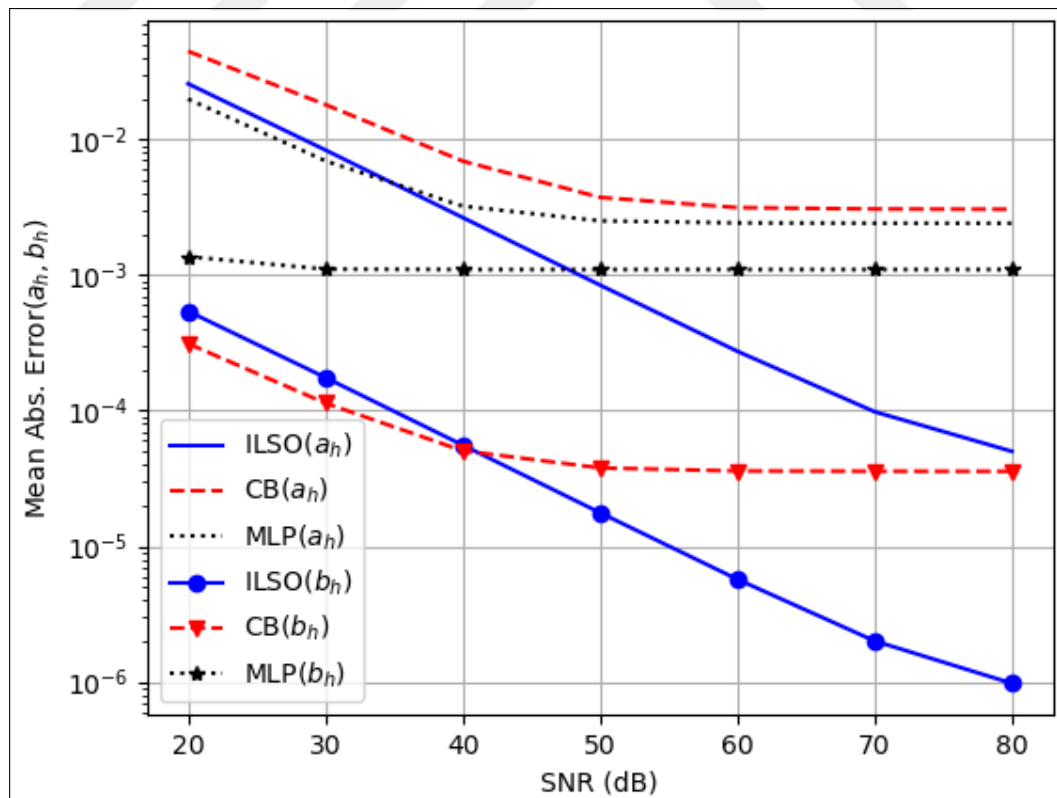
Figure 8.19. Distance to the left line (d_l) estimation

values. Since d_l and d_r have same characteristics in road model and almost same results, we represented only the results for d_l .

Similarly, we compare lateral angle (α) with MLP and CB in Figure 8.20. The proposed algorithm starts to pass other algorithms after 40dB. Both CB and MLP suffer saturation in higher signal quality. If we compare the results with previous results in Figure 8.11, which was plotted for the hard estimations of Basic RM based on classification bins over the possible parameter values, soft estimations of MLP is better than CB, although CB was slightly better than MLP in hard estimation in Figure 8.11. It may be due to following reason; MLP may have better interpolation in soft estimation than CB.

In Figure 8.21, we compare horizontal parameters (a_h, b_h) with CB and MLP. 3 plots without markers are for a_h and 3 others with markers are for b_h . After 40dB ILSO is better than the others that they have saturation again. Although these parameters are small, ILSO continues to find better results in high SNR region.

In Figure 8.22, we compare vertical parameters (a_v, b_v) with CB and MLP. 3 plots without

Figure 8.20. Lateral angle (α) estimationFigure 8.21. Horizontal parameters (a_h, b_h) estimation

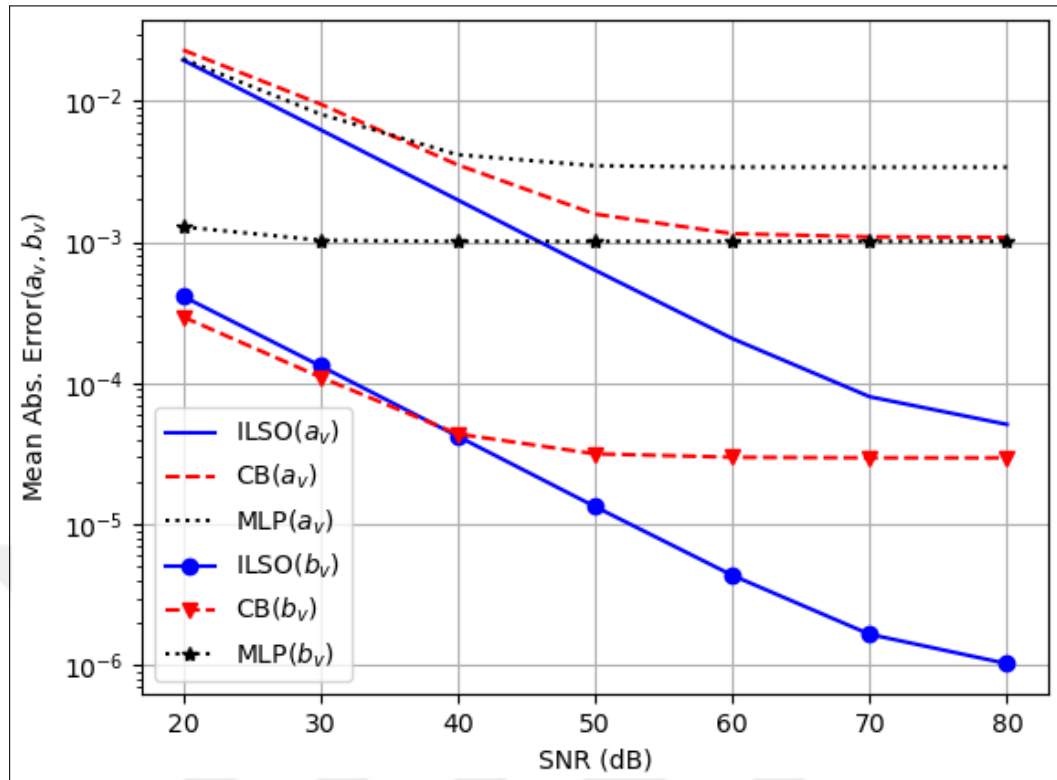


Figure 8.22. Vertical parameters (a_v, b_v) estimation

markers are for a_v and 3 others with markers are for a_v . Beginning with 40dB, ILSO is better for both parameters than other two algorithms. CB and MLP have again saturation for smaller noise conditions, however ILSO does not suffer from this phenomenon.

As seen from the figures CB and MLP have saturation for all parameters in high SNR. The reason is following: Both CB and MLP are trained for discrete values of the parameter. In the testing time piecewise linearization process, which is not optimal, is used between discrete values. Weights of discrete values, which are used for linearization (interpolation), does not optimally represent infinitely many values between class bins. Therefore, CB and MLP are not perfect estimators instead they are predictors based on reference data in training time.

8.2.4. Virtual and Real Images Results

After verification and analysis of the ILSO algorithm in simulation environment, we give results for virtual images, which is produced by CarMaker, and for real images, which are taken by monocular camera in Samsung Galaxy S6 mobile phone. Both have the HD resolution of 1280x720 (horizontal by vertical) and both are located at height from ground at $Y_c = 1.2m$.

Because of difficulty of having Ground Truth for road parameters, it is very trustworthy to use synthetic images and road lines as done in [52, 53]. CarMaker can create a controlled environment (traffic scenes) as seen in Figure 8.23 and Figure 8.24, where road parameters can be arranged inside the program. (Note that CarMaker may use different road model in the program.) That can help verification of vision algorithms. Equivalent focal length is found as $f = 1980$ pixel for virtual images. Both road lines are sampled inside the lane such that right highest gradient of left line and left highest gradient of right line are used as input to the algorithm. For road line detection a gray image can be used. Car on the left lane in Figure 8.23 is located at 30m and car on the middle lane in Figure 8.24 is located at 100m. The last samples are approximately at 60m. The last pixel's vertical position should be greater than 360 that makes v-axis value less than 0 in Figure 3.6d. Algorithm takes 2D pixel location pairs, finds 3D road parameters and based on 3D parameters it converts to 2D image back as seen blue (left) and red (right) lines. Therefore, drawn lines are not the detected road lines, but they are the projection of 3D parameters onto image by using Pinhole Camera model.

Performance of the ILSO method can be found in Table 8.2 for virtual images. The parameters d_l, d_r, α can be compared with CarMaker settings as seen in the table. However, since CarMaker uses its own road model, we can use visual comparison on road lines as seen in Figure 8.24 and Figure 8.23 for horizontal parameters a_h, b_h and vertical parameters a_v, b_v . That shows, our horizontal and vertical parameters converges visually to CarMaker's parameters.

For the real images, Samsung Galaxy S6 mobile phone is attached on windshield using

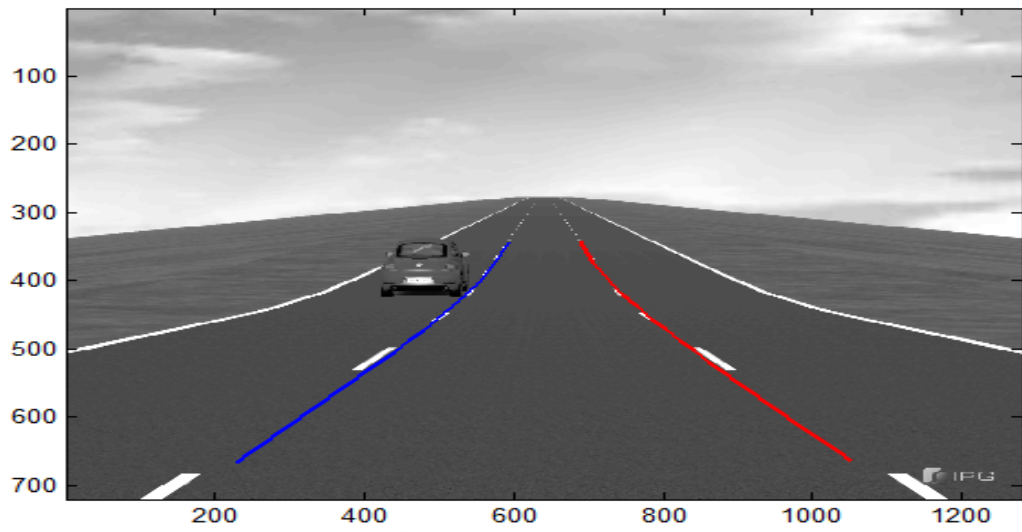


Figure 8.23. Longitudinal slope estimation for 5 percent (2.862°) after 30m

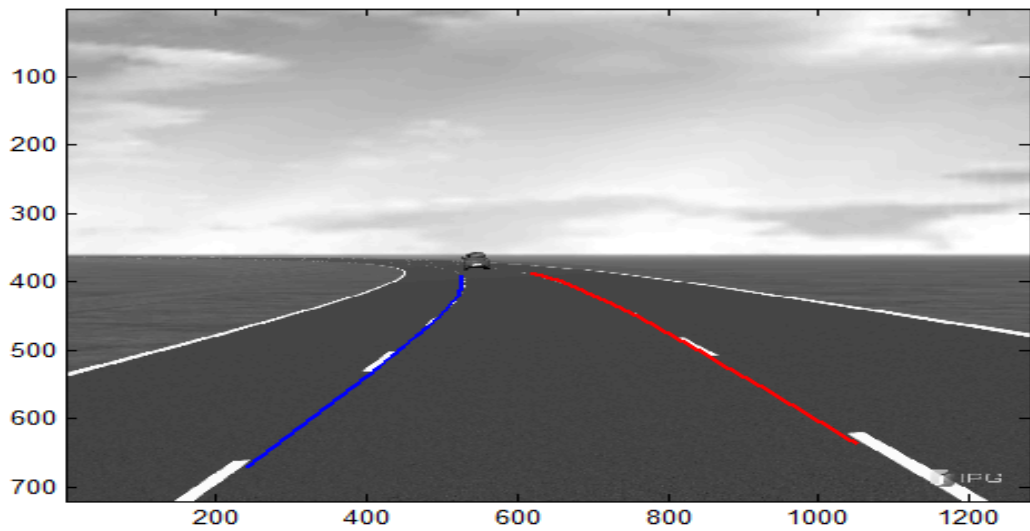


Figure 8.24. Lateral slope estimation for 5 percent (2.862°) in the curve

Table 8.2. Estimated parameters for synthetic images

Figures	CarMaker	Estimation	CarMaker	Estimation	CarMaker	Estimation
	$d_l(m)$	$\hat{d}_l(m)$	$d_r(m)$	$\hat{d}_r$	$\alpha(^{\circ})$	$\hat{\alpha}(^{\circ})$
Figure 8.23	1.65	1.65	1.65	1.64	0	-0.056
Figure 8.24	1.65	1.57	1.65	1.67	-2.862	-2.406



Figure 8.25. A straight road A656 from Mannheim to Heidelberg

a holder. It has a monocular rear camera, which can produce HD images with 1280x720 pixels. Its equivalent focal length is found as $f = 950$ pixel. First a calibration image is created on a laterally and longitudinally straight road, which is visually proven road as seen in Figure 8.25 at the world coordinates (49.431828, 8.609685). Blue (left) and red (right) lines are drawn by using estimated 3D road parameters, which are called calibration parameters show in Table 8.3. The calibration parameters will be subtracted from the parameters in test images to remove unknown rotation issues of camera in holder. That is an easy calibration especially for orientation changes in α, a_h, a_v that may result in different positions of camera in its holder. Note that distance to left (d_l) and right line (d_r) are not used for calibration.

Table 8.3. Calibration parameters of Figure 8.25

$d_l(m)$	$d_r(m)$	$d(m)$	$\alpha(^{\circ})$	a_h	b_h	a_v	b_v
1.91	1.63	3.54	0.038	-0.1155	-2.7691	-0.0820	-0.0001

Third column in Table 8.3 is lane width as $d = d_l + d_r$ in meters. After having calibration image and its parameters, we tested our algorithm in different scenarios as seen from Figure 8.26 to Figure 8.28. The corresponding parameters results for real test images are found in Table 8.4.

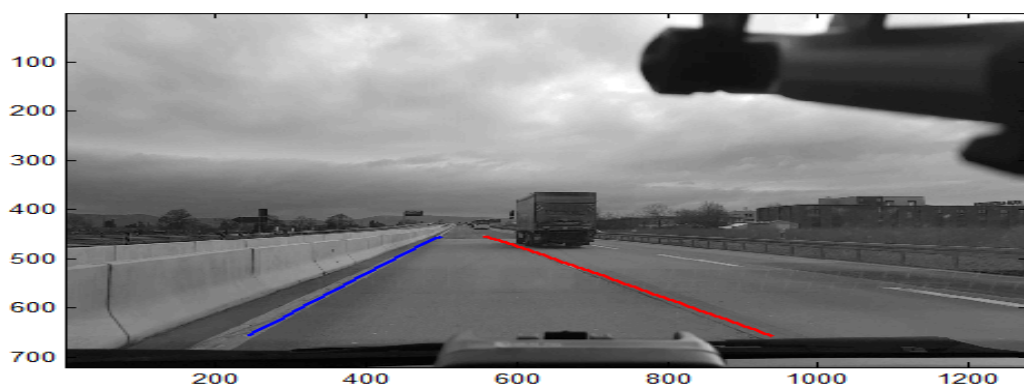


Figure 8.26. Straight left lane on uphill

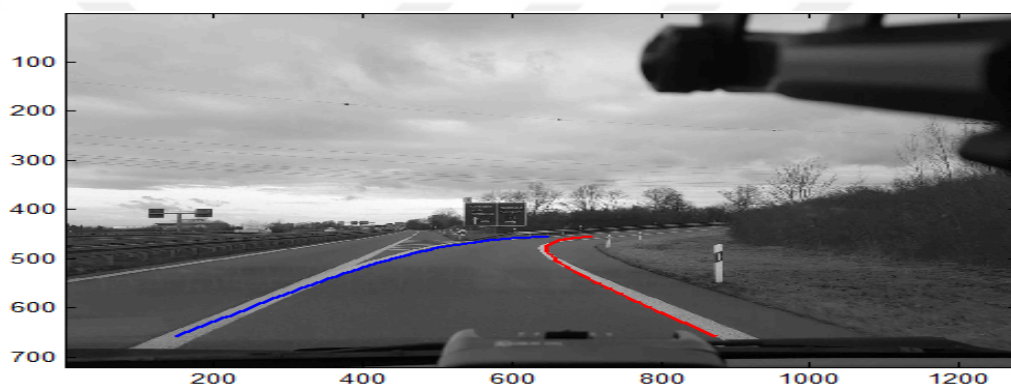


Figure 8.27. Right highway exit on uphill

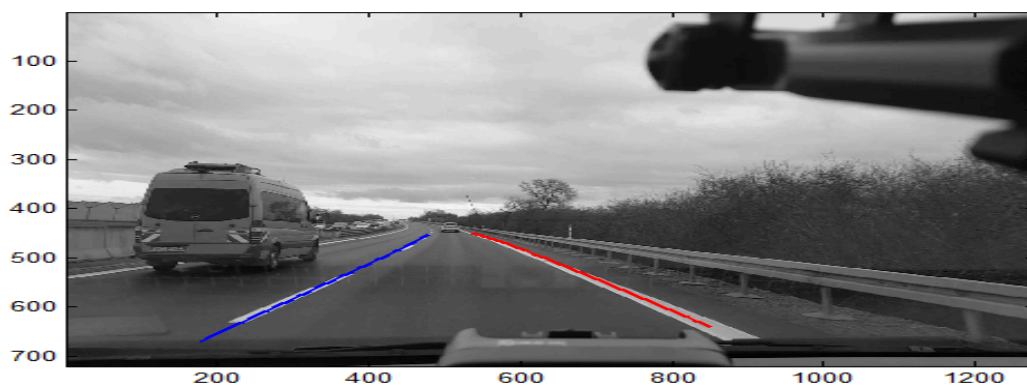


Figure 8.28. Left banked curve on uphill

Figure 8.26, Figure 8.27 and Figure 8.28 are recorded at the coordinates (49.447088, 8.577402), (49.460169, 8.541563), and (49.438420, 8.598464) respectively on German highway A656 from Mannheim to Heidelberg.

Table 8.4. Estimated parameters for real test images

Figures	$d_l(m)$	$d_r(m)$	$d(m)$	$\alpha(^{\circ})$	a_h	b_h	a_v	b_v
Figure 8.26	1.46	2.19	3.66	-0.06	-0.0084	3.4e-5	0.0095	1.3e-5
Figure 8.27	1.91	1.89	3.8	0.53	-0.0331	0.0032	0.0083	1.5e-5
Figure 8.28	1.81	1.73	3.55	-0.45	-0.0067	-0.0002	0.0118	2.7e-5

For the distance at 50m, using the equation $\Delta Y = a_v 50 + b_v 50^2$ differential height of the roads in figures will be 0.51m, 0.46m, and 0.66m respectively. That means after 50m, vehicle will climb about 0.51m, 0.46m, and 0.66m on Y axis. All correspond to about 1.02 percent, 0.92 percent and 1.32 percent uphill slope using the formula $100\% \Delta Y / 50$.

Considering the b_h values in Table 8.4, we can understand that first row is straight road since $|b_h|$ is very small, second row is a right turn since $b_h > 0$ and third row is left turn since $b_h < 0$. Similarly, lateral angle (α) complies with b_h so that left turn has negative α value and right turn has positive α value.

Lane widths (d) can be verified by Google map, on which one can measure distance of a lane. We measured it as 3.56m, 4.03m, and 3.59m respectively by using Google map for test images. In general lane width on German highways is about 3.5m [89]. Note that lane width can be greater on exit lanes.

9. DISCUSSIONS

We give discussions about our results and working conditions. The overall summary is given in Table 9.1. Analytic solution is a mathematical solution based on a model. Explicit solution, which has a closed form expression, is able to leave the parameter alone in the equations. Artificial neural network (MLP) solutions are considered as data driven techniques that teach the network how to decide based on vast amount of data. MLP is given for comparison purpose.

Clothoid RM is used as second and new road model that is more generic than Basic RM. CRM reflects real road into a model successfully. Therefore, it is our preferred model.

GB method strictly depends on road model that uses BRM. It works on perfect straight or perfect circular road and assumes constant lane width, which is 3.5m accepted as in the case of German highways. It is sensitive to noise and requires high resolution for circular curves for a better accuracy.

CB results are the best as shown in Section 8.1 that uses BRM and classification type estimation. It can be easily adapted to CRM as our improved results for a continuity of our work. CB does not need constant road width. In thesis, we interpolate between classes for a continues estimation by using two closest distances to the class bins. Although CB is a data driven method, it is considered as analytic solution since it uses the rule of being perpendicular of column space and null space of the covariance matrix.

ILSO is newly developed by us using CRM and it outperforms CB and MLP results. ILSO can work on only one road line or on both road lines. The work in literature is biased to use both lines, however our simulations show that only one road line can be enough to estimate parameters. That means, occlusions on road lines are welcome as a result of different pixels on each road line. It does not require training and its test time 3rd best in Table 9.2. Therefore, ILSO is the best among all 5 algorithms. In addition to road parameters, ILSO finds longitudinal distances to the pixels on road lines. That helps finding

distances of vehicles by using the information on lines. If object is close to left or right line, the corresponding distance on that line can be used.

There is a small restriction for ILSO that we kept the range till 60m in case of high vertical change. If we find vertical parameters (a_v, b_v) especially b_v are high, we can reduce the range for higher accuracy for high vertical curvatures. In the same way, if the (a_v, b_v) pairs are small, we can increase range for distant parts of the road. We chose to use vertical parameters in optimization step by assuming that lateral curvatures exist higher likely than vertical curvatures. If effect of (a_h, b_h) pairs on horizontal curvature is smaller than the effect of (a_v, b_v) pairs on vertical curvature, we can use horizontal parameters in the optimization as well. Another restriction is downhill scenarios, where the road lines behind the hill may not be in visible region. Most important influence on ILSO will be accuracy of distance estimations. It increases accuracy of road parameters estimation. Therefore, distance values coming from stereovision or lidar in a fusion system can help Least Squares parameter estimations directly.

LFB is our only one method that uses different features than road lines. Used features in LFB are SIFT descriptors. However, this method requires special software [60] to build image frames. Therefore, it can not be simulated as easy as road lines. Its performance is the worst between GB, CB and MLP methods Section 8.1.

MLP, which a famous data driven artificial neural network method, is our benchmark to compare our model driven methods. It is worth mentioning that our method ILSO easily outperforms MLP, which is a famous deep learning regression method for given features.

Finally, we compare all 5 algorithms in terms of complexity in time, which is important for real time operations, in Table 9.2. The table values represent the time per parameter. We run simulation on Python 3.7 installed on 64bit *Microsoft Windows 10 Pro* with hardware Intel Core i7-4510U processor (2GHz, 2 Cores) and 16GB DDR3 L Memory. As seen in the table, GB and ILSO are labeled as Not Applicable since both does not have training step. Since we solve the problem analytically by deriving mathematical equations in Basic RM in GB, testing time is negligibly small. ILSO's time is measured for 5 iterations. LFB actually

Table 9.1. Summary for the methods

Methods	<i>In Advance</i>	<i>Road Model (RM)</i>	<i>Used Features</i>	<i>Analytic Solution</i>	<i>Explicit Solution</i>	<i>Needs Training</i>
GB	Yes	Basic RM	Road lines	Yes	Yes	No
CB	Yes	both	Road lines	Yes	No	Yes
ILSO	Yes	Clothoid RM	Road lines	Yes	Yes	No
LFB	No	Basic RM	SIFT	Yes	Yes	No
<i>MLP</i>	Yes	both	Road lines	No	No	Yes

does not have training step, but we consider getting SIFT descriptors and corresponding matching pairs as training step for consecutive 2 frames.

Table 9.2. Comparison of complexities (in time)

<i>Steps</i>	GB	CB	ILSO	LFB	MLP
<i>Training</i>	NA	101s	NA	2.948s	771s
<i>Test</i>	~0ms	0.179ms	1.0348ms	2.367ms	4.964ms

10. CONCLUSION

In this study, we considered monocular camera located on the vehicle windshield to be used for road slope estimation. We proposed GB, CB, ILSO and LFB road slope or road parameter estimation methods and provided the performance of the methods in both controlled (road line simulations, CarMaker images [60]) and uncontrolled (real road images) environments. Knowing the road parameters can help important ADAS functions such as lane control (Lane Keeping Assistance, LKA) or speed control (Adaptive Cruise Control, ACC). These are basis functions to improve autonomous driving quality.

For GB slope estimation, under certain assumptions on the road model, we derived easy analytic solutions, in which there are no dependencies on focal length, radius and distance as our geometric invariances. Note that GB method works well on straight roads and curves with large radius where exact road lines are provided.

We proposed secondly CB method as a supervised learning algorithm which requires training using the road model and classifies the road slopes into bins or linearizes the estimation between bins as improved solution. Since CB is a learning technique, it does not require a projection model between 3D environment and 2D images in the testing time.

Using the benefits of continuity and low pass character of road lines that complies with lateral and vertical curvature requirements of highways, we implemented a systematic solution (ILSO) to estimate 3D road parameters. Unknown parameters are grouped as road parameters and distances found by Least Squares and optimization respectively. These two group of parameters are found in an iterative manner as improving outputs for each iteration. Algorithm neither requires a specific sequence among parameters nor a specific condition or assumption to find estimates. Proposed algorithm is a direct method and applied for finding the road parameters on left and right lines separately. Even non equal number of samples for left and right lines can be used. Moreover, our method does not require correspondence between left and lines such that pixels on both lines can have different distances. In addition, ILSO finds longitudinal distances of pixels on road lines that can help estimating the distance

of a detected vehicle by using the closest pixel of a line to the vehicle.

LFB based solution using SIFT descriptors is the only one that does not use road lines. We provided easy analytic solutions. LFB may work better in real images due to much more descriptors than in virtual images as seen in Table 8.1. However, in this case getting the ground truth of road parameters in real environment is extremely difficult.

Furthermore, we also compared the performances of the proposed methods with a state-of-the-art artificial neural network method MLP. Our results in Section 8.1 showed that CB results have better performance and easy to implement in different road models. Therefore, we retain CB and MLP methods to evaluate the new method ILSO. ILSO outperforms CB and MLP after 40dB SNR, which can be called as operating conditions. Therefore, ILSO is the most promising solution in generic road conditions.

Each method has its own limitations. GB solutions have limitations of accepting a simple road model. Moreover it needs at least coarse distance to left or right line estimation. For lateral estimation in GB based method requires high resolution. For complex road shapes and under noisy extraction of road lines, GB method performance can deteriorate. CB solutions has a saturation for high SNR regions. ILSO method is limited by slowly changing road in vertical profile. To satisfy high accuracy, ILSO is limited to work in close ranges till 60m to avoid the effect of high vertical curvatures. LFB may be limited because of its time complexity to have SIFT descriptors. LFB is developed for Basic RM, therefore there must be a mapping between slope angles in Basic RM, and road horizontal and vertical parameters in Clothoid RM. In addition, LFB usage is limited to use special software e.g. CarMaker [60], which virtually plots 2D view of the driving environment. Illumination change between day and night or in tunnels can be general limitations for algorithms

Synthetic images are becoming more realistic as shown in [58] and can be used extensively in future. Virtual images can be created by a controlled environment using a software. Technically that is the easiest solution for road parameter estimation problem to have ground truth and train CNN. However, software licence cost sets a crucial limitation to have it. The most important limitation occurs in lack of real reference data for comparison. Majority

of reference data in computer vision is for object detection and road line detection. For example, popular datasets in [62, 63, 64, 66, 67] provide only pixel road area in their ground truth images for drivable road detection. They don't provide 3D parameters of roads. In addition, on board mechanical sensors e.g. inclinometers, lidars, radars on a vehicle are highly affected by vehicle's lateral and longitudinal acceleration and yaw rate. That limits acquiring reference data from other sensors.

All sensors in a vehicle such as lidar, camera, radar, and ultrasonic sensors are affected by the motion of the vehicle. Especially lateral, longitudinal accelerations and yaw rate (angular velocity) will turn vehicle body with respect to earth surface. This phenomenon should be compensated by using odometer signals, which are produced by other systems in a vehicle, as an important future work.

Using optimization for u -axis for high curvature roads can improve ILSO performance as future work. Also, we considered daylight in this study where the performance of the proposed algorithms needs to be tested at night as well. Furthermore, we determined the road slope estimate from a single snapshot road image. By incorporating a series of images, including the road model, and speed of the vehicle, the road slope can be tracked over time.

We used scripting environments such as Matlab and Python. Since mobile applications are used by vast amount of consumers, C/C++ implementation should be done to see algorithms' real time performances as future work. Actual road line detections should be implemented before road parameters estimations in real images. Fusion of algorithms or different sensors GPS, lidar, radar can be added for better 3D road structure estimations.

As mentioned above in limitations, annotations are necessary for real images. A ground truth data can be created using real images as a future work. Horizontal parameters can be extracted by using 2D maps, since horizontal parameters are about bird eye view structure of the road. Samples on 2D map can be fitted to horizontal parameters by using least squares. Distance to left or right line can be manually measured on roads, where there is no traffic. As another way, a vehicle changes lanes from right to left by cross-passing middle lane on a 3-lane highway. That enables to see smooth change on d_l, d_r between consecutive frames.

Obtaining vertical parameters and lateral slope is a tough problem. Datasets, which have lidar information about height for each laser point, can be used. All lidar points can be fitted to a surface, which will have same vertical parameters and lateral slope as in CRM. In this approach, firstly lidar points on road surface should be detected. Then, fitting to a surface is implemented to have labels for vertical and lateral slope. As the parameters are divided in two groups a_h, b_h, d_l, d_r and a_v, b_v, α , they can be annotated and tested separately.



REFERENCES

1. Johansson K. *Road slope estimation with standard truck sensors*. Stockholm: KTH; 2005.
2. Jo K, Kim J, Sunwoo M. Real-time road-slope estimation based on integration of onboard sensors with GPS using an IMMPDA filter. *IEEE Transactions on Intelligent Transportation Systems*. 2013;14(4):1718–1732.
3. Jo K, Lee M, Sunwoo M. Road slope aided vehicle position estimation system based on sensor fusion of GPS and automotive onboard sensors. *IEEE Transactions on Intelligent Transportation Systems*. 2016;17(1):250–263.
4. Liao X, Huang Q, Sun D, Liu W, Han W. Real-time road slope estimation based on adaptive extended Kalman filter algorithm with in-vehicle data. *29th Chinese Control And Decision Conference (CCDC)*; 2017. p. 6889–6894.
5. Kim M, Kim B, Kim C, So M, Lee G, Lim J. Vehicle dynamics and road slope estimation based on cascade extended Kalman filter. *2018 International Conference on Information and Communication Technology Robotics (ICT-ROBOT)*; 2018. p. 1–4.
6. Tareen SAK, Khan HM. Novel slope detection and calculation techniques for mobile robots. *2nd International Conference on Robotics and Artificial Intelligence (ICRAI)*; 2016. p. 158–163.
7. Boniolo I, Tanelli M, Savaresi SM. Roll angle estimation in two-wheeled vehicles. *2008 IEEE International Conference on Control Applications*; 2008. p. 31–36.
8. Jo K, Lee M, Sunwoo M. Road slope aided vehicle position estimation system based on sensor fusion of GPS and automotive onboard sensors. *IEEE Transactions on Intelligent Transportation Systems*. 2016;17(1):250–263.

9. Han J, Kim D, Lee M, Sunwoo M. Enhanced road boundary and obstacle detection using a downward-looking LIDAR sensor. *IEEE Transactions on Vehicular Technology*. 2012;61(3):971–985.
10. Yalcin O, Sayar A, Arar OF, Apinar S, Kosunalp S. Detection of road boundaries and obstacles using LIDAR. *2014 6th Computer Science and Electronic Engineering Conference (CEECE)*; 2014. p. 6–10.
11. Zhang W. LIDAR-based road and road-edge detection. *IEEE Intelligent Vehicles Symposium*; 2010. p. 845–848.
12. Li Q, Chen L, Li M, Shaw S, Nüchter A. A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios. *IEEE Transactions on Vehicular Technology*. 2014;63(2):540–555.
13. Dieterle T, Particke F, Patino-Studencki L, Thielecke J. Sensor data fusion of LIDAR with stereo RGB-D camera for object tracking. *2017 IEEE SENSORS*; 2017. p. 1–3.
14. Sun P, Zhao X, Xu Z, Wang R, Min H. A 3D LiDAR data-based dedicated road boundary detection algorithm for autonomous vehicles. *IEEE Access*. 2019;7:29623–29638.
15. Yan F, Wang K, Zou B, Tang L, Li W, Lv C. LiDAR-based multitask road perception network for autonomous vehicles. *IEEE Access*. 2020;8:86753–86764.
16. Sivaraman S, Trivedi MM. Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. *IEEE Transactions on Intelligent Transportation Systems*. 2013;14(4):1773–1795.
17. Zehang Sun, Bebis G, Miller R. On-road vehicle detection: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2006;28(5):694–711.

18. Li B, Yao Q, Wang K. A review on vision-based pedestrian detection in intelligent transportation systems. *Proceedings of 2012 9th IEEE International Conference on Networking, Sensing and Control*; 2012. p. 393–398.
19. Nadav I, Katz E. Off-road path and obstacle detection using monocular camera. *2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE)*; 2016. p. 1–5.
20. Mogelmose A, Trivedi MM, Moeslund TB. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*. 2012;13(4):1484–1497.
21. Xing Y, Lv C, Chen L, Wang H, Wang H, Cao D, et al. Advances in vision-based lane detection: Algorithms, integration, assessment, and perspectives on ACP-based parallel vision. *IEEE/CAA Journal of Automatica Sinica*. 2018;5(3):645–661.
22. McCall JC, Trivedi MM. Video-based lane estimation and tracking for driver assistance: Survey, system, and evaluation. *IEEE Transactions on Intelligent Transportation Systems*. 2006;7(1):20–37.
23. Jia B, Chen J, Zhang K, Wang Q. Sequential monocular road detection by fusing appearance and geometric information. *IEEE/ASME Transactions on Mechatronics*. 2019;24(2):633–643.
24. Danescu R, Nedevschi S, Meinecke MM, To TB. Lane geometry estimation in urban environments using a stereovision system. *2007 IEEE Intelligent Transportation Systems Conference*; 2007. p. 271–276.
25. Bouteau R, Savatier X, Bonardi F, Ertaud JY. Road-line detection and 3D reconstruction using fisheye cameras. *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*; 2013. p. 1083–1088.

26. Sivaraman S, Trivedi MM. A review of recent developments in vision-based vehicle detection. *2013 IEEE Intelligent Vehicles Symposium (IV)*; 2013. p. 310–315.
27. Lowe DG. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*. 2004;60(2):91–110.
28. Kyrki V, Kamarainen JK, Kälviäinen H. Simple Gabor feature space for invariant object recognition. *Pattern Recognition Letters*. 2004;25(3):311 – 318.
29. Leutenegger S, Chli M, Siegwart RY. BRISK: Binary robust invariant scalable keypoints. *2011 International Conference on Computer Vision*; 2011. p. 2548–2555.
30. Bay H, Ess A, Tuytelaars T, Van Gool L. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*. 2008;110(3):346 – 359.
31. Nguyen H, Kieu L, Wen T, Cai C. Deep learning methods in transportation domain: A review. *IET Intelligent Transport Systems*. 2018;12(9):998–1004.
32. Kuo T, Lo Y. Depth estimation from a monocular view of the outdoors. *IEEE Transactions on Consumer Electronics*. 2011;57(2):817–822.
33. Liu B, Gould S, Koller D. Single image depth estimation from predicted semantic labels. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*; 2010. p. 1253–1260.
34. Liu F, Chunhua Shen, Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2015. p. 5162–5170.
35. Godard C, Aodha OM, Brostow GJ. Unsupervised monocular depth estimation with left-right consistency. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2017. p. 6602–6611.
36. Stein GP, Ferencz AD, Avni O. Estimating distance to an object using a sequence of

- images recorded by a monocular camera; 2012. Mobileye Vision Technologies Ltd, US8164628B2.
37. Jian MH, Lin H, Wang J. Method of estimating depths from a single image displayed on display; 2012. ArcSoft Corp Ltd, US8284998B2.
 38. Meng Z, Kong X, Meng L, Tomiyama H. Distance measurement and camera calibration based on binocular vision technology. *2018 International Conference on Advanced Mechatronic Systems (ICAMechS)*; 2018. p. 342–347.
 39. Liu Z, Chen T. Distance measurement system based on binocular stereo vision. *2009 International Joint Conference on Artificial Intelligence*; 2009. p. 456–459.
 40. Geiger A, Ziegler J, Stiller C. Stereoscan: Dense 3d reconstruction in real-time. *2011 IEEE Intelligent Vehicles Symposium (IV)*; 2011. p. 963–968.
 41. Zhang K, Chen J, Li Y, Zhang X. Visual Tracking and depth estimation of mobile robots without desired velocity information. *IEEE Transactions on Cybernetics*. 2020;50(1):361–373.
 42. Gao Y, Chen J, Zhang K, Jia B. A 2-point pose estimation algorithm for monocular visual odometry of ground vehicles. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; 2017. p. 1610–1615.
 43. Karaduman O, Eren H, Kurum H, Celenk M. Road-geometry-based risk estimation model for horizontal curves. *IEEE Transactions on Intelligent Transportation Systems*. 2016;17(6):1617–1627.
 44. Chausse F, Aufrere R, Chapuis R. Recovering the 3D shape of a road by on-board monocular vision. *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. vol. 1; 2000. p. 325–328 vol.1.
 45. Aufrere R, Chapuis R, Chausse F. A fast and robust vision based road following

- algorithm. *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511)*; 2000. p. 192–197.
46. Aufrère R, Chapuis R, Chausse F. A model-driven approach for real-time road recognition. *Machine Vision and Applications*. 2001;13(2):95–107.
 47. Tian M, Liu F, Hu Z. Single camera 3D lane detection and tracking based on EKF for urban intelligent vehicle. *2006 IEEE International Conference on Vehicular Electronics and Safety*; 2006. p. 413–418.
 48. Dickmanns ED, Mysliwetz BD. Recursive 3-D road and relative ego-state recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1992;14(2):199–213.
 49. Garnett N, Cohen R, Pe’er T, Lahav R, Levi D. 3D-LaneNet: End-to-end 3D multiple lane detection. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*; 2019. p. 2921–2930.
 50. Nedeveschi S, Schmidt R, Graf T, Danescu R, Frentiu D, Marita T, et al. 3D lane detection system based on stereovision. *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*; 2004. p. 161–166.
 51. Nedeveschi S, Oniga F, Danescu R, Graf T, Schmidt R. Increased accuracy stereo approach for 3D lane detection. *2006 IEEE Intelligent Vehicles Symposium*; 2006. p. 42–49.
 52. Labayrade R, Aubert D. A single framework for vehicle roll, pitch, yaw estimation and obstacles detection by stereovision. *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683)*; 2003. p. 31–36.
 53. Benmansour N, Labayrade R, Aubert D, Glaser S, Gruyer D. A model driven 3D lane detection system using stereovision. *2008 10th International Conference on Control,*

Automation, Robotics and Vision; 2008. p. 1277–1282.

54. Jia B, Chen J, Zhang K. Drivable road reconstruction for intelligent vehicles based on two-view geometry. *IEEE Transactions on Industrial Electronics*. 2017;64(5):3696–3706.
55. Feng YH, Zhang RH, Zhai S. Road elevation map estimation based on affine transformation and stereo matching. *Journal of Physics: Conference Series*. vol. 1601; 2020. p. 062015–062023.
56. Fan R, Ai X, Dahnoun N. Road surface 3D reconstruction based on dense subpixel disparity map estimation. *IEEE Transactions on Image Processing*. 2018;27(6):3025–3035.
57. Evans M, Fan R, Dahnoun N. Iterative roll angle estimation from dense disparity map. *2018 7th Mediterranean Conference on Embedded Computing (MECO)*; 2018. p. 1–4.
58. Gaidon A, Wang Q, Cabon Y, Vig E. Virtual worlds as proxy for multi-object tracking analysis. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2016. p. 4340–4349.
59. Deter D, Wang C, Cook A, Perry NK. Simulating the autonomous future: A look at virtual vehicle environments and how to validate simulation using public data sets. *IEEE Signal Processing Magazine*. 2021;38(1):111–121.
60. Ipg-Automotive GmbH, CarMaker software;. [cited 2020 11 January]. Available from: <https://ipg-automotive.com/products-services/simulation-software/carmaker>.
61. dSpace, ADAS and autonomous driving;. [cited 2021 11 January]. Available from: https://www.dspace.com/en/inc/home/applicationfields/our_solutions_for/driver_assistance_systems.cfm.

62. Yu F, Chen H, Wang X, Xian W, Chen Y, Liu F, et al. BDD100K: A diverse driving dataset for heterogeneous multitask learning. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2020. p. 2633–2642.
63. Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, et al. The Cityscapes dataset for semantic urban scene understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2016. p. 3213–3223.
64. Behley J, Garbade M, Milioto A, Quenzel J, Behnke S, Stachniss C, et al. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*; 2019. p. 9296–9306.
65. Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*; 2012. p. 3354–3361.
66. Aly M. Real time detection of lane markers in urban streets. *2008 IEEE Intelligent Vehicles Symposium*; 2008. p. 7–12.
67. Lee S, Kim J, Yoon JS, Shin S, Bailo O, Kim N, et al. VPGNet: Vanishing point guided network for lane and road marking detection and recognition. *2017 IEEE International Conference on Computer Vision (ICCV)*; 2017. p. 1965–1973.
68. Vaswani N, Chellappa R. Principal components null space analysis for image and video classification. *IEEE Transactions on Image Processing*. 2006;15(7):1816–1830.
69. Ustunel E, Chen X, Schonfeld D, Khokhar A. Null-space representation for view-invariant motion trajectory classification-recognition and indexing-retrieval. *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*; 2008. p. 809–812.
70. Guzel T, Ustunel E. Principal components null space analysis based non-intrusive

- load monitoring. *2015 IEEE Electrical Power and Energy Conference (EPEC)*; 2015. p. 420–423.
71. Ustunel E, Masazade E. Vision-based road slope estimation methods using road lines or local features from instant images. *IET Intelligent Transport Systems*. 2019;13(10):1590–1602.
 72. Strang G. *Introduction to linear algebra*. USA: Wellesley-Cambridge Press; 2016.
 73. Distinctive image features from scale-invariant keypoints. Springer;. [cited 2021 04 January]. Available from: <https://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94>.
 74. Goodfellow I, Bengio Y, Courville A. *Deep learning*. Cambridge, MA: MIT Press; 2016.
 75. Yuksel C. *Bird call detection using deep learning*. Istanbul: Yeditepe University; 2020.
 76. Haykin S. *Neural networks: A comprehensive foundation (2 ed.)*. New Jersey: Prentice Hall; 1998.
 77. Setayeshi S, El-Hawary F, El-Hawary ME. Underwater signal prediction and parameter estimation using artificial neural networks. *Proceedings 1995 Canadian Conference on Electrical and Computer Engineering*. vol. 2; 1995. p. 621–626 vol.2.
 78. TensorFlow - Multilayer perceptron learning;. [cited 2020 25 October]. Available from: https://www.tutorialspoint.com/tensorflow/tensorflow_multi_layer_perceptron_learning.htm.
 79. Kanjee R, Bachoo AK, Carroll J. Vision-based adaptive cruise control using pattern matching. *2013 6th Robotics and Mechatronics Conference (RobMech)*; 2013. p. 93–98.

80. Chenchen L, Fulin S, Haitao W, Jianjun G. A camera calibration method for obstacle distance measurement based on monocular vision. *2014 Fourth International Conference on Communication Systems and Network Technologies*; 2014. p. 1148–1151.
81. Bartlett R. Minimum horizontal radius;. [cited 2020 15 August]. Available from: https://www.researchgate.net/publication/290805520_Minimum_horizontal_radius_rev_05.
82. Haykin S. *Adaptive filter theory*. New Jersey: Prentice-Hall, Inc.; 1986.
83. Kariya T, Kurata H. *Generalized least squares*. New Jersey: Wiley; 2004.
84. Boyd S, Vandenberghe L. *Convex optimization*. Cambridge: Cambridge University Press; 2004.
85. Rao SS. *Engineering optimization theory and practice*. New Jersey: Wiley; 2009.
86. Tareen SAK, Saleem Z. A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*; 2018. p. 1–10.
87. Peng W, Lee M, Li T, Huang C, Lin P. Performance comparison of image keypoint detection, description, and matching methods. *2016 IEEE 5th Global Conference on Consumer Electronics*; 2016. p. 1–2.
88. Roadway design manual - US Department of Transportation;. [cited 2020 15 September]. Available from: http://onlinemanuals.dot.state.tx.us/txdotmanuals/rdw/vertical_alignment.htm#i1086276.
89. Lane width in German highways;. [cited 2021 15 January]. Available from: <http://www.german-autobahn.eu/index.asp?page=design>.