Computer Vision
Task 2-OBJECT DETECTION AND DEPTH ESTIMATION

Pravin Tulshidas Palve, Matriculation Number-13541620
Manal Ajit Vartak, Matriculation Number-29241614

June 9, 2024

1 Introduction

This task aims to extract depth information from a 2D image using internal camera characteristics, also known as the Intrinsic Matrix (IM). The dataset we are using [1] is a selectively sampled subset from the KITTI dataset [2]. The calculated distances will be compared with the distances provided in the dataset labels. We are using the YOLOv5s [4] object detection model to detect and locate cars in the scene. Note that detection is limited to cars, and all other objects such as people, buses, and trucks are filtered out.
Further into the report, we will discuss methodology, visualize results and analyze them.

2 Methodology

Python was chosen as the programming language due to its robust object detection models and efficient data handling and visualization libraries. Before we dive in, we will need the following libraries: `os, sys, numpy, cv2, csv, torch, shapely, matplotlib`. These libraries will allow us to access, read, manipulate, detect, combine, slice, process, convert, plot, and write various data types. Below, we outline the strategy of our solution.

2.1 Object Detection

To calculate distance, we first need to locate the object in the image. This was the simplest part of the task, thanks to the availability of various pre-trained and open-source models. One widely used model is YOLO (You Only Look Once), known for its strong documentation and community support. As mentioned earlier, we are using YOLOv5s.
The inner workings of object detection models are beyond the scope of this report. We will treat them as a black box that takes an image as input and outputs bounding boxes with classification labels. For those interested in the details, further reading can be found in the cited article [3].

2.2 2D to 3D

In this section, we will discuss how to transition from 2D to 3D. Before that, to make things easier for the reader to comprehend, let's briefly discuss an important term.
Intrinsic Matrix: The intrinsic camera matrix, also called the camera calibration matrix, is a 3x3 matrix that captures the internal properties of a camera. These parameters influence

how the 3D world is projected onto the camera's 2D image plane. In simple terms, this matrix defines which pixels will record specific points in the 3D space of camera coordinates.

$$\mathbf{K} = \begin{bmatrix} f.k & skw & s_1 \\ 0 & f.k & s_2 \\ 0 & 0 & 1 \end{bmatrix}$$

where:
1) $f.k$- Focal length in pixels: Represents the distance between the image plane and the optical center. Here, f is a focal length and k is pixel density (usually same in x and y directions).
2)skw- Skew coefficient: Accounts for any non-perpendicularity between the image sensor rows and columns (often zero).
3)s_1 and s_2- Principal point: Represent the the offset between the optical center and the image origin.
Returning to our main discussion of expanding the dimension, let's split the entire process into different steps:

1. **Augmenting Dimensions**- The pixels have only z1 and z2 coordinates. We add '1' as 3rd coordinate in every pixel coordinate. This can be simply done by `append` function in Python. Here is what it should look like in mathematical terms-

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \longrightarrow \begin{bmatrix} z_1 \\ z_2 \\ 1 \end{bmatrix}$$

2. **Partial Inversion of Projection**- To convert from pixel coordinates to world coordinates, we utilize the following relation [5], where the intrinsic matrix helps us to reverse map the pixel coordinates into the 3D world. In our case, the camera coordinates are identical to the world coordinates.

$$\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} = z \cdot \mathbf{K}^{-1} \begin{pmatrix} z_1 \\ z_2 \\ 1 \end{pmatrix} \tag{1}$$

   In this transformation, the world coordinates (X_w, Y_w, Z_w) are derived from the pixel coordinates $(z_1, z_2, 1)$ using the intrinsic matrix $\mathbf{K}$ and the scalar z. Cited document [5] explains these things very intuitively.

3. **The Catch**- If you've been following closely so far, you've likely noticed the catch. The scalar 'z' in the previous relation is unknown. This means that the point corresponding to a certain pixel could lie anywhere on an infinite line along the (Z_w) axis. This is where a crucial piece of information from the problem statement comes into play.

   The camera used to capture the images is mounted at a height of 1.65 meters from the driving plane and is aligned with the horizon. We assume the driving plane to be perfectly flat, and both the Ego and Target cars are driving on the same plane. Given these two hints, it's intuitive that selecting the bottom point of the bounding box will provide us with the intersection of the camera's line of sight and the driving plane. For further understanding, refer to Figure 1.

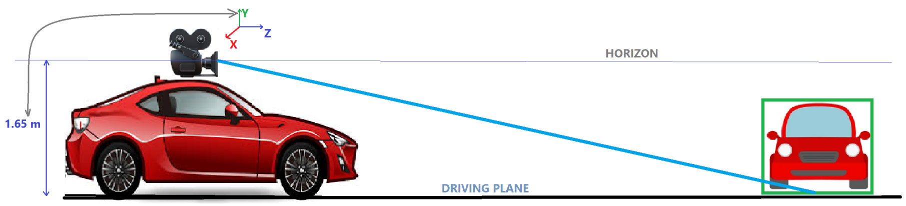

Figure 1

4. **Finding the unknowns-** From all the information that we have, we can plug the known parameters in the equation (1) and get the following...

$$\begin{pmatrix} X_w \\ 1.65 \\ Z_w \end{pmatrix} = z \cdot [\#] \begin{pmatrix} (\#) \\ (\#) \\ 1 \end{pmatrix}$$

where #'s are some known values which we have calculated and/or got from object detector. The terms z and (X_w, Z_w) are still unknown. But if we solve for z using $(Y_w = 1.65m)$, we will get the value of scalar quantity z. Hence we can also solve for (X_w, Z_w).

2.3 Matching With Labels

Upon plotting the labels provided in the dataset on the image, one will notice that they do not cover all the cars in the scene. Conversely, we have observed that the object detector detects more cars than the labels in the scene. Refer to a sample image (id-006042) from the dataset with labels (in blue) and detections (in green) plotted in Figure 2. Additionally, the sequence of the labels and bounding boxes provided by the object detector do not match.

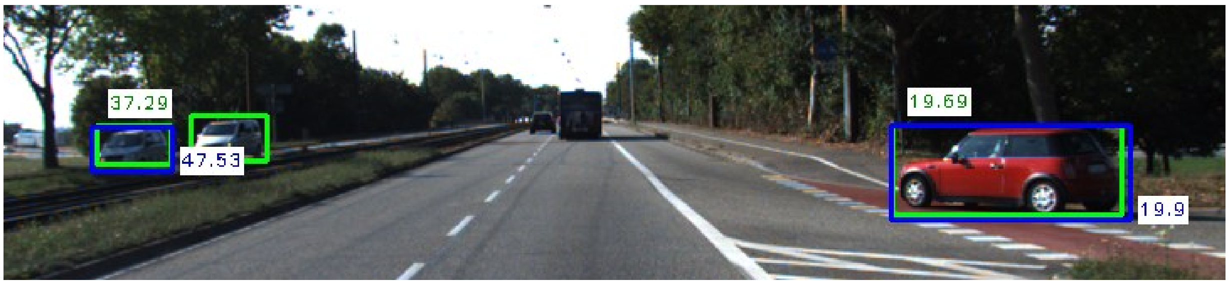

Figure 2

To eliminate extra detections and match the detections with corresponding labels, we employed a technique called Intersection over Union (IOU). It is simply a ratio of the intersection area of two overlapping boxes to the union of those same boxes. The mathematical expression is shown in Figure 3. Our algorithm checks the IoU of a given detection with all the labels and

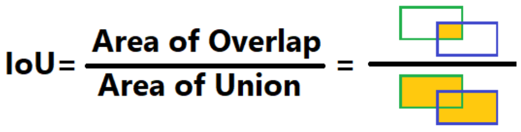

Figure 3

selects the label with the maximum IoU as a match. We also set the IoU threshold to be 0.4 for any detection to be considered as a match because in crowded scenarios, multiple boxes overlap each other (as shown in Figure 4, dataset id-006211), which may lead to mismatches between

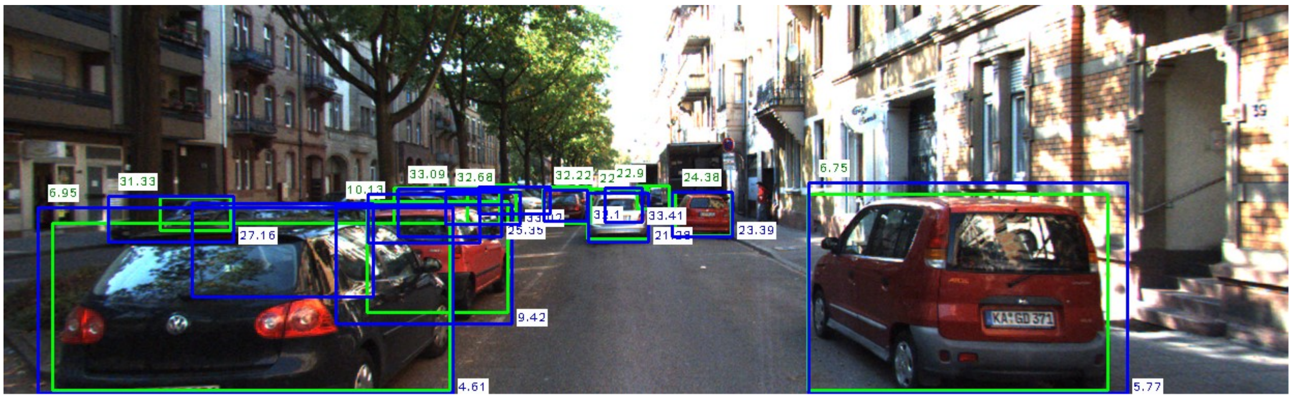

Figure 4

labels and detections. The distances corresponding to matched labels and detections are then stored in lists. These lists are plotted as scatter charts using `matplotlib`. We will discuss this charts in upcoming section of the report.

3 Results

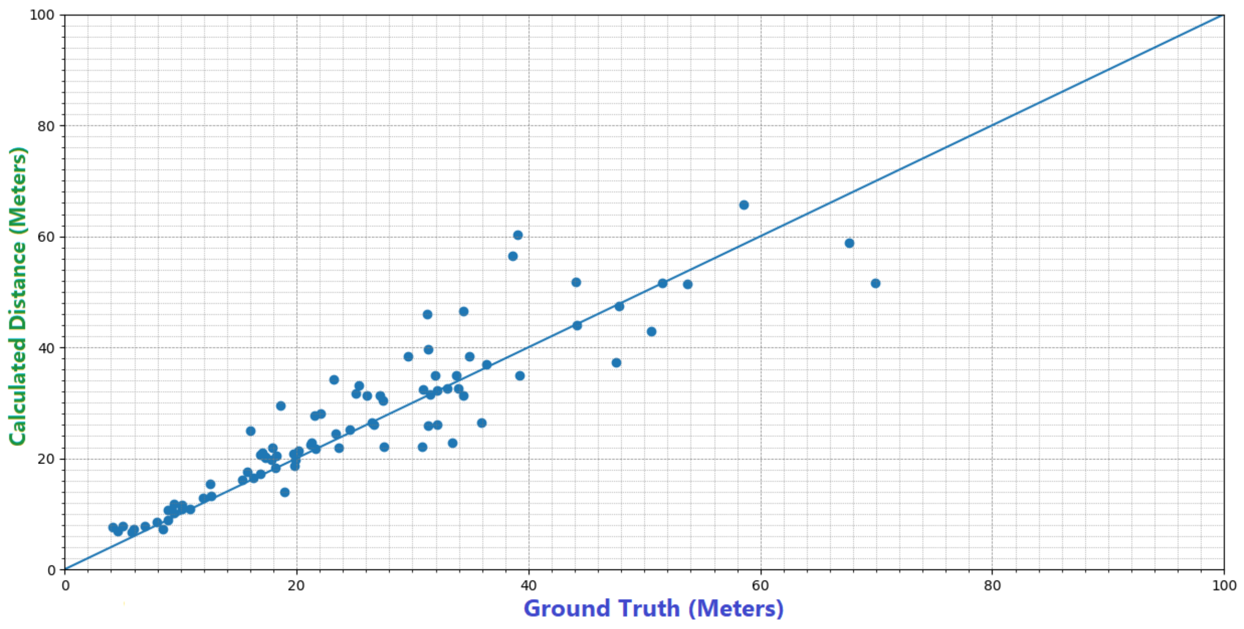

Figure 5: Plot of Ground Truth vs Calculated Distances

Here is the plot of distances of all matched detections and their corresponding labels. The closer the points are to the diagonal line in the plot, the more accurate the results. Before we proceed with further discussion, let us first establish some declarations-

1. Total number of images in data set- 20
2. Total Number of Labels in data set-98
3. total bounding boxes detected by YOLOv5s in all images- 117
4. total points plotted in Figure 5 (scatter graph)- 86

There are two main reasons behind this difference of 12 missing data points-

1. There were no detections by YOLO for a given label in some images. For example refer Figure 6, data set id-006098. A brief overview of such cases can be found in table below.

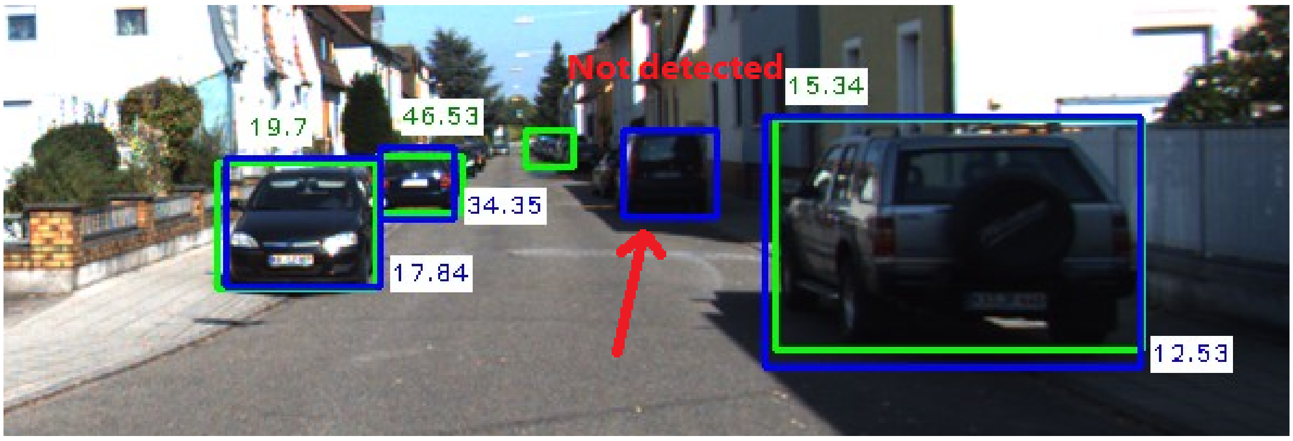

Figure 6: Example of undetected labeled car.

Img. ID	No. of undetected cars
006037	1
006048	1
006054	1
006098	1
006211	3
006310	1
006312	1

Table 1: Undetected labeled cars in each Image

2. We filtered out anomalies with extreme values ($> 100\%$ error) because these points were causing trouble in the proportional visualization of the rest of the data. We have listed these outliers in the table below to ensure that we don't overlook any aspect of the actual results. We will delve into these anomalies further in the report with a detailed explanation.

Img. ID	Ground Truth (m)	Estimation (m)
006097	2.62	7.23
006291	1.80	7.19
006310	67.33	232.82

Table 2: Details of Anomalies

Now let us get back to discussing the results.

- Between 6.5 meters and 16 meters, the predictions remained relatively close to the ground truths. However, beyond 16 meters, as the distance of the cars increases, the difference between predictions and labels gradually increases up to 40 meters. This trend is observable in the Error distance vs. Ground truth graph depicted in Figure 7.

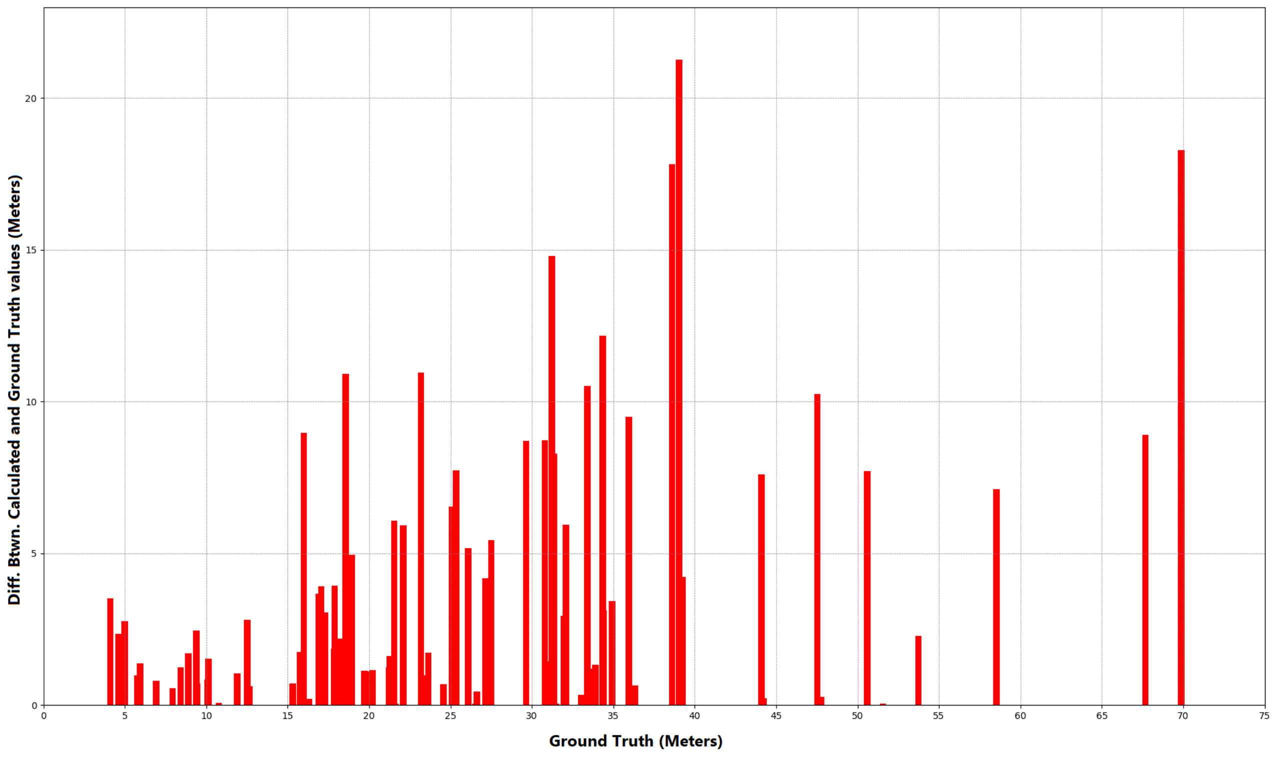

Figure 7: Prediction error in Meters vs. Ground Truths

- Beyond 40 meters, there are not enough data points available to establish the trend and draw a concrete conclusion. However, based on Figure 7, a ballpark estimation can be made that there is an average prediction error of 8 meters.
- Before we make any further conclusions, we need to go through the % error in predictions vs Ground Truth chart in Figure 8.

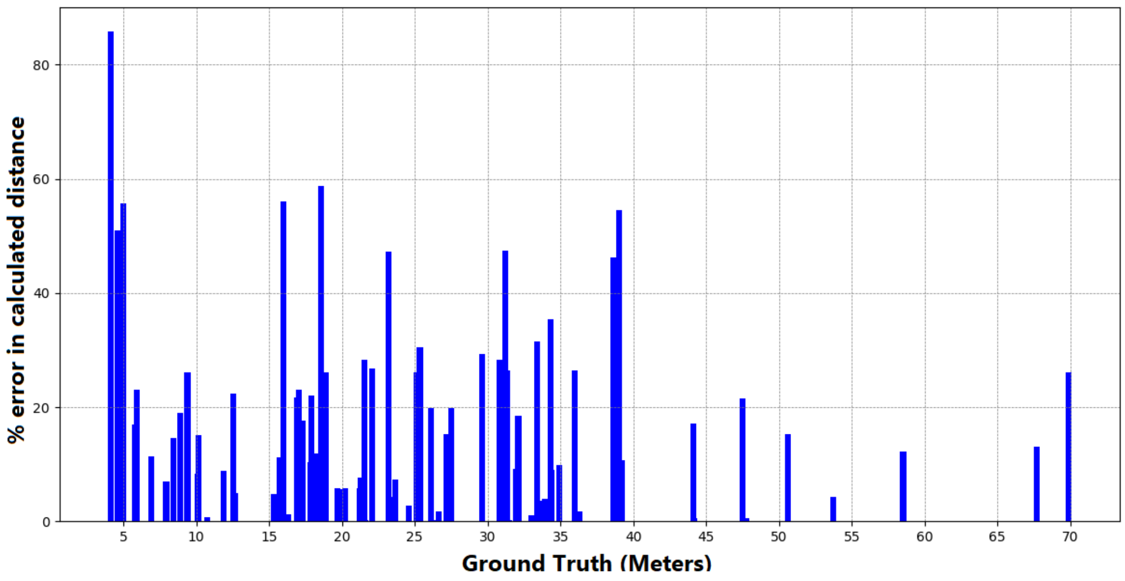

Figure 8: Prediction error in % vs. Ground Truths

- Figure 8 makes us realize that average % error in predictions remains within a constant range of 18% to 20% in broad perspective. The reasons behind this behavior might be dependent on the accuracy of object detector or accuracy of Intrinsic matrix of camera.

- Another important behavior we observed is that for any ground truth values less than 7 meters, our algorithm returns values close to 7 meters. This phenomenon is also the reason behind the occurrence of the first two anomalies in Table 2. Upon close inspection of images related to these anomalies, this behavior can be explained as follows:

  1. In simple terms, the camera's field of view starts from approximately 7 meters for an ideal driving plane. So, even if cars in the image are at 2 meters, we will still obtain the minimum possible distance as per the camera, which is around 7 meters.
  2. In more detailed terms, if we calculate the distance for the bottom row pixels of the camera image, they will map us back to a distance close to 7 meters. For cars that are too close to the camera and only partially visible in the image from the bottom, their corresponding bounding boxes would always be at the bottom of the image, thereby returning a distance of approximately 7 meters. An example of such a scenario, along with an explanation, can be found in Figure 9.

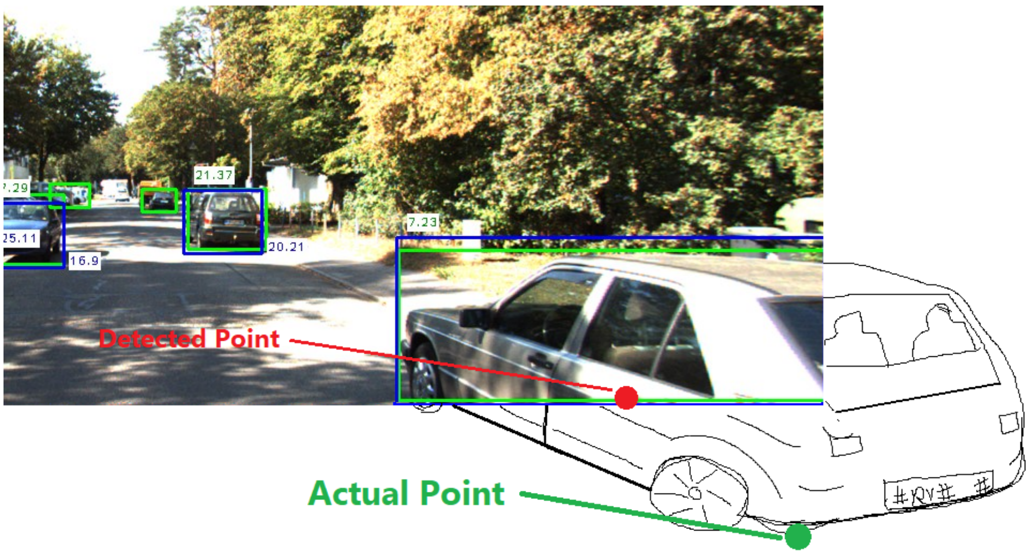

Figure 9: Anomaly in short distances

- The reason behind 3rd anomaly in the table 2 is pretty intuitive. If we refer to the Figure 10, which is the image corresponding to this anomaly, we can notice the recording is from hilly area and the driving plane is has gradually increasing gradient. This positioning causes the target car to be placed significantly higher in the image, leading our algorithm to perceive it as farther away.

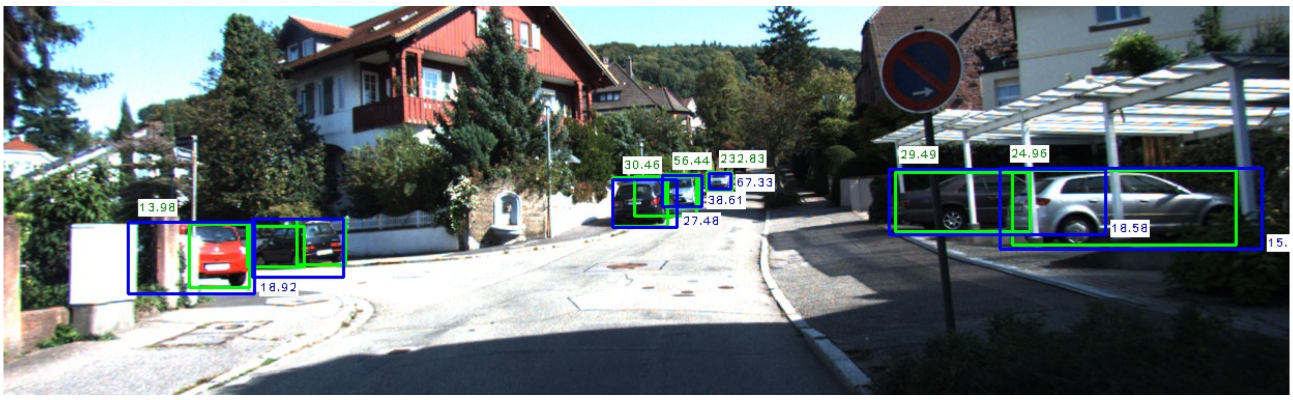

Figure 10: Recording from hilly area

4 Conclusion

In hindsight of the results, the error percentage and anomalies observed in depth estimation using a single camera are quite significant, indicating that it should not be relied upon as the sole source of depth information. Furthermore, parameters such as the uneven gradient of the driving plane, lens distortion, object detector accuracy, and proper calibration play crucial roles in the overall performance.
Despite its drawbacks, depth estimation using a monocular camera can still be implemented in highway scenarios, particularly in vehicles without radar sensors, to maintain a safe distance and alert drivers in case of close proximity.

5 Further Scope

In the future scope of this task, we can explore even larger datasets to collect more data points and assess the performance and reliability of monocular cameras more comprehensively. Additionally, we can consider implementing different object detectors with higher accuracy performances.

References

[1] KITTI selection Internal link- https://elearning.rwu.de/mod/resource/view.php?id=193664

[2] KITTI dataset homepage- https://www.cvlibs.net/datasets/kitti/index.php

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. https://arxiv.org/abs/1506.02640v5

[4] YOLO Documentation Link- https://github.com/ultralytics/yolov5

[5] Dr. rer. nat. STEFAN ELSER, Ravensburg Weingarten University Of Applied Science, Computer Vision- Chapter 7: 3D TO 2D PROJECTIONS, 2D TO 3D PROJECTIONS AND 3D ROTATIONS

[6] Dr. rer. nat. STEFAN ELSER, Ravensburg Weingarten University Of Applied Science, Computer Vision- Chapter 4: Geometric Transformation and Simple Model Descriptions, Slide 51