Object classification and tracking for autonomous car navigation in Indian road scenarios

A **PROJECT REPORT**
SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR AWARD OF THE DEGREE OF
BACHELOR OF TECHNOLOGY
by

Athul Zac Joseph(B150473EC)
Ch. Lakshmi Priyanka(B150837EC)
Malavika Nair M(B150581EC)
Namburi GNVV Satya Sai Srinath(B150843EC)
S Umamaheswaran(B150176EC)

under the guidance of

Dr. Praveen Sankaran

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY CALICUT
NIT CAMPUS PO - 673601, KOZHIKODE, KERALA, INDIA
APRIL 2019

Acknowledgement

We would like to extend our sincere gratitude to our project guide Dr. Praveen Sankaran for his support during the entire project. The completion of our project would not have been possible without the valuable suggestions from the evaluation panel. We would like to acknowledge the Electronics and Communication Department for extending its full support during the entirety of the project. We would also like to thank TEQIP for granting us fund under the Innovative UG Projects Scheme for the purchase of necessary components.

Declaration

We hereby declare that except where specific reference is made to the work of others, the contents of this project report are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This project report is our own work and does not contain any outcome of work done in collaboration with others, except as specified in the text and acknowledgement.

Athul Zac Joseph(B150473EC)
Ch. Lakshmi Priyanka(B150837EC)
Malavika Nair M(B150581EC)
Namburi GNVV Satya Sai Srinath(B150843EC)
S Umamaheswaran(B150176EC)

Certificate

This is to certify that the report entitled **Object classification and tracking for autonomous car navigation in Indian road scenarios** submitted by **Athul Zac Joseph(B150473EC), Ch. Lakshmi Priyanka(B150837EC), Malavika Nair M(B150581EC), Namburi GNVV Satya Sai Srinath(B150843EC), S Umamaheswaran(B150176EC)** to National Institute of Technology Calicut for the award of the degree of Bachelor of Technology is a bonafide record of the research work carried out by them under my guidance. The content of the report, in full or parts have not been submitted to any other institute or university for the award of any degree or diploma.

Dr. Praveen Sankaran(Project Guide)
Assistant Professor
Electronics And Communication Engg.
NIT Calicut

Dr. A.V. Babu
Professor and Head of the Department
Electronics And Communication Engg.
NIT Calicut

Place : NIT Calicut
Date : 25/04/2019

(Office Seal)

Abstract

Statistics say that there were over 4,80,652 accidents in 2016, leading to 1,50,785 deaths. Further analysis shows that this is about 17 deaths per hour. One solution to reduce the mortality rate caused by road accidents is to introduce autonomous vehicles which can work with higher accuracy than humans. Wide range of research is going on in areas of object classification,tracking and controlling the motion of vehicles to avoid accidents. The project primarily intends to detect, classify and track objects by a car travelling on road.

1. Developing algorithms on KITTI data-set
2. Creating our own data-set
3. Testing the algorithms on our own data-set

Initial stages of project involves development of algorithms on KITTI data-set. There are less number of data-sets available mimicking Indian road scenarios. A new data-set will be created by mounting cameras on top of a car in such a way that this can be used for further research in the field of autonomous vehicles in Indian road scenarios. At later stages,testing the algorithms with the created data-set will be done.As the performance of algorithms vary from a foreign road scenario to a typical Indian scenario, necessary modifications will be made to improve the performance further.

Contents

List of Figures **xv**

List of Tables **xvii**

1 Introduction **1**

2 Literature review **5**

3 Segmentation of LiDAR point cloud **9**
3.1 Polar Grid mapping 10
3.2 Gaussian Process Regression 12
3.2.1 Central Limit theorem 12
3.2.2 Mathematics 13
3.3 Pointwise Segmentation 15
3.4 Fast Segmentation Algorithm 16
3.5 Segmentation of Non Ground Points 21
3.5.1 Connected Component Labelling using Two Pass Algorithm 22

4 Data Set for Object Detection and Stereo vision for Indian Roads **27**
4.1 Object Detection Dataset 27
4.2 Stereo Dataset 28

5 Real-time object detection **31**
5.1 R-CNN and its variants 31
5.2 Approach used by YOLO 34

5.2.1 Outputs obtained from YOLO v2 36

6 Depth estimation using stereo vision **39**
6.1 Disparity Map 39
6.2 General matching cost computation 41
6.3 MC-CNN by fast architecture 42
6.3.1 Matching cost computation in fast architecture 42
6.3.2 Stereo method 42
6.3.3 SemiGlobal matching: 43
6.3.4 Computing the Disparity image 43
6.4 Results 44
6.5 Validation with 2D LiDAR 45

7 Odometry **47**
7.1 Problem Formulation of Monocular Visual Odometry 48
7.2 Algorithm 48
7.3 Results from KITTI Dataset 50
7.4 Results from NITCOD Dataset 51

8 Velocity Estimation **53**
8.1 Scale Invariant Feature Transform 55
8.2 Result of velocity estimation 56

References **57**

List of Figures

3.1 Partitioning into segments 10
3.2 Partitioning segments into bins 11
3.3 Obstacle point obtained after Gaussian Process Regression 16
3.4 Initial 3D point cloud 17
3.5 Ground points obtained from Fast Segmentation 18
3.6 Non ground obtained from Fast Segmentation 19
3.7 A scenario which depicts the slope road and obstacles 20
3.8 Ground point detection from the slope scenario(Fast Segmentation) . 20
3.9 Ground point detection from the slope scenario(Gaussian Process Regression) 21
3.10 Flowchart of connected component labelling - Pass 1 22
3.11 Flowchart of connected component labelling - Pass 2 23
3.12 CCL on Fast Segmented LiDAR data 24
3.13 CCL on Fast Segmented LiDAR data(close view) 24
3.14 CCL on Fast Segmented LiDAR data(closer view of poles) 25
3.15 CCL on GPR LiDAR data 25
3.16 CCL on GPR LiDAR data(Close view) 26
3.17 CCL on GPR LiDAR data(person) 26

4.1 An example image from object detection dataset 29
4.2 An example image pair taken from Intel Realsense camera 29

5.1 Main difference in architecture of R-CNN and Faster R-CNN 32
5.2 Output of Faster R-CNN 33

5.3 Architecture Of YOLO 34
5.4 Architecture Of YOLO v2 35
5.5 Wrong prediction by YOLO v2. A car is identified as a person . . . 36
5.6 Correct prediction by YOLOv2 37
5.7 Output of YOLOv3 38

6.1 Disparity output from kitti dataset 40
6.2 Geometry for finding disparity 40
6.3 An example image pair taken from Intel Realsense camera 44
6.4 Outputs of Disparity maps obtained by camera, MC-CNN and OpenCV 44
6.5 Isometric view of the room 45
6.6 Left image shows the r-θ plot and right image shows the distance of obstacle vs θ plot with obstacle placed around 250° 45
6.7 Left image shows the r-θ plot and right image shows the distance of obstacle vs θ plot with obstacle placed around 70° 46

7.1 Block Diagram of Monocular Visual Odometry Algorithm 49
7.2 Feature Tracking using KLT Tracker in KITTI 50
7.3 Estimated Trajectory using Visual Odometry in KITTI 50
7.4 Feature Tracking using KLT Tracker in NITCOD Dataset 51
7.5 Estimated Trajectory using Visual Odometry in NITCOD Dataset . 51

8.1 Flow chart describing the steps to estimate the velocity 54
8.2 Velocity estimation on KITTI 56

List of Tables

4.1 Frequency of objects of each class 28

5.1 Different metrics obtained after training Faster R-CNN 32

Chapter 1

Introduction

Autonomous cars can be very useful to avoid accidents on roads by reducing human intervention. This project tries to handle the main challenges in the field of autonomous navigation. Object detection and classification are crucial for autonomous vehicles to ensure safety.

There are several techniques for effective autonomous navigation. These include navigation using ultrasonic sensors, LiDAR (Light Detection and Ranging) systems, preloaded maps, landmarks, vision based navigation etc. While image from a single camera based methods give satisfactory outputs for object detection, they lack a three dimensional view of the object which includes the depth measure. LiDAR systems are perfect tools because of their accuracy and speed but LiDARs are expensive and this becomes their greatest disadvantage. Vision based navigation techniques are intelligent systems that enable machines to detect objects just like humans. As depth information becomes very crucial in autonomous navigation systems, the best suited vision based technique is of stereo vision. Stereo vision gives a three dimensional view of the environment and thus helps tremendously in the decision making process of how objects can be avoided to safely navigate through the environment.

As part of the project, the KITTI image dataset was visualised. By using training image set and the corresponding label text files, 3D bounding boxes were constructed over the image, which highlight objects like cars, pedestrians, cyclists etc. OpenCV library was used for this purpose. The LiDAR point cloud data present in KITTI was also visualised using MayaVi library.

There are two different approaches that were implemented for segmentation of the point cloud. One is based on Gaussian Process Regression which is a statistical approach and the other is based on the geometrical properties of the surface. The ground plane extraction is followed by connected components labelling on the non-ground points for segmentation.
For segmentation of non-ground points, connected components algorithm is applied. The algorithm in its first pass label all the foreground points with a new label if none of its neighbours are labelled else labels the bin with the lowest label value of its neighbour. Also, it makes the neighbour with the lowest label of a bin as the root label. In the second pass, all the labels will be replaced with the value of its root label. In this application since the point cloud became sparse after the ground point removal, 24-nearest neighbours are searched for establishing neighbourhood connections.
In polar grid mapping the entire point cloud is divided into segments and bins based on its angular position in xy plane and distance from the sensor respectively. A set of prototype points are formed by selecting the point with least z coordinate from a non-empty bin.
The Gaussian process algorithm starts with a set of 3D cloud points and it finally ends in labelling each and every point as ground or obstacle. It mainly consists of 4 steps: Polar grid mapping, seed evaluation (Estimation of initial ground), Gaussian Process Regression with non-stationary covariance function and pointwise segmentation (Estimation of final ground).
In fast segmentation algorithm, after extracting the set of prototype points, evaluation of ground points involves formulating simple conditions based on the slope and intercept of the lines formed by joining prototype points that belong to the same segment. Set of all the points which satisfy these conditions forms the reduced set of ground points in that segment. The final set of ground points is removed from the actual point cloud to find the set of non-ground point.
A dataset that is suitable for Indian roads was created that contained different classes like auto, two-wheeler, car, bus etc. The images were annotated using an online tool called Label box. The dataset created was then visualised.
Objection detection algorithms like Faster R-CNN, YOLO(You Only Look Once) are studied in detail and are implemented on our dataset. Their performance speed vs

accuracy is compared.

To get depth information from the stereo images that were collected, fast version of MC-CNN(Matching Cost-Convolutional Neural Network) is applied to the pair of image patches. The disparity maps generated is validated with the OpenCV implementation.

Visual odometry uses the sequence of images from single or multiple cameras for estimating the relative motion of the camera with respect to its initial position. The aim of the procedure is to find the rotation matrix and the translation vector for every two consecutive frames. These are the two parameters which describe the motion of the vehicle.

Vehicle speed is an important traffic factor to study in autonomous driving systems. Objects are tracked in the video frames using SIFT algorithm. Disparity maps of stereo images are generated using MC CNN. Further, velocity of the tracked objects are calculated from disparity and FPS.

Chapter 2

Literature review

KITTI data-set, which is build for autonomous navigation system has been properly understood and analysed[1]. It consists of data corresponding to various road scenes with corresponding training and testing data. Files corresponding to labelling, calibration has been properly understood. This standard data-set will be used for classification and tracking.
Data corresponding to LIDAR has to be segmented in order to detect objects. Several algorithms were proposed in this domain.
In [2] a mixture of bag of words approach has been discussed where hierarchical segmentation has been done.In [3],[4] fast segmentation of 3D point clouds has been discussed where the problem of segmentation has been divided into subproblems as the local ground plane estimation and fast 2D connected components labelling. In local ground plane estimation, an efficient partitioning of data is done by representing as a circle and divided into segments and in fast 2D connected components the data will be mapped to a particular segment and bin accordingly.These fast connected components are then mapped to 2D occupancy grid fixed to ground.The problem of under segmentation is solved using 3D voxel grid segmentation.In [5], connected components algorithm has been used to segment LIDAR data and the centroids of clusters thus formed is calculated. Tracking of centroids of clusters is achieved by kalman filter and data is associated to objects accordingly by a threshold.In [6], the concept of surface matching is used to segment and classify objects where the object surface is compared with scene surface and thus an association is made between

them.
In [7], YOLO9000, an object detection algorithm was developed which uses a hierarchical view of object classification where it can detect over 9000 object classes.In [8], SSD has been implemented for object detection where a single deep neural network is used.In [9], fast R-CNN has been implemented which uses deep convolution neural networks for object detection.
In [10], the focus on short range sensing is discussed where the setup on car senses not only the front part but also around the vehicle for safe driving and to avoid collisions.
In [11], the features from LIDAR and image frames are classified and combined for posterior classification using a single classifier. Two fusion architectures, centralized and decentralized schemes were discussed. The pair of classifiers for best classification were taken based on maximum relevance and minimal redundancy(mRMR) criterion. In [12],[13],[14] different universities have build their own autonomous cars using data from both LIDAR and camera data and fusing them using different architectures for the DARPA urban challenge which involves with real-world object classification and tracking. In [15] a survey of the stereo vision matching algorithms that have been used in the past are discussed. Both dense disparity and sparse disparity algorithms are reviewed in this paper. In dense disparity matching algorithm, both local and global methods are compared using their speed, image size, disparity levels etc.
[16] uses a guided image filter (GIF) based method to produce the disparity map. Its corresponding hardware implementation is also described.
[17] uses Symmetric Dynamic Programming Stereo which constructs a Cyclopean image - one that would be seen by a virtual camera placed midway between the optical centres of the two real cameras rather than reconstructing the depths for the left and right images. This is implemented using FPGA and GPU and the results were compared in this paper.
In [18] a network architecture is proposed which first calculates the multi scale shared features. Using these shared features matching cost calculation, matching cost aggregation and disparity calculations are performed to estimate the initial disparity. The initial disparity and shared features are then used to calculate feature constancy which are then finally fed into a subnetwork to refine initial disparity.

In [19] two architectures are designed, fast and accurate where the left and right image patches can be given as input to the siamese structured network and get the disparity map after performing some post processing steps.

Chapter 3

Segmentation of LiDAR point cloud

Light Detection and Ranging, LiDAR is a method which uses light in the form of pulsed laser to scan the environment and to measure distance to the surrounding points. Due to it's high data density and fast data acquisition and processing LiDAR has become an essential sensor in Autonomous Vehicles. The output of the three dimensional scanning is given in the form of point clouds. This makes the calculation of velocity, acceleration of other vehicles and obstacles efficient and direct which is very essential in the case of a self driving vehicles where the scene is highly dynamic and quick decision making is essential. Point clouds can also be used for the purpose of object detection and tracking as well. LiDAR point cloud data which is available as a part of KITTI Object Tracking Evaluation dataset, was used for the testing of algorithms in this work. There are two different approaches that were implemented for segmentation of the point cloud. The difference between these two approaches is the way in which ground points are removed from the scene. One is based on Gaussian Process Regression which is a statistical approach and the other is based on the geometrical properties of the surface. The ground plane extraction is followed by connected components labelling on the non ground points for segmentation. For an organized and efficient implementation of both algorithms, the frame is divided into to 2D polar grid as follows:

3.1 Polar Grid mapping

Polar Grid mapping is done by constructing a circle of radius R = 30 meters in the xy plane from the position of LiDAR and then dividing it into segments. The set of points that belong to the frame and that are closer than 30 m, P_t are considered. $\Delta\alpha$ describes the angle covered by each segment as shown in Fig. 3.1.

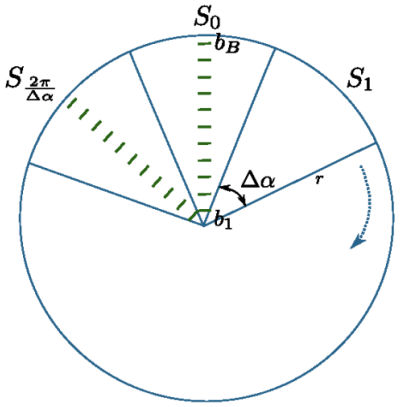


Fig. 3.1: Partitioning into segments

Hence the total number of segments in the grid is given by num_segments = 2π / $\Delta\alpha$. Thus the index of the segment to which a point (x_i,y_i,z_i) in P_t is mapped is given by,

$$S(p_i) = \frac{atan(x_i, y_i)}{\Delta\alpha} \tag{3.1}$$

The set of points which belong to a given segment is,

$$P_m = \{p_i \in P_t | S(p_i) = m\} \tag{3.2}$$

Now each of the segment is further divided into smaller bins b_n^m as given shown in Fig.3.2.

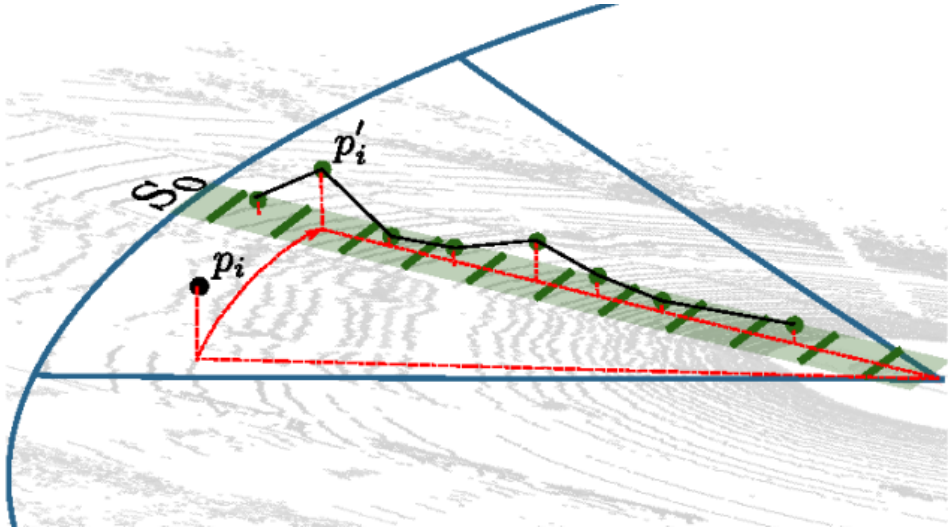


Fig. 3.2: Partitioning segments into bins

The n^{th} bin in segment m, b_n^m covers the range from r_n^{min} to r_n^{max} and a point p_i maps to b_n^m iff r_i satisfies the inequality $r_n^{min} < \text{r}_\text{i} < r_n^{max}$.

The 3D points that belongs to n^{th} bin in m^{th} segment forms a set $P_{b_n^m}$ and for every set $P_{b_n^m}$ there exists a one-one set $P'_{b_n^m}$ of 2D points $P'_{b_n^m} = \{p_i^{'} = (r_i, z_i)^T | p_i \in P_{b_n^m}\}$ where $\text{r}_\text{i} = \sqrt{{x_\text{i}}^2 + {y_\text{i}}^2}$. The advantage of this polar grid mapping is that complexity of the ground point removal algorithms can be tuned by varying the number of segments and bins considered. Hence there is a trade off between computational complexity with accuracy of ground extraction. Increasing the number of segments and bins increases the accuracy but at the same time increases the run time of the algorithm. Now the entire point cloud is divided into segments and bins based on its angular position in xy plane and distance from the sensor respectively. There will be bins which donot have any points in it and bins having multiple points in it. From each non-empty bin select the point which has the minimum z coordinate. Such a point is called the prototype point of that bin . The prototype point is selected to be point with minimum z coordinate since it has the maximum probability to belong to the ground from that bin. Construction of the set of prototype points drastically reduces the number of points that have to be considered for formulating the conditions for

ground points removal.

$$PG_m = \{p_i^{'} | p_i^{'} \in P_{b_n^m}^{'}, z_i = min(H_n^m), n = 1,2,...N\} \tag{3.3}$$

where H_n^m is the set of z coordinates mapped into bin b_n^m There exist cases where the z coordinate of the prototype point of a bin is very high that it cannot be considered as ground. So,

$$PG_m = \{p_i^{'} | p_i^{'} \in P_{b_n^m}^{'}, z_i = min(H_n^m) < z_{th}, n = 1,2,...N\} \tag{3.4}$$

where z_{th} is the threshold height and is a parameter of choice.
Now that the set of prototype points in a frame is found, the next step is to apply the algorithms for extracting and removing ground points.

3.2 Gaussian Process Regression

3.2.1 Central Limit theorem

When independent random variables which follow some unknown distribution functions(with finite mean and variance) are added, their properly normalized sum(mean), tends towards a normal distribution.
So, it is safe to assume that each point in the cloud forms a Gaussian distribution. Also, from multivariate central limit theorem, the sum of Gaussian distributions can be assumed to be a Gaussian Process.
So, Gaussian Process Regression(GPR) has been used for segmentation and removal of ground points. The LiDAR data in general does not fit any of the standard regression models like linear, cubic etc., Another advantage of Gaussian process is that it does not represent a function directly thus avoiding the problem of overfitting, but gives more details about the function, by getting trained by the data and updating parameters more efficiently. Still GPR is considered as a supervised learning algorithm. Other advantages of GPR is it is linearly smoother i.e the predicted output is linearly weighed combination of the past outputs and it is less parametric. But, the basic assumption about the functions needs to be assumed, otherwise more general techniques needs to be implemented by the principle of maximum entropy[20].

Gaussian distribution also follows marginalization property which means if the larger set follows a Gaussian, then the smaller set also follows Gaussian distribution i.e, if

$$(x_1, x_2) \sim N(\mu, \Sigma) then x_1 \sim N(\mu_1, \Sigma_{11}) and x_2 \sim N(\mu_2, \Sigma_{22}) \tag{3.5}$$

3.2.2 Mathematics

A Gaussian Process Regression is mainly characterized by its mean and covariance function. In general, the mean can be assumed as zero and it is the covariance function that mainly determines how well the algorithm is learning. It is similar to choosing the type of regression function(linear, cubic etc;) but in a more generic way as in case of Gaussian Process Regression, it varies the weights according to data. The basic regression equation with Gaussian noise is

$$f(x) = x^{\mathrm{T}} w \tag{3.6}$$

$$y = f(x) + \varepsilon \tag{3.7}$$

with x as input vector, w as weights of the model and f predicting the outputs. The predicted outputs(f) and actual outputs(y) differ by some noise which can be assumed as additive, independent, identically distributed Gaussian variable with zero mean and ${\sigma_{\mathrm{n}}}^2$

$$\varepsilon \sim \mathcal{N}(0, {\sigma_{\mathrm{n}}}^2) \tag{3.8}$$

In Bayesian, a prior estimation of weights needs to be assumed to get the posteriori probability. So we assume it to be Gaussian with

$$w \sim \mathcal{N}(0, \Sigma_{\mathrm{p}}) \tag{3.9}$$

So, the most common covariance function is squared exponential(SE) or Radial basis function(RBF) kernel as it is simplest one to implement and interpret[21]. GPs with this kernel have mean squared derivatives of all orders(as SE is infinitely differentiable) and thus they are very smooth . The covariance between a pair of random variables is

$$cov(f(x_{\mathrm{p}}), f(x_{\mathrm{q}})) = k(x_{\mathrm{p}}, x_{\mathrm{q}}) = {\sigma_{\mathrm{f}}}^2 \exp\left(-\frac{\|x - x'\|^2}{2l^2}\right) + {\sigma_{\mathrm{n}}}^2 \delta_{\mathrm{pq}} \tag{3.10}$$

where l is the length scale and ${\sigma_f}^2$ is the signal covariance and ${\sigma_n}^2$ is the noise variance which adds only when p = q as δ_{pq} is Kroneckar delta which is one iff p = q and zero otherwise.

If there are n_{train} = n training points and n_{test} test points then $K(X_{train}, X_{test})$ denotes n_{train} x n_{test} matrix containing covariances evaluated at all the training and test points[10]. Similarly it follows for $K(X_{train}, X_{train})$ and $K(X_{test}, X_{test})$

$$K(X_{train}, X_{train}) = K = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & ... & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & ... & k(x_2,x_n) \\ . & . & & \\ . & . & & \\ k(x_n,x_1) & k(x_n,x_2) & ... & k(x,x_n) \end{bmatrix} \tag{3.11}$$

$$K(X_{train}, X_{test}) = K_1 = \begin{bmatrix} k(x_{test},x_1) & k(x_{test},x_2)...k(x_{test},x_{train}) \end{bmatrix} \tag{3.12}$$

$$K(X_{test}, X_{test}) = K_2 = k(x_{test}, x_{test}) \tag{3.13}$$

As the data can be represented as a sample from multivariate Gaussian distribution, we have

$$\begin{bmatrix} y_{train} \\ \\ y_{test} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K & {K_1}^T \\ \\ K_1 & K_2 \end{bmatrix}\right) \tag{3.14}$$

The prediction for the new test point y_{test} given y_{train} can be formulated as

$$y_{test}|y_{train} \sim \mathcal{N}(K_1 K^{-1} y, K_2 - K_1 K^{-1} {K_1}^T) \tag{3.15}$$

with mean of this distribution(which gives the locality of the test point) as

$$\bar{y}_{test} = K_1 K^{-1} y \tag{3.16}$$

and the uncertainty in the prediction given in its variance as

$$var(y_{test}) = K_2 - K_1 K^{-1} {K_1}^T \tag{3.17}$$

Algorithm for Gaussian Process Regression[22]

Input: $P_{\mathrm{t}} = \{p_1, p_2, ...p_{\mathrm{t}}\}, M, N, B, z_{\mathrm{th}}, T_{\mathrm{r}}, \sigma_{\mathrm{f}}, \sigma_{\mathrm{n}}$
Output: label of $\mathrm{p_i}$,i = 1,...,l
1:(PG,$\mathrm{P}'_{b_n^m}$) = $PolarGridMap(P_t, M, N)$;
2:**for** i = 1 : M - 1 do
3: $\mathrm{s}_{new} = \emptyset$
4: $\mathrm{s}_{new} = seed(PG_i, R, z_{th})$;
5: $model$ = $regression$(GP,s_{new})
6: for j = 0 : N - 1 do
7: $segment$($model$,$\mathrm{P}'_{b_i^j}, T_r$);
8: end for
9:end for

The algorithm starts with a set of 3D cloud points $\mathrm{P_t}$ and it finally ends in labelling each and every point as ground or obstacle. It mainly consists of 4 steps: Polar grid mapping, seed evaluation(Estimation of initial ground), Gaussian Process Regression with non-stationary covariance function and pointwise segmentation(Estimation of final ground).

3.3 Pointwise Segmentation

For segment i, the final seeds has been obtained in the *seed* function and those have been trained in *regression* function. When a new value is given to this trained model, it gives mean and covariance values from which it can be classified as either ground point or obstacle.
For j^{th} bin in i^{th} segment, $(r_j^{min} + r_j^{max})/2$ is used as the radial coordinate of bin b_j^i. So, r_j^i is given as input to the model which returns mean height of the ground $\bar{z}_j^i$ in bin b_j^i. This mean height is taken as reference ground height H_{ij} and for every point $p_k^{'}$ in set $P_{b_j^i}^{'} = \{p_k^{'} = (r_k, z_k) | p_k \in P_{b_j^i}\}$ if its relative height $\|z_k - H_{ij}\| \leq T_r$ it is classified as ground. Otherwise it is classified as obstacle point.
By this way, all the 2D coordinates are classified as ground point or obstacle point

and this set is converted to 3D points as the mapping between $P_{b^i_j}$ and $P_{b^{i'}_j}$ is one-one. Now all the points in the cloud have been assigned a label, either it belongs to ground plane or it belongs to obstacle plane.
So, ground points have been removed by using Gaussian Process Regression and the labels thus obtained is passed to a function which performs Connected Component Labelling, which segments the obstacles.

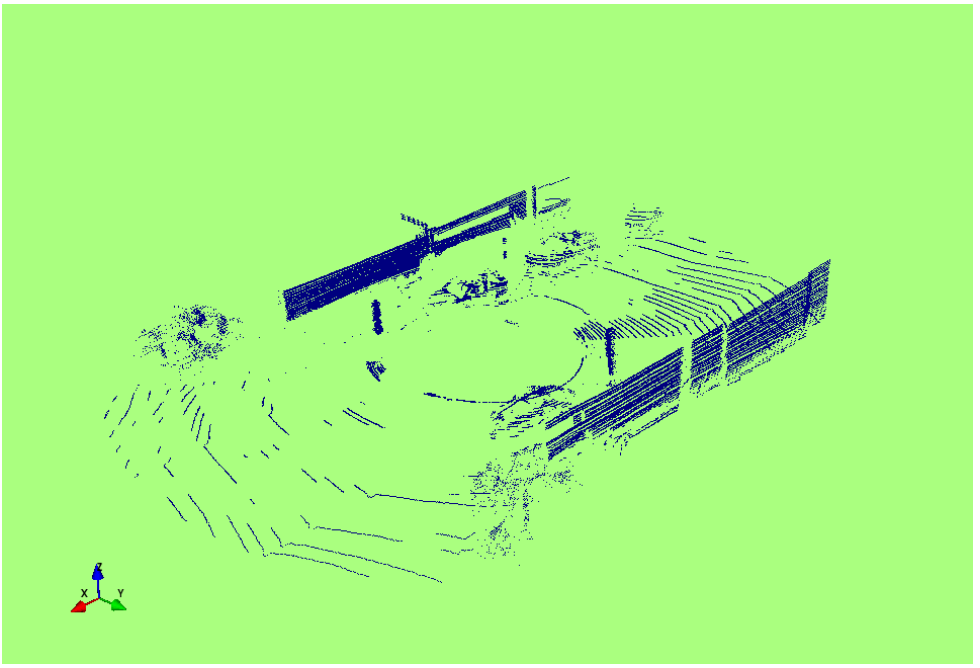


Fig. 3.3: Obstacle point obtained after Gaussian Process Regression

3.4 Fast Segmentation Algorithm

Efficient implementation of point cloud segmentation is essential for autonomous ground vehicle mobility as segmentation is considered to be a preliminary step of classification which itself is a computationally intensive operation. The efficiency stems from the fact that the computationally intensive three dimensional segmentation is broken down in to two computationally simpler sub problems : local ground plane estimation and then two dimensional connected components labelling.

Fast Segmentation gives the labels whether the point is ground or obstacle. In this algorithm, the initial step is Polar Grid Mapping and Seed extraction which is also used in Gaussian Process Regression.
After extracting the set of prototype points evaluation of ground points involves formulating simple conditions based on the slope and intercept of the lines formed by joining prototype points that belong to the same segment. Set of all the points which satisfy these conditions forms the reduced set of ground points in that segment. Conditions that were considered to form the reduced set of ground points were

1. The magnitude of slope of the line formed by joining prototype points of adjacent points should be less than a threshold value slope_thresh. This condition removes the sharp vertical structures from the ground plane.

2. If the slope of the line is less than a minimum slope value slope_min then the z intercept should be less than a threshold slope threshz_intercept. This removes the plateau kind of structure from the ground plane
   The actual point cloud from the sensor and the set of points that satisfy these conditions and were plotted

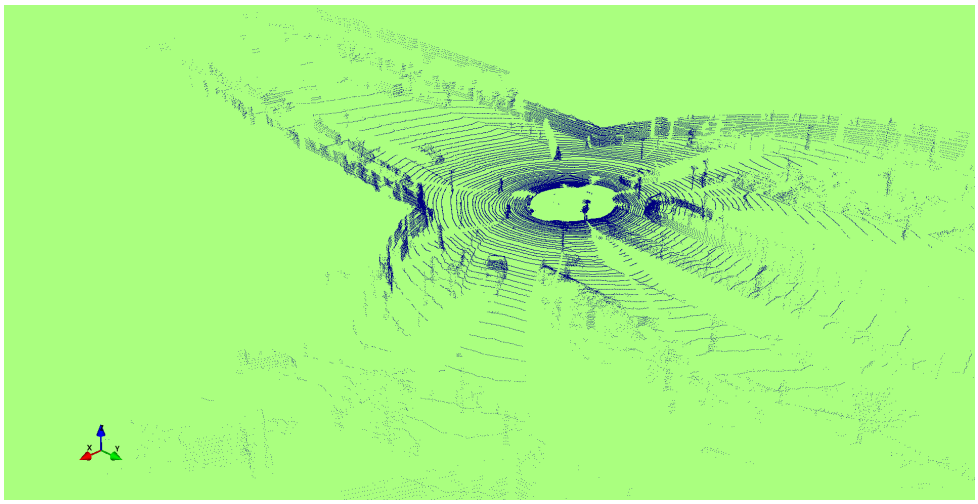


Fig. 3.4: Initial 3D point cloud

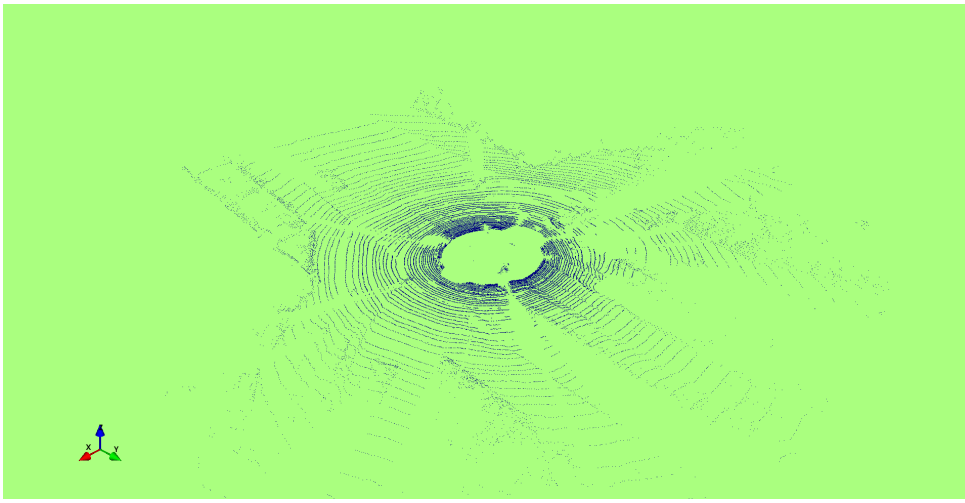


Fig. 3.5: Ground points obtained from Fast Segmentation

Inorder to find the actual set of ground points , the euclidean distance of rest of the points in the neighbouring bins of the bins having a ground point in the reduced ground point set is calculated. If this distance is found to be less than a threshold distance, thresh_dist than those points are also considered to belong to the set of ground points. This set of ground points is removed from the actual point cloud to find the set of non ground point.

Algorithm for Fast Segmentation

1.gndpts = ∅
2.for i = 1 to num_bins do
3. if |bin| >0 then
4. pt_begin = prototype_pt[bin]
5. break
6.for i = 1 to num_bins do
7. if |bin| >0 then
8. slope = findslope(pt_begin,prototype_pt[bin])
9. intercept = findintercept(ptbegin,prototype_pt[bin])
10. if ((|slope| <slope_thresh) ∧ (|bin| >slope_min)) ∨ ((|slope| <slope_min) ∧ (intercept <threshz_intercept)
11. gndpts = gndpts ∪ prototype_pt
12. ptbegin = prototype_pt

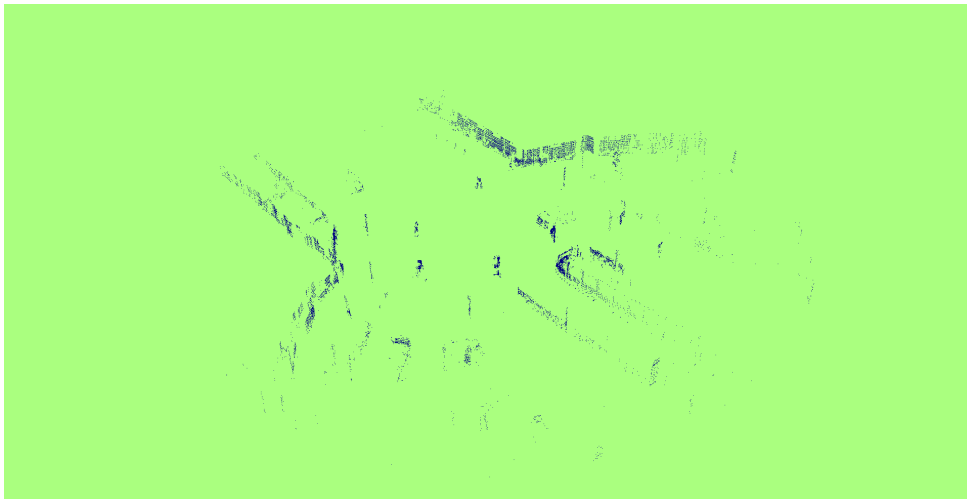


Fig. 3.6: Non ground obtained from Fast Segmentation

An important advantage of this method is that since it is the local geometrical features of the point cloud considered for finding the ground plane rather than a

global threshold height ground plane extraction from complex scenarios like a slant road is possible with this method. Fig 3.7 depicts scenario where the sides of the road are slanted.

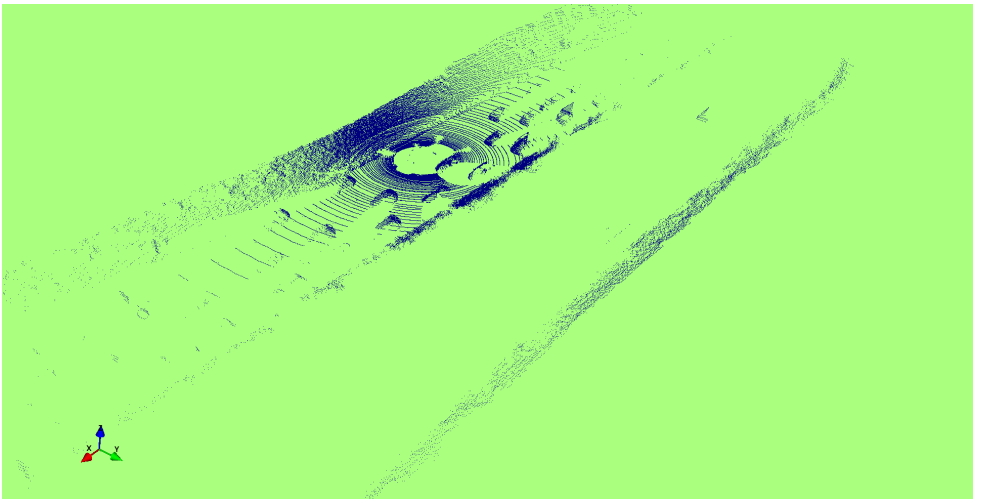


Fig. 3.7: A scenario which depicts the slope road and obstacles

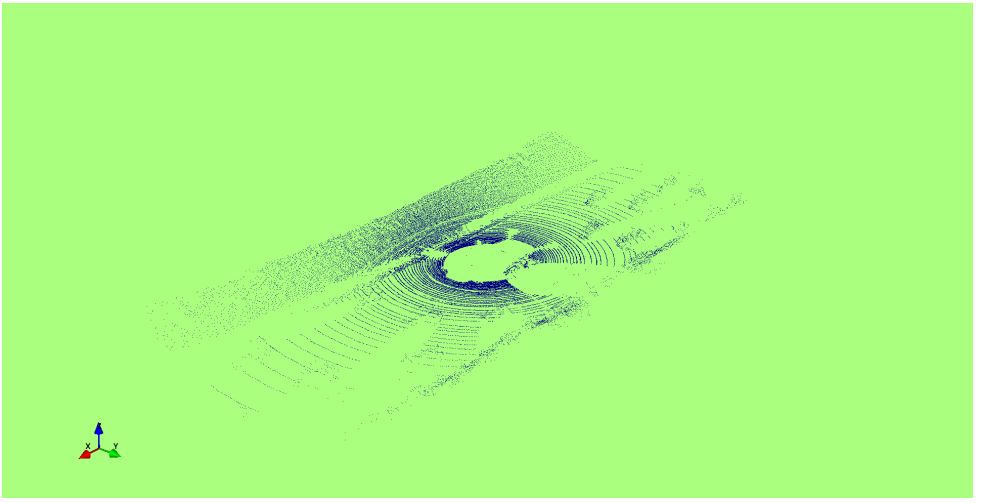


Fig. 3.8: Ground point detection from the slope scenario(Fast Segmentation)

Fig 3.8. and Fig 3.9. depicts the extracted ground points from a scenario which has a slope in it. It is obvious from these images that fast segmentation which in fact exploits the local geometrical features of the scene performs better in scenarios where the ground points are no uniformly distributed.

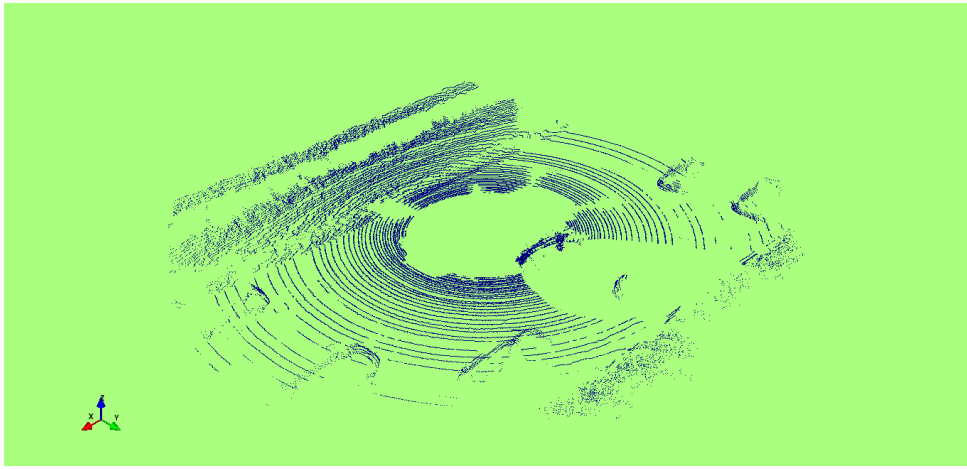


Fig. 3.9: Ground point detection from the slope scenario(Gaussian Process Regression)

3.5 Segmentation of Non Ground Points

Now all the points in the point cloud is labelled as either ground points or non ground points. The concept of connected components labelling on a binary image from computer vision is used for segmentation of the non ground ground points. A binary representation of the grid is first obtained by labelling the bins which contains points in as binary one and those bins which which contain any points as binary zero. Now a binary matrix is obtained on which connected components labelling can be applied. The two pass algorithm for connected components labelling is applied here.
The algorithm in its first pass label all the foreground points with a new label if none of it's neighbours are labelled else label the bin with the lowest label value of its neighbour. Also it make the neighbour with the lowest label of a bin as the root label. In the second pass all the labels will be replaced with the value of its root label.

3.5.1 Connected Component Labelling using Two Pass Algorithm

Pass 1

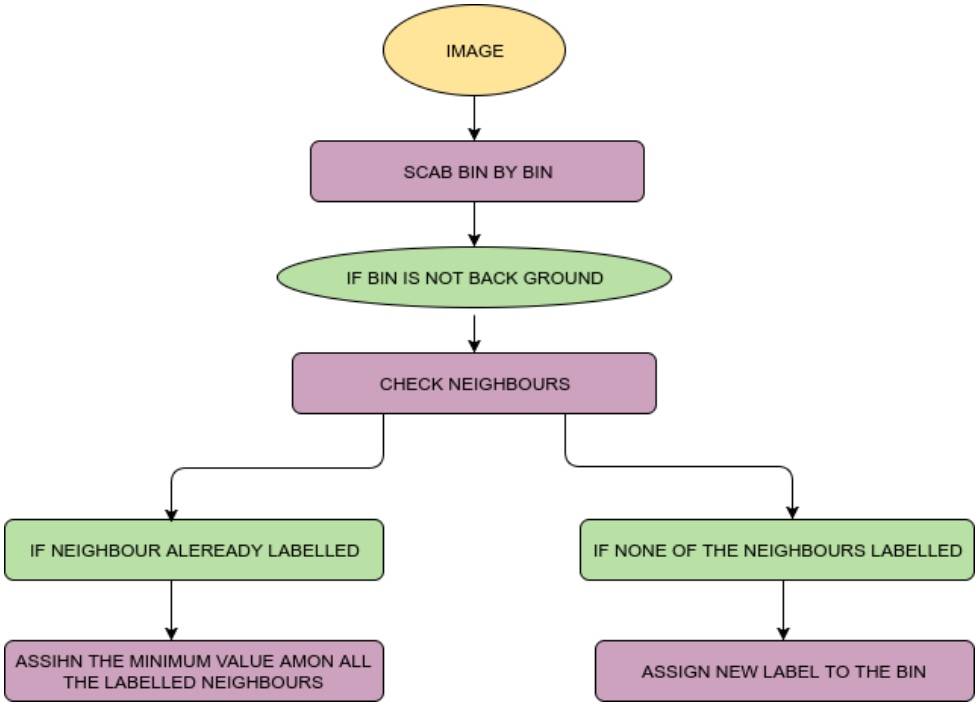


Fig. 3.10: Flowchart of connected component labelling - Pass 1

Pass 2

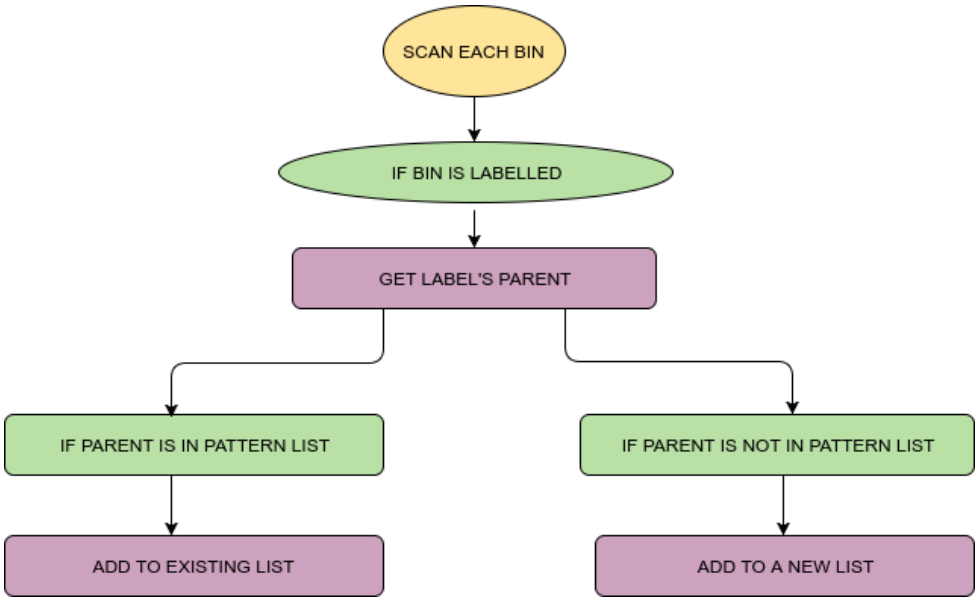


Fig. 3.11: Flowchart of connected component labelling - Pass 2

The given above connected component labelling was applied to the binary grid generated by the two ground point removal algorithms and the segmentation result was observed as shown in Fig. 3.12 - Fig. 3.17.

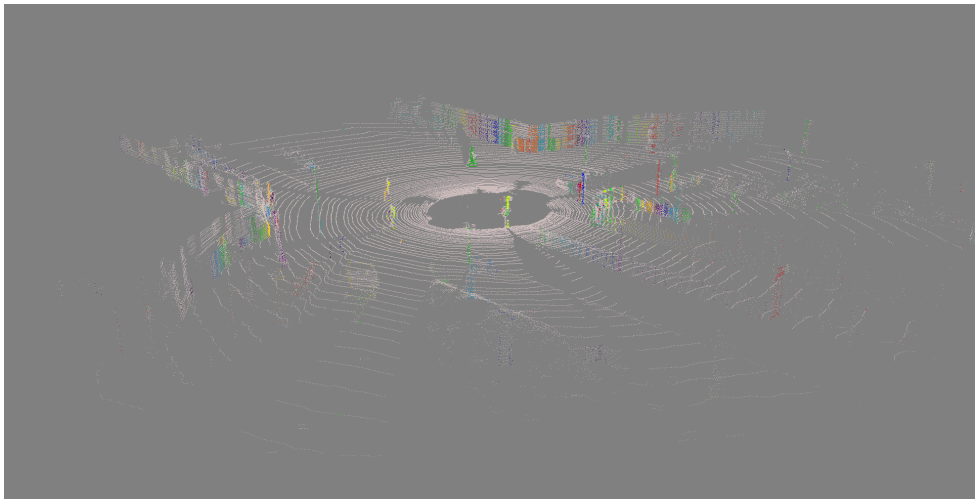


Fig. 3.12: CCL on Fast Segmented LiDAR data

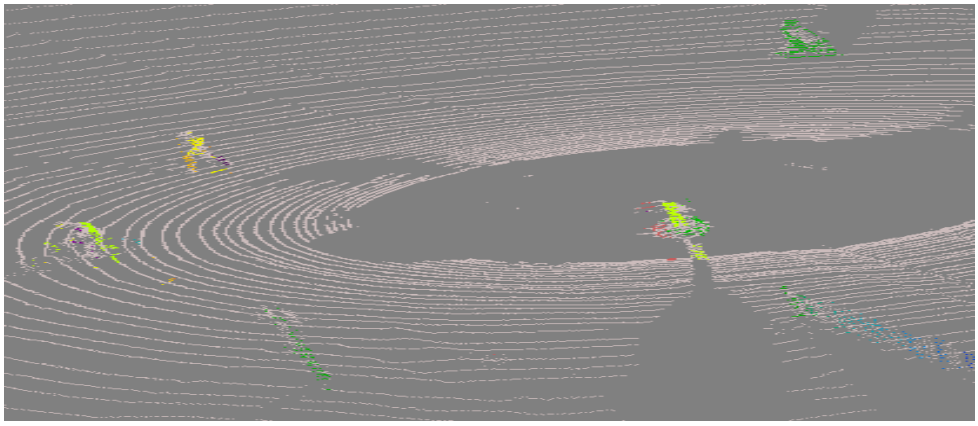


Fig. 3.13: CCL on Fast Segmented LiDAR data(close view)

Usually in connected components labelling of neighbouring images 8 - nearest neighbours are considered for labelling. In this application since the point cloud became sparse after the ground point removal 24-nearest neighbours are searched for establishing neighbourhood connections.

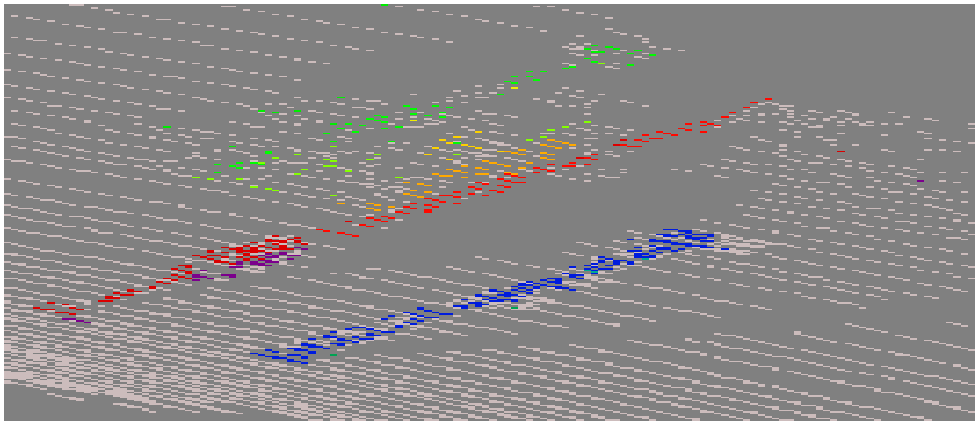


Fig. 3.14: CCL on Fast Segmented LiDAR data(closer view of poles)

Even though it is the process of ground point removal that caused the loss of some of the required points, the step is unavoidable as removal of ground points reduced the complexity of segmentation very much. Also applying a two dimensional labelling algorithm to a three dimensional scenario without removing ground points can cause under segmentation.

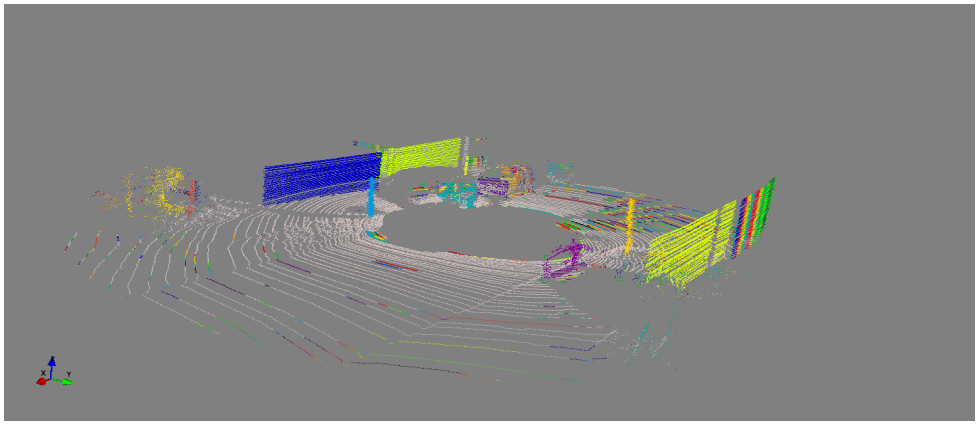


Fig. 3.15: CCL on GPR LiDAR data

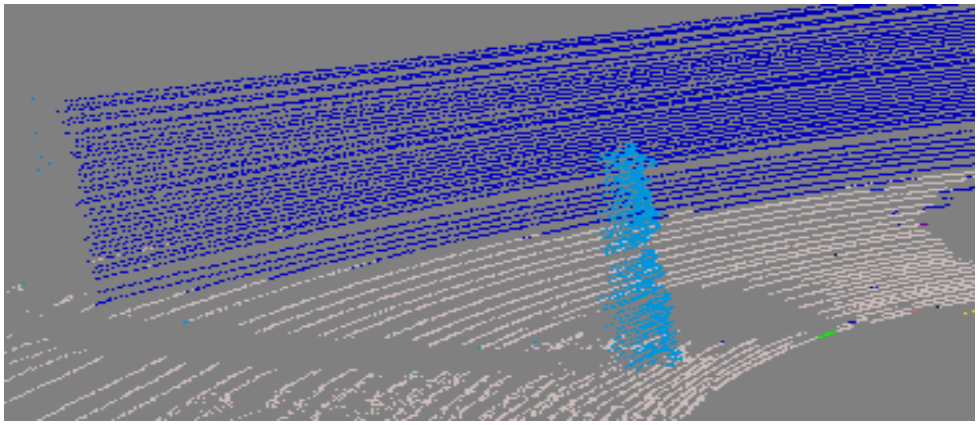


Fig. 3.16: CCL on GPR LiDAR data(Close view)

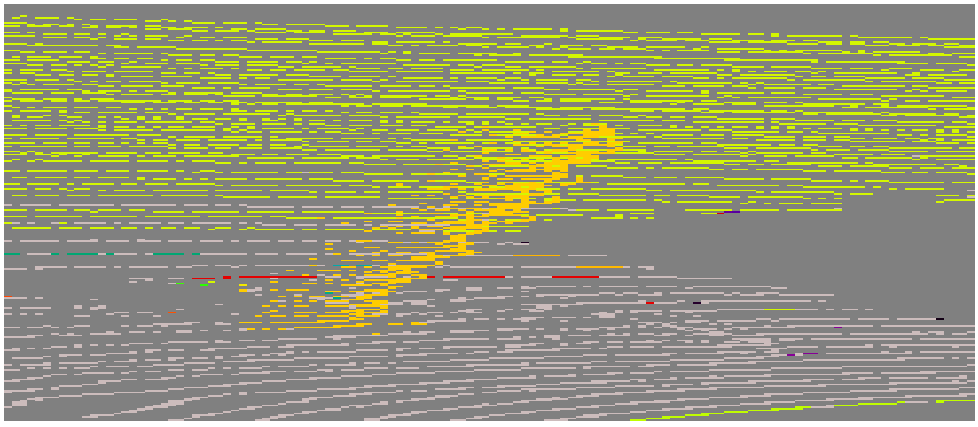


Fig. 3.17: CCL on GPR LiDAR data(person)

Chapter 4

Data Set for Object Detection and Stereo vision for Indian Roads

4.1 Object Detection Dataset

For training the system to classify the different objects that could be encountered on an Indian road, a data-set including a variety of classes needs to be created. By using Noise Play 2 Action camera, traffic in and around Kottayam district, Kerala and was recorded in 720p at 60fps. A set of images (at a rate of 5 images per second) were generated from this recorded video footage.These images were annotated using an online tool - 'Label box' and a text file was created for each scene which has information about the location of various objects in that particular frame. These files can then be used for the visualization of the data-set and to train a system to detect the obstacles or objects on road. There are seven classes labelled namely

- Car
- Pedestrain
- Auto Rickshaw
- Truck
- Two Wheeler

- Bus
- Van

4800 images were taken under different traffic conditions and are labelled. The frequency of each class is listed in Table 4.1

Class	Frequency
Car	8127
Pedestrain	1930
Auto Rickshaw	4724
Truck	576
Two Wheeler	5721
Bus	892
Van	186

Table 4.1: Frequency of objects of each class

The collected data is labelled according to the classes given above and the dataset was prepared according to PASCAL VOC format for training different architectures. An example image of the created dataset can be seen in Fig 4.1

4.2 Stereo Dataset

To work on stereo algorithms, data is collected in and around Kottayam district using Intel RealSense Depth camera D435. This depth camera have 2 infrared cameras having global shutter that are triggered simultaneously so that calculation of disparity for a particular scene is possible. It also has a color image sensor that can be used for object detection. The range is over 10m which makes this suitable for outdoor applications. This stereo camera has an inbuilt vision processor that enables to obtain real time depth information which can be used for validation of the results obtained after performing stereo algorithms. An example image of the created stereo dataset can be seen in Fig 4.2

Fig. 4.1: An example image from object detection dataset

(a) Left image

(b) Right image

Fig. 4.2: An example image pair taken from Intel Realsense camera

Chapter 5

Real-time object detection

Real-time object detection is the primary task an autonomous car needs to perform for its safe journey. Accuracy and speed are the tradeoffs for this real-time application. A desirable system should be very fast in detecting objects as it is a real-time process while maintaining a decent accuracy in its detection. Some state-of-the-art algorithms such as R-CNN, Fast R-CNN, Faster R-CNN, SSD and YOLO can be considered as the candidates for this task.

5.1 R-CNN and its variants

Faster R-CNN as discussed in [9] comprises of a deep convolutional neural network and a detector network. Region proposal network(RPN) takes an image of any size and outputs a set of rectangular regions with scores of probabilities. RPN is implemented by the deep convolutional neural network and the regions generated by this is used in the next stage to detect objects.
The basic R-CNN as discussed in [23] has a region proposal layer which proposes around 2000 object boundaries. Region proposals are estimated by a region proposal algorithm which uses selective search as described in [24]. After selecting the regions, these are resized and are given to CNN to extract features. At the end, a classifier is present which predicts the classes and its corresponding probabilities.
The major improvement from R-CNN to Faster R-CNN is that in the former there

will be around 2k convolutional networks which process on the 2k proposed regions but in the later there will be only one convolutional network which outputs these 2k region proposals. So, the computation speed is improved drastically.

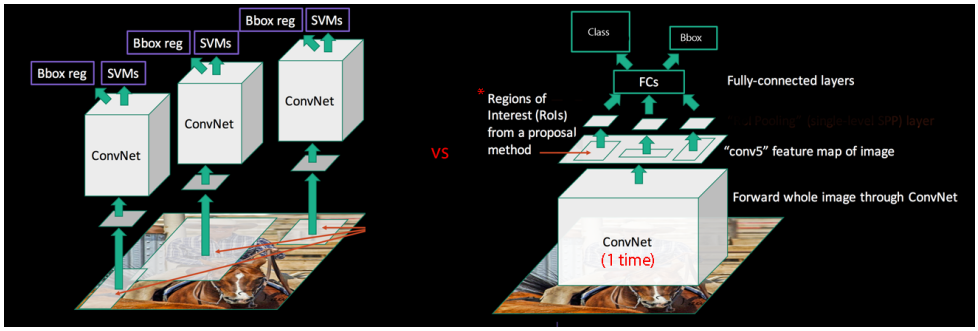


Fig. 5.1: Main difference in architecture of R-CNN and Faster R-CNN

1200 images are trained for 70 epochs, each epoch having 200 iterations in 4GB GPU system. Resnet is used as the base architecture to train and extract features.

Classifier Accuracy	0.894
Loss RPN Classifier	0.057
Loss RPN Regression	0.0846
Loss Detector Classifier	0.26
Loss Detector Regression	0.11

Table 5.1: Different metrics obtained after training Faster R-CNN

Fig. 5.2: Output of Faster R-CNN

The main drawback of Faster R-CNN is it takes nearly 2sec to detect classes present in each image which is not suitable for real-time object detection. So, work on YOLO was carried to obtain real-time detections.

Most of the above mentioned approaches use a region proposal method to generate potential bounding boxes and run a classifier on these proposed boxes to identify the object, but these methods are hard to optimise. YOLO observes the object detection task as a single regression problem where a single convolutional network simultaneously predicts multiple bounding boxes and associated class probabilities for those boxes. This allows YOLO to be extremely fast in predicting what all objects are present and where they are in the provided image frame. YOLO also have the feature of viewing the entire image during training and testing which proves to be advantageous when compared to region based or sliding window techniques, enabling YOLO to encode contextual information about the classes. This feature improves YOLO's capability of avoiding background detections which is common in other popular algorithms.

5.2 Approach used by YOLO

The images given to the YOLO network is divided into an $S \times S$ grid .If the centre of the concerned object is present in any of the available grids, then that grid is responsible for detecting that particular object. Hence each grid cell predicts bounding boxes and an associated confidence score which implies how likely an object is present and how accurate the box is. Each grid also predicts conditional class probabilities one per class. So bounding box prediction contains 5 elements: (x_{mid}, y_{mid}, w, h) and a box confident score where x_{mid}, y_{mid} are the normalised offset of a bounding box and w, h are the normalised width and height of the box. Since all the values are normalised they lie in the range of [0,1].

The figure shown below represents the YOLO architecture

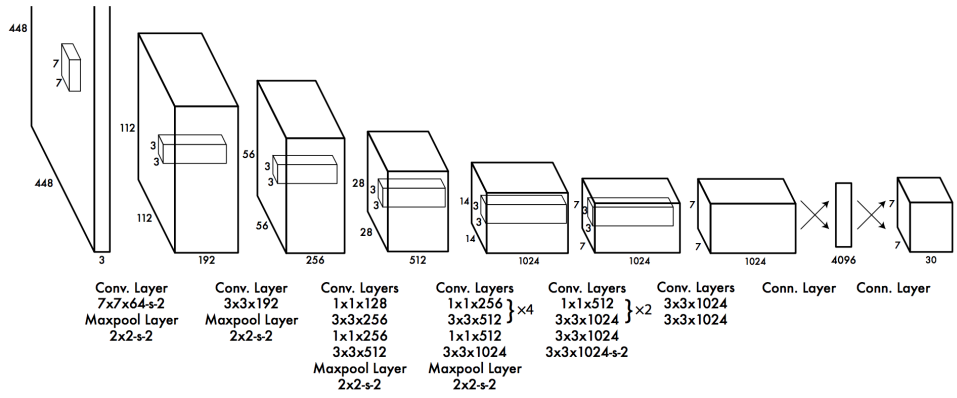


Fig. 5.3: Architecture Of YOLO

YOLO makes multiple predictions of the same object because of the grid cell division. So, it uses Non-max suppression(NMS) to finally predict one box for a particular object. For that the bounding box predictions are sorted based on their confidence score and neglect the ones with lower confidence score. Each box predicted will also have a corresponding IoU(Intersection over Union) value which is defined as ratio of area common to bounding box predicted by the algorithm and the ground truth bounding box to the common area bounded by both the predicted bounding box and the ground truth bounding box. A particular threshould like IoU$>$0.5 is kept so that predictions of the same classes below that threshould is also neglected. Hence

issue of multiple detections of the same object can be reduced by increasing the IoU threshold.
YOLO makes wrong predictions if there are multiple objects which are close to each other in an image due to the grid cells and bounding box constraints posed during its prediction. It also struggles with small objects due to the same reason.
So, YOLO v2 was proposed to overcome these errors and which performs faster than YOLO[13]. Its main difference lies in the usage of :

Batch Normalisation: Making any changes in the input distribution has less effect on the hidden layers since this technique ensures the initial layers to be bounded by a specific mean and variance when viewed from the point of view of the hidden layers. This facilitates the hidden layers to be more stable towards co-variance shifts done at input during training and testing data.

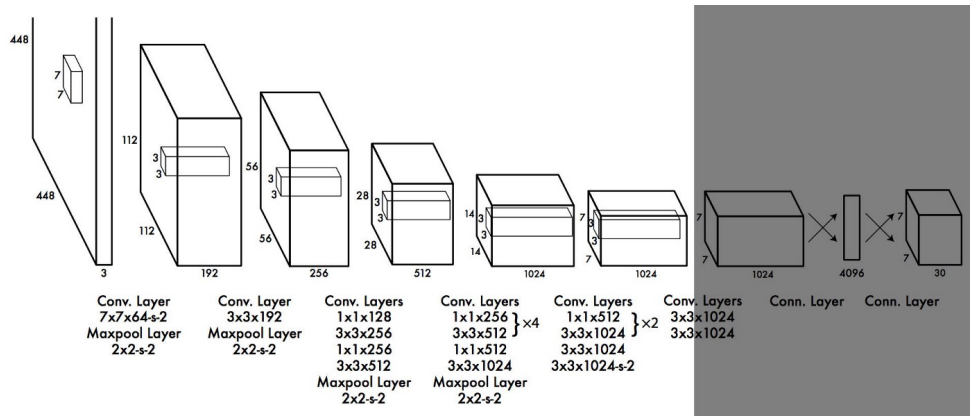


Fig. 5.4: Architecture Of YOLO v2

Compared to YOLO, in the YOLO v2 architecture the fully connected layers for predicting the bounding boxes are removed which can result in improved performance.

Convolution with anchor boxes: Instead of making an arbitrary guess for bounding boxes, YOLO v2 starts with an arbitrary guess of 5 bounding boxes according to

aspect ratios of the most common classes i.e cars, persons etc. The best anchor boxes are selected by applying k-means clustering on boxes by using IoU scores as it is independent of the size of the box. In YOLOv2 filters = 5*(classes + 5) at the last layer. So, as seven classes are used for detection, the number of filters that were used are 60. All the other layers specification filters are not changed. Darknet, an open source neural network framework was used to implement YOLO v2. We trained YOLO v2 for about 1200 images and were trained for about 4000 epochs in a 2GB RAM NVIDIA GeForce 940M for about 50 hours with batch = 64 and subdivisions = 64 and obtainedloss of 0.64

5.2.1 Outputs obtained from YOLO v2

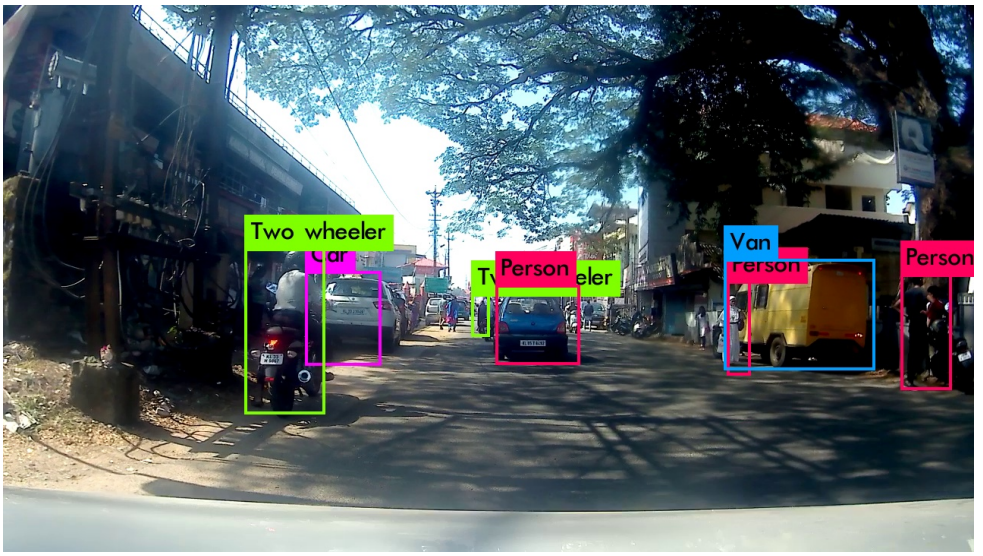


Fig. 5.5: Wrong prediction by YOLO v2. A car is identified as a person

Further training was made for additional 10 hours and accuracy had been improved.

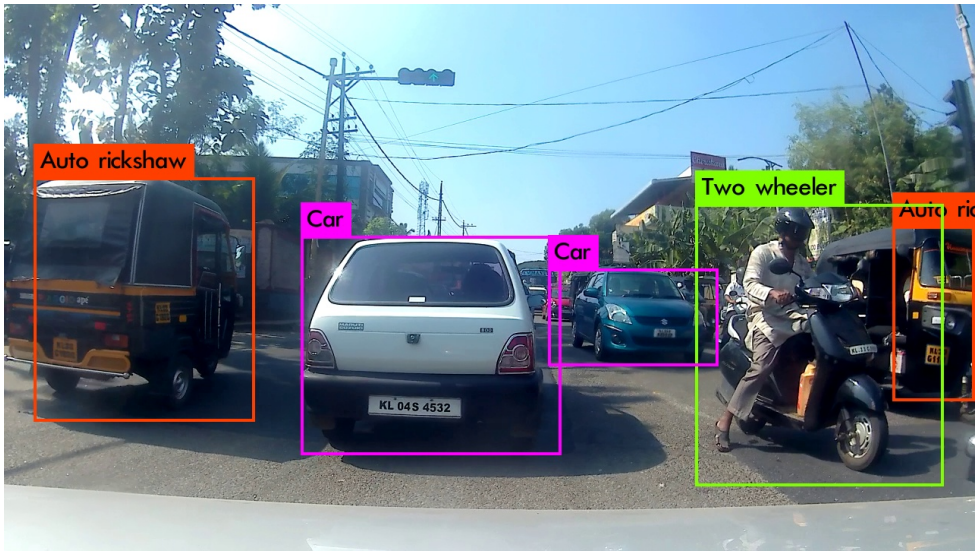


Fig. 5.6: Correct prediction by YOLOv2

YOLOv3 has some added advantage over v2 and some of them are

- More accurate than YOLOv2 with a small reduction in detection rate because of increase in number of fully convolutional layers from 30 to 106
- Has residual blocks, skip connections and upsampling
- While YOLOv2 fails to detect small objects, YOLOv3 detects objects at 3 different scales.

Fig. 5.7: Output of YOLOv3

Chapter 6

Depth estimation using stereo vision

Stereo matching is the process by which a three dimensional model of a scene is estimated from two or more images of the same scene by finding the matching pixels in these images.

6.1 Disparity Map

A normal camera lacks the information about depth and it becomes very essential to have this information for an autonomous car so that the decisions taken by the car will be more judicious. To acquire data corresponding to depth, a minimum of 2 cameras are used known as stereo camera. Any object placed in the plane of the camera will get captured by these 2 cameras but with slight horizontal displacement. This horizontal displacement is known as “disparity” of an object which is created due to parallax. The disparity value of a pixel is equal to the shift value that leads to the minimum sum-of-squared-differences for that pixel.

Fig. 6.1: Disparity output from kitti dataset

Consider the image given below :

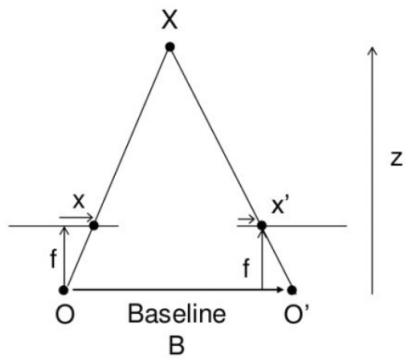


Fig. 6.2: Geometry for finding disparity

In the figure given above x and $x^{'}$ are the distance between points in image plane corresponding to the scene point X and their camera centres O and $O^{'}$.B is the inter

camera distance also known as the baseline distance and f is the focal length of camera (focus is known prior to the disparity estimation). Disparity of the point X at a distance Z from the center of the baseline is given by,

$$disparity = x - x^{'} = \frac{Bf}{Z} \tag{6.1}$$

Given the 2 images from the cameras, the disparity d for every pixel can be calculated. In other form, it is equivalent to mapping the given pixel in left image to its corresponding right image i.e (x,y) in left image is same as (x-d,y) in right image where d will be the disparity of that particular pixel. Given disparity d, the depth can be calculated as

$$z = \frac{fB}{d} \tag{6.2}$$

So, the stereo vision problem can be classified into 4 main steps according to [25]

1. Matching cost computation
2. Cost aggregation
3. Optimization
4. Disparity refinement

In [26] the first two steps are referred to as computing the matching cost and the last two steps as stereo methods. The technique to give the pair of image patches to a siamese network as discussed in [19] which is Matching Cost CNN(MC-CNN) was implemented and discussed in the next sections.

6.2 General matching cost computation

Initially, the sum of absolute differences is calculated for a given position 'p' and for all disparities 'd' around it.

$$C_{SAD}(\mathbf{p}, d) = \sum_{q \varepsilon \mathcal{N}_p} |I^L(\mathbf{q}) - I^R(\mathbf{q\text{-}d})| \tag{6.3}$$

Where $I^L(q)$ corresponds to pixel values in a rectangular image patch $\mathcal{N}_p$ centered at position 'p' in the left image and $I^R(q)$ corresponds to pixel values in a rectangular

image patch.$\mathcal{N}_{p-d}$ centered at position 'p-d' in the right image.If the image patch centered at 'p' in the left image is similar to the image patch centered at 'p-d' in right image, then C_{SAD} will be low, indicating that the two patches are centered around the same 3D point in the given scene and vice versa. To learn the similarity measure on the image patches, two neural network architectures, namely, fast and accurate were proposed in [19] from which fast architecture was implemented because of the time constrains involved in making decisions by an autonomous car.

6.3 MC-CNN by fast architecture

6.3.1 Matching cost computation in fast architecture

Matching cost between two image patches is evaluated by using a Siamese network i.e 2 shared-weight sub-networks joined at the head. Each subnetwork is an individual convolutional neural network with having rectified linear units at the end of each layer except the last layer. Both subnetworks output vectors extracting the properties of the input patches. These two vectors are compared using cosine similarity measure to get final network output. The above network is trained to minimize hinge loss i.e. the network will give more preference to a case where the image patch chosen in the right image is shifted to left position. The matching cost is evaluated as:

$$C_{CNN}(\mathbf{p},d) = -s(< \mathscr{P}^L(\mathbf{p}), \mathscr{P}^R(\mathbf{p}\text{ - }\mathbf{d}) >) \tag{6.4}$$

where $s(< \mathscr{P}^L(\mathbf{p}), \mathscr{P}^R(\mathbf{p}\text{ - }\mathbf{d}) >)$ is the output of the network when run on input patches $\mathscr{P}^L(\mathbf{p})$ and $\mathscr{P}^R(\mathbf{p}\text{ - }\mathbf{d})$ and the minus sign converts the similarity score produced by the network to a matching score

6.3.2 Stereo method

To compute more accurate disparity maps, some post-processing steps needs to be implemented after obtaining the matching cost with convolutional networks.Post-processing steps like semiglobal matching, subpixel enhancement, left-right consistency check constitutes the stereo method.

6.3.3 SemiGlobal matching:

The obtained matching cost is smoothened by applying an energy function (E(D)) to the disparity map as

$$E(D)=\sum_{p}(C^{4}_{CBCA}(\mathbf{p},D(\mathbf{p}))+\sum_{q\varepsilon\mathcal{N}_p}P_1.1\{|D(\mathbf{p})-D(\mathbf{q})|=1\}+\sum_{q\varepsilon\mathcal{N}_p}P_2.1\{|D(\mathbf{p})-D(\mathbf{q})|>1\} \tag{6.5}$$

In semiglobal matching it is required to reduce the above energy function in a direction r by defining a matching cost $C_r(\mathbf{p},d)$ with the relation

$$C_r(\mathbf{p},d)=C^{4}_{CBCA}(\mathbf{p},d)-\min_k C_r(\mathbf{p}\text{ - }\mathbf{r},k)+\min\Bigg\{C_r(\mathbf{p}\text{ - }\mathbf{r},d),C_r(\mathbf{p}\text{ - }\mathbf{r},d-1)$$
$$+P_1,C_r(\mathbf{p}\text{ - }\mathbf{r},d+1)+P_1,\min_k C_r(\mathbf{p}\text{ - }\mathbf{r},k)+P_2\Bigg\}$$

The second term is subtracted to prevent values of $C_r(\mathbf{p},d)$ from growing too large thereby affecting the optimal disparity map. The final cost $C_{SGM}(\mathbf{p},d)$ is computed by taking the average across all four directions:

$$C_{SGM}(\mathbf{p},d)=\frac{1}{4}\sum_r C_r(\mathbf{p},d) \tag{6.6}$$

6.3.4 Computing the Disparity image

The disparity image $D(p)$ is computed by the winner-takes-all strategy, that is, by finding the disparity d that minimizes $C(\mathbf{p},d)$

$$D(\mathbf{p})=\min_d C(\mathbf{p},d) \tag{6.7}$$

Interpolation

The disparity map predicted by left and right image may have differences and some of them can be rectified by performing left-right consistency check. Let $D^L(\mathbf{p})$ corresponds to disparity map generated by taking left image as reference and $D^R(\mathbf{p})$

be the disparity map generated by taking right image as reference.The final disparity for position **p** is decided based on following conditions:

$$\begin{gathered}\text{correct if } |d - D^R(\mathbf{p} - d)| \leq 1 \text{ for d} = D^L(\mathbf{p}),\\ \text{mismatch if } |d - D^R(\mathbf{p} - d)| \leq 1 \text{ for any other } d,\\ \text{occlusion otherwise}\end{gathered}$$

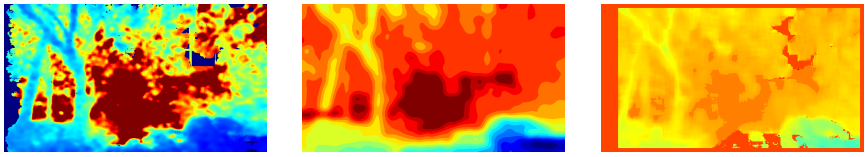
In case the label at a position is occlusion, a new disparity value is obtained by taking the value at left position that is labelled as correct. For positions marked as mismatch, the median of the surrounding disparity values is taken as the disparity at that position.

6.4 Results

So, these images are placed in respective folders, their paths are specified so that they will be given as inputs to the subnetworks. The outputs are obtained with alpha(a color visualising parameter)= 15 and are as follows

(a) Left image (b) Right image

Fig. 6.3: An example image pair taken from Intel Realsense camera



(a) Intel camera (b) MC-CNN (c) OpenCV

Fig. 6.4: Outputs of Disparity maps obtained by camera, MC-CNN and OpenCV

6.5 Validation with 2D LiDAR

To validate the depth obtained from stereo camera, RP LiDAR can be used. Data corresponding to the room as seen in Fig 6.6 is visualised from RP LiDAR and is plotted in Fig. 6.7 and 6.8

Fig. 6.5: Isometric view of the room

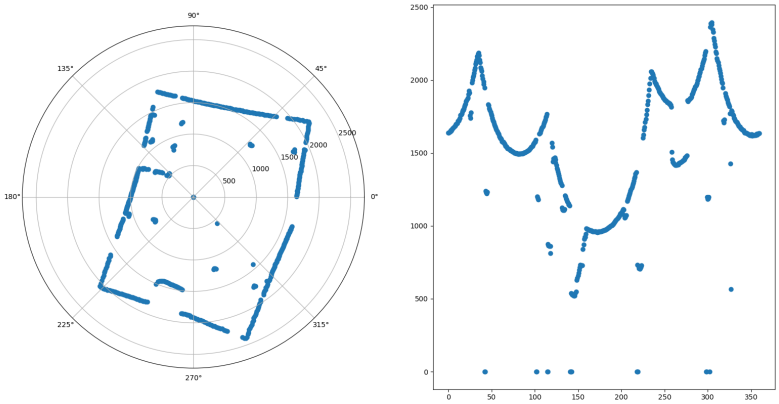


Fig. 6.6: Left image shows the r-θ plot and right image shows the distance of obstacle vs θ plot with obstacle placed around 250°

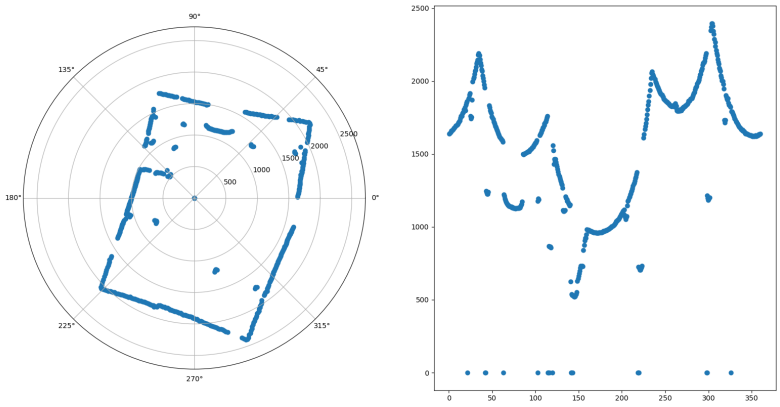


Fig. 6.7: Left image shows the r-θ plot and right image shows the distance of obstacle vs θ plot with obstacle placed around 70°

From the above pictures, it can be concluded that the LiDAR has a static internal axis. The suitcase is placed at around 250° and 70° and the bulges can be observed in Fig 6.9 and Fig 6.10.

Chapter 7

Odometry

Localization and navigation are among two of the most important problems in the field of mobile robotics. Localization problem involves estimating the position of the mobile agent with respect to its environment. Odometry refers to a class of techniques which estimates the change in position of a system over time using various sensors. This continuous localization of its position by the robot is necessary for its autonomous navigation and hence forms one of the corners of a self driving vehicle. Visual odometry uses the sequence of images from single or multiple cameras for estimating the relative motion of the camera with respect to its initial position. When the camera is rigidly mounted to the robot the motion of the camera and the robot are the same. Depending on the setup of the camera it can be classified into monocular (uses a single camera) and stereo (uses two cameras) visual odometry. Even though stereo odometry techniques achieve higher accuracy compared to its monocular counterpart, the monocular techniques are also widely used. This is mostly attributed to the lesser hardware demand the monocular implementation has as well as easiness in the calibration procedure. Also a monocular set up is much cheaper compared to stereo. An important aspect of the monocular technique is that it can be fused with other sensors for improving its accuracy as well as for finding the scaling factor for determining the exact three dimensional position.

7.1 Problem Formulation of Monocular Visual Odometry

The input given to the visual odometer is the stream of images coming from one of the cameras. Two consecutive frames are considered at an instant for the rotation and translation estimation. Also the prior knowledge of the intrinsic parameters of the camera is necessary for the procedure.
The aim of the procedure is to find the rotation matrix and the translation vector for every two consecutive frames. These are the two parameters which describe the motion of the vehicle. However the technique being monocular we can estimate the translation only upto a scaling factor which has to found by other means.

7.2 Algorithm

The core idea of the algorithm is the capture two consecutive frames of images, undistorting those and then tracking the features detected in the initial frame to the next frame. Then estimating the essential matrix from the tracked features and finally computing the rotation matrix and translation vector from the essential matrix.
The feature detectors used for this implementation is the FAST corner detectors [27] Once the features in two consecutive frames are detected a Kanade–Lucas–Tomasi feature tracker is used for tracking the features from one frame to the next. If the number of trackable features at any iteration is found to be lesser than a threshold a re-detection of features at that frame is triggered. Otherwise the essential matrix is computed. Essential matrix relates the corresponding points in the two images and can be decomposed to find the rotation matrix and translation matrix. The algorithm is depicted in Fig 7.1.

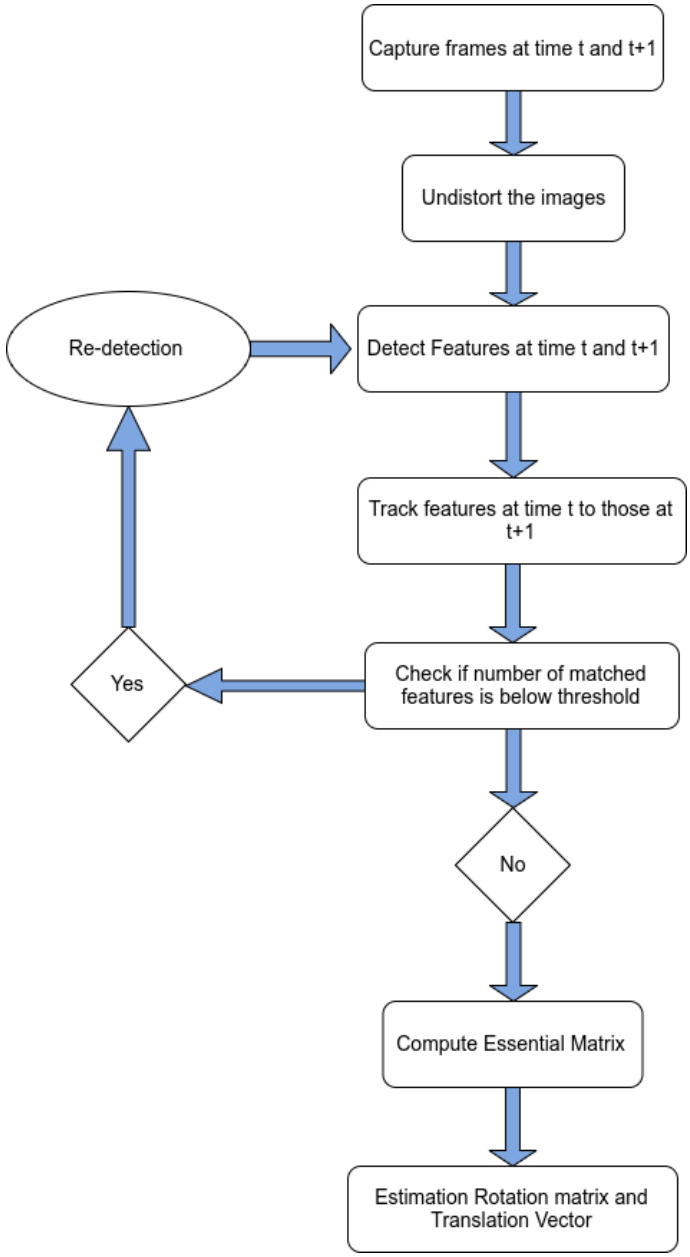


Fig. 7.1: Block Diagram of Monocular Visual Odometry Algorithm

7.3 Results from KITTI Dataset

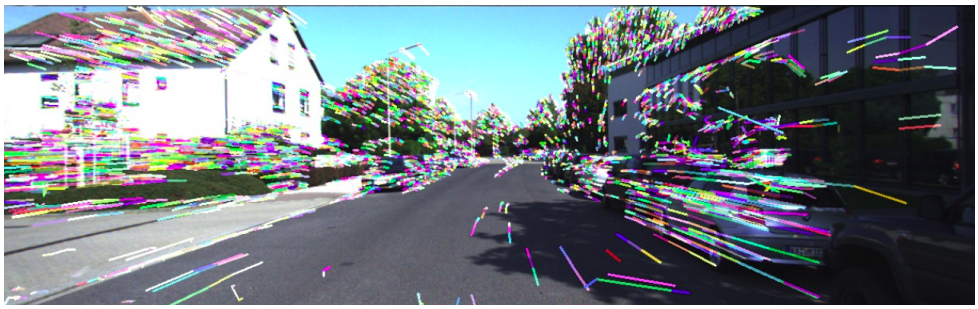


Fig. 7.2: Feature Tracking using KLT Tracker in KITTI

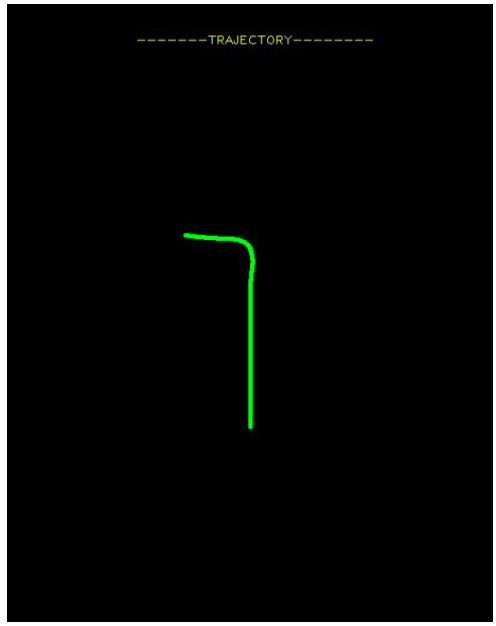


Fig. 7.3: Estimated Trajectory using Visual Odometry in KITTI

7.4 Results from NITCOD Dataset

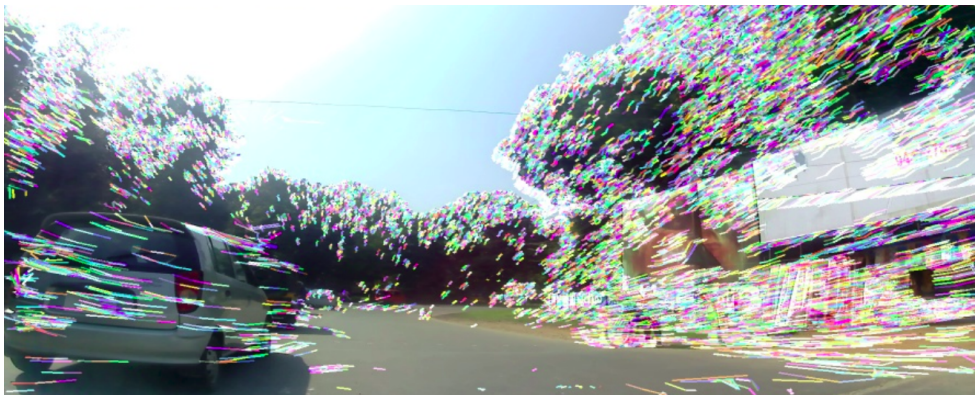


Fig. 7.4: Feature Tracking using KLT Tracker in NITCOD Dataset

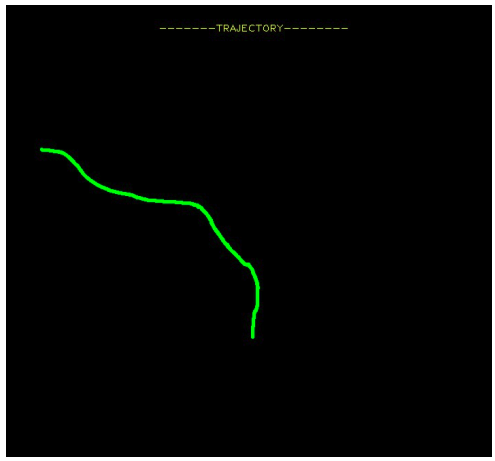


Fig. 7.5: Estimated Trajectory using Visual Odometry in NITCOD Dataset

Chapter 8

Velocity Estimation

Vehicle speed is an important traffic factor to study in autonomous driving systems. The first step in estimating the velocity of vehicles is detecting moving vehicles in all video frames. Detection of objects in the frame is done using You Only Look Once (YOLO) algorithm. YOLO detects different classes of objects in each frame and computes the bounding box around each object in every frame. Once the dimensions of the bounding box are obtained, these objects must be accurately tracked on at least two successive video frames. This is done using Scale Invariant Feature Transform (SIFT) as discussed in [28]. SIFT computes the number of matching features in two different images. The objects in the frame at time t and those in the frame at time t+1 are matched based on maximum number of matching features. The centroid of the objects that are matched are found and used to find the depth.

To have accurate speed measurement, it is necessary to generate a good depth map of the traffic scene. Disparity maps are generated by computing the horizontal difference of the coordinates of objects in the left and right frames, obtained from the stereo camera which is done in chapter 6 by MC-CNN. Depth of the objects at the position of its centroid are further calculated using 6.2. Later, we estimate instant vehicle speed from distance variation and Frame Per Second (FPS) which gives information about time elapsed between two captured consecutive images. [29] is a similar implementation of the above task.

The steps followed in estimating velocity of vehicles is given below as a flow chart

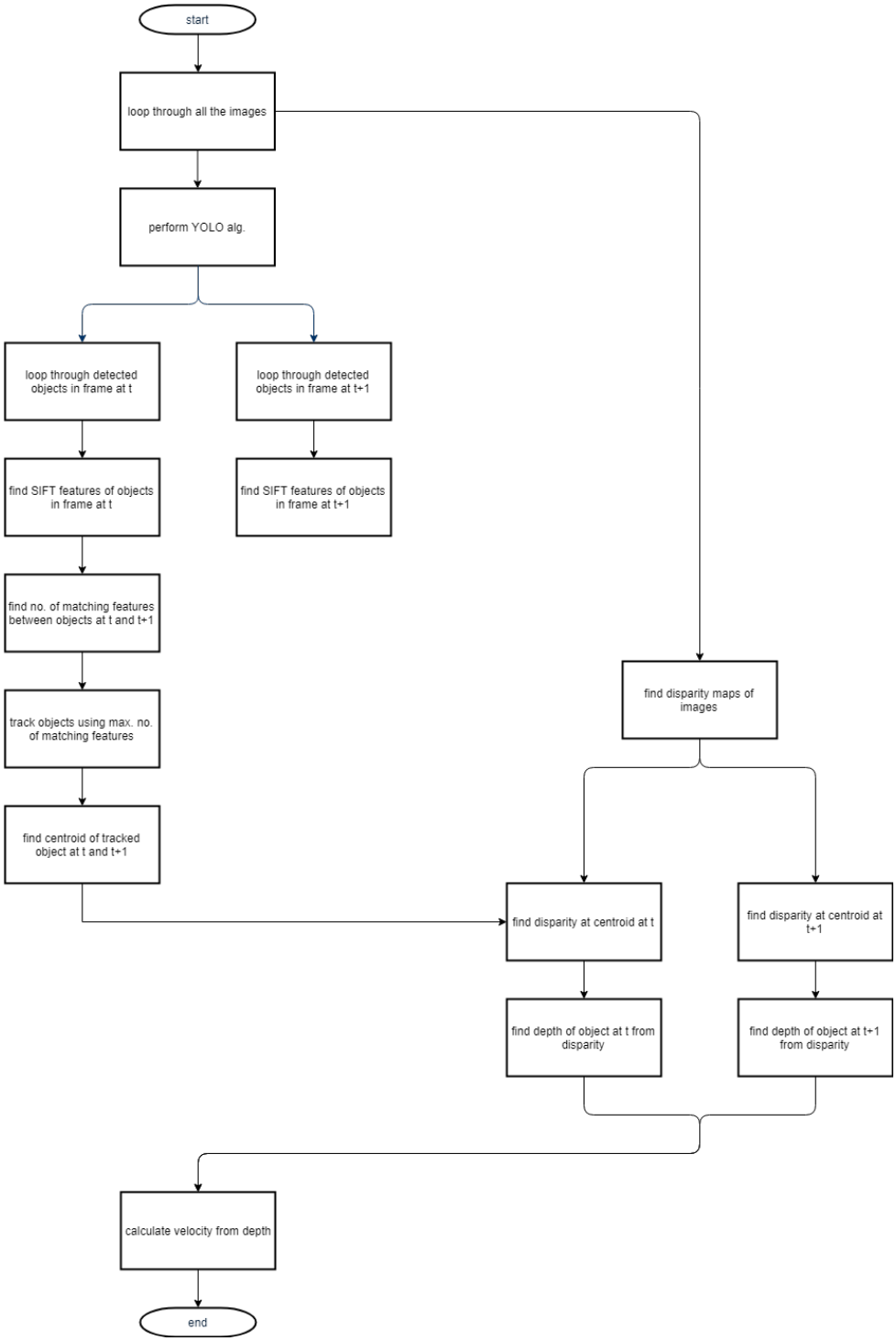


Fig. 8.1: Flow chart describing the steps to estimate the velocity

8.1 Scale Invariant Feature Transform

Image Matching is an essential part of various problems in object and scene recognition, stereo correspondence, motion tracking, and solving for 3D structure from multiple images. Matching different images of an object or scene can be done by utilizing the various properties of image features. This features are unalterable to rotation and scale and are proven to provide strong matching across a considerable range of affine distortion, change in 3D viewpoint, noise addition and illumination change. They are aptly confined in both frequency and spatial domains, thus minimising disruption probability by occlusion, clutter and noise. Moreover, these features are highly typical, which allows a single feature to be correctly matched with high probability against a large database of features, providing a support for object and scene recognition.
The following are the stages in SIFT algorithms as discussed in [28] which help in extracting the image features set.

1. Scale-space extrema detection: In this step, the algorithm searches over all scales and image locations. Efficient implementation can be done by using DOG(difference of Gaussians) function to identify the prospective feature points that are invariant to scale and rotation.

2. Keypoint localization: A comprehensive model is fit at each candidate location to determine scale and location. Keypoints are located based on their measures of stability.

3. Orientation assignment: Local Image gradients enables to assign additional orientations to each keypoint location. Invariance can be achieved by performing all the further operations on the image data that been transformed relative to assigned scale, orientation and location for each feature.

4. Keypoint descriptor: In the region around each keypoint local image gradients are measured at the selected scale. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

8.2 Result of velocity estimation

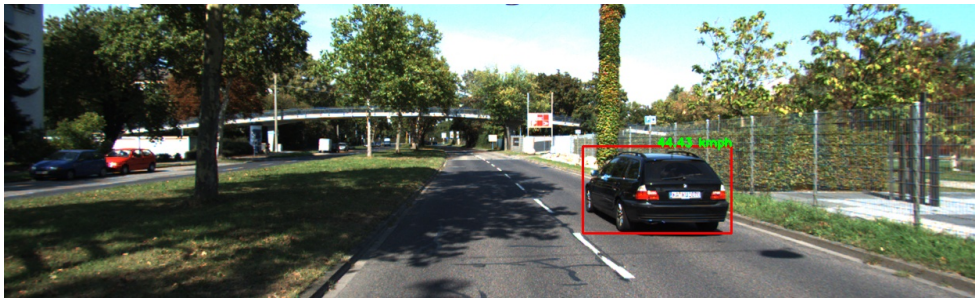


Fig. 8.2: Velocity estimation on KITTI

References

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[2] J. Behley, V. Steinhage, and A. B. Cremers, "Laser-based segment classification using a mixture of bag-of-words," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4195–4200.

[3] M. Himmelsbach, T. Luettel, and H.-J. Wuensche, "Real-time object classification in 3d point clouds using point feature histograms," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 994–1000.

[4] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *2010 IEEE Intelligent Vehicles Symposium*. IEEE, 2010, pp. 560–565.

[5] A. Teichman, J. Levinson, and S. Thrun, "Towards 3d object recognition via classification of arbitrary object tracks," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4034–4041.

[6] A. E. Johnson, "Spin-images: a representation for 3-d surface matching," 1997.

[7] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[9] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[10] R. Aufrère, J. Gowdy, C. Mertz, C. Thorpe, C.-C. Wang, and T. Yata, "Perception for collision avoidance and autonomous driving," *Mechatronics*, vol. 13, no. 10, pp. 1149–1161, 2003.

[11] C. Premebida, O. Ludwig, and U. Nunes, "Lidar and vision-based pedestrian detection system," *Journal of Field Robotics*, vol. 26, no. 9, pp. 696–711, 2009.

[12] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[13] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[14] I. Miller, M. Campbell, D. Huttenlocher, F.-R. Kline, A. Nathan, S. Lupashin, J. Catlin, B. Schimpf, P. Moran, N. Zych *et al.*, "Team cornell's skynet: Robust perception and planning in an urban environment," *Journal of Field Robotics*, vol. 25, no. 8, pp. 493–527, 2008.

[15] N. Lazaros, G. C. Sirakoulis, and A. Gasteratos, "Review of stereo vision algorithms: from software to hardware," *International Journal of Optomechatronics*, vol. 2, no. 4, pp. 435–462, 2008.

[16] C. Yang, Y. Li, W. Zhong, and S. Chen, "Real-time hardware stereo matching using guided image filter," in *2016 International Great Lakes Symposium on VLSI (GLSVLSI)*. IEEE, 2016, pp. 105–108.

[17] R. Kalarot and J. Morris, "Comparison of fpga and gpu implementations of real-time stereo vision," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. IEEE, 2010, pp. 9–15.

[18] Z. Liang, Y. Feng, Y. Guo, H. Liu, W. Chen, L. Qiao, L. Zhou, and J. Zhang, "Learning for disparity estimation through feature constancy," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2811–2820.

[19] J. Zbontar, Y. LeCun *et al.*, "Stereo matching by training a convolutional neural network to compare image patches." *Journal of Machine Learning Research*, vol. 17, no. 1-32, p. 2, 2016.

[20] D. Sivia and J. Skilling, *Data analysis: a Bayesian tutorial*. OUP Oxford, 2006.

[21] C. E. Rasmussen and C. Williams, "Gaussian processes for machine learning the mit press," 2006.

[22] T. Chen, B. Dai, R. Wang, and D. Liu, "Gaussian-process-based real-time ground segmentation for autonomous land vehicles," *Journal of Intelligent & Robotic Systems*, vol. 76, no. 3-4, pp. 563–582, 2014.

[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[24] K. E. Van de Sande, J. R. Uijlings, T. Gevers, A. W. Smeulders *et al.*, "Segmentation as selective search for object recognition." in *ICCV*, vol. 1, no. 2, 2011, p. 7.

[25] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[26] H. Hirschmuller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 9, pp. 1582–1599, 2009.

[27] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, vol. 1, May 2006, pp. 430–443. [Online]. Available: http://www.edwardrosten.com/work/rosten_2006_machine.pdf

[28] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[29] A. El Bouziady, R. O. H. Thami, M. Ghogho, O. Bourja, and S. El Fkihi, "Vehicle speed estimation using extracted surf features from stereo images," in *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*. IEEE, 2018, pp. 1–6.