**MACHINE LEARNING - GROUP PROJECT**

**Report**

**NAME ID No.s**

| Ratnaker Gautam | 2019A4PS0990H |
| --- | --- |
| Kanduri Saranya | 2019A3PS1318H |
| Haarika Manda | 2019AAPS1226H |
| B Sriharsh | 2019A7PS0165H |
| Chitrapu Sai Sudarsan | 2019A3PS0461H |
| Siddharth Dronamraju | 2019A4PS0978H |

**Introduction**

There is a surge in interest in autonomous vehicles and automated driving systems in recent times in both research/ academia and in industry. One of the main features of automated driving systems would be safety. Safety of both the driver and the people and vehicles on the road. This can be achieved by using advanced deep learning architectures and computer vision. The safety of a system can be measured using two important parameters. One is the reaction time of the system (fastness) and the accuracy with which the system identifies the object. So, a faster model that can work in real-time and yet have a good accuracy is needed to tackle this problem.

In this report, we have attempted to solve problems pertaining to this category- Object detection and depth estimation.

**Related works and methodology- *Object detection and depth estimation***

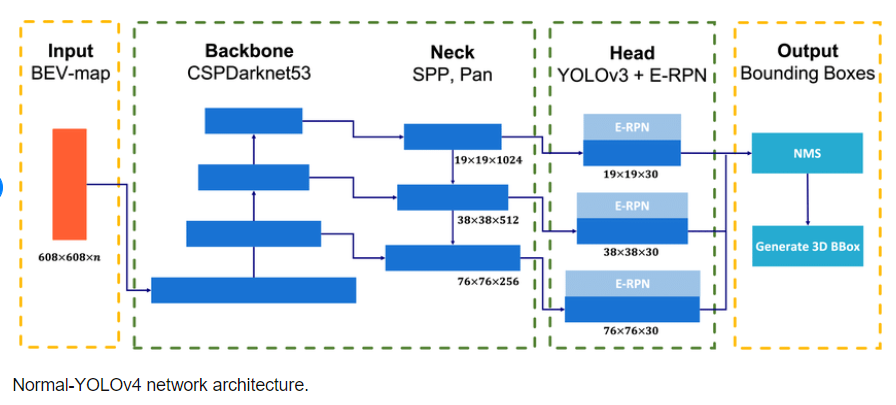
This model is used to detect pedestrians, cars, and other objects from the live camera or lidar sensors and estimate the distance of these objects from the sensors. This helps in planning the maneuvers of the vehicle by avoiding the obstacles. As mentioned above, this has two sub-tasks – object detection and depth estimation.

Our project here focuses on object detection and depth estimation using YOLO and OpenCV framework, and Tensorflow. We attempted to develop a machine learning model capable of locating and identifying objects while also calculating depth.

1.1. Object detection

The YOLO v4 algorithm that we are using is a one-stage model thus making it faster compared to its two-stage counterparts like faster-RCNN and R-FCN. This makes YOLOv4 a computationally light algorithm used for real-time purposes. When it comes to accuracy, scaled- YOLOv4 is as accurate as EfficientDet (a multi-stage detector) and yet 3.7 times faster.

Using CSPDarknet53 as a backbone, SPP (Spatial Pyramid Pooling) and PAN (Path Aggregation Network) for "the Neck," and YOLOv3 for "the Head," YOLOv4 is built based on the latest research findings.



Unlike the previous models, YOLOv4 came with a set of features that made it stand out. Some of these improvements were:

a) Increase in inference accuracy without any additional cost to the hardware. This is done by employing mosaic data augmentation techniques. This method forces the network to generalize to new features that can help it with detection.

b) A set of techniques that significantly increase the accuracy for very little inference cost were used. These techniques include

· Using the MISH function which is an activation function designed to push signals to the left and right.

· Another technique called DIoU- NMS (Non-maximum separation Distance IoU) to separate out predicted bounding boxes.

· And drop block regularization was utilized to force our model to learn features that it may not otherwise rely upon by removing the first layer of the network.

1.2. Depth estimation

Our model uses an image-based stereo vision depth estimation method to generate a disparity map. The ability to perceive depth due to two different perspectives of an object is called stereopsis. This phenomenon is used in calculating depth from stereo images.

We are using the Stereo-block matching algorithm to find the disparity among the right and left stereo images. This disparity is internally used to calculate depth. We used the Sum of absolute differences as our cost function for finding disparity. The pair with the lowest SAD score is the best match as the overall pixel-wise sum is the least. OpenCV provides an implementation for the Block Matching algorithm – StereoBM class. The method of the StereoBM class calculates a disparity map for a pair of rectified stereo images.

**Experiments conducted**

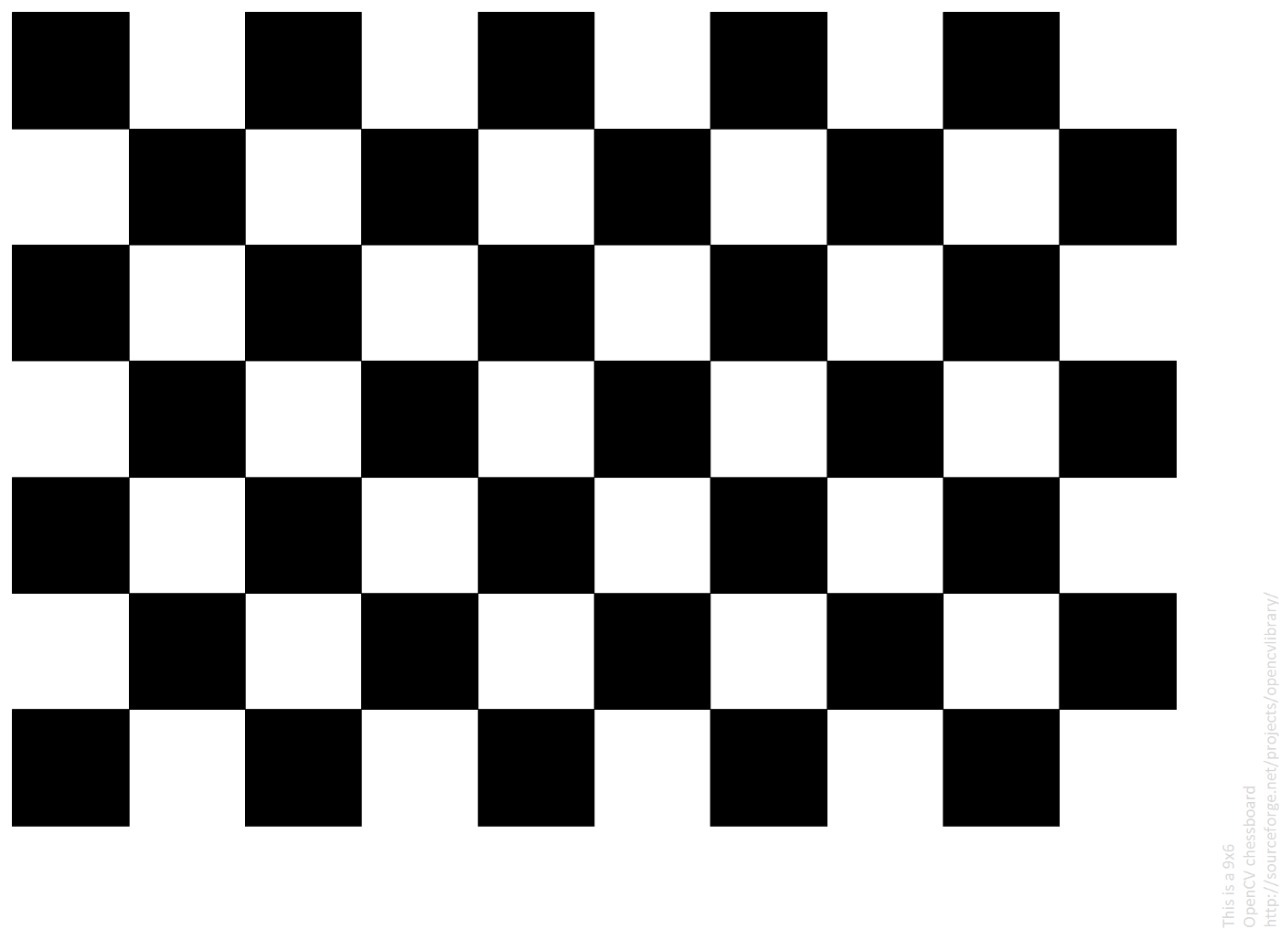
We used the available public datasets on Roboflow to train our model for object detection and obtain respective weights.

About Roboflow

Roboflow is an online platform that allows us to train our models using publicly available datasets, as well as construct and annotate our own. The use of Roboflow is justified because it streamlines the entire training process. We used a Colab notebook to import the Roboflow libraries, and then we used our custom dataset to train our model. The model's weights are then obtained and used for further analysis.

About Camera Calibration

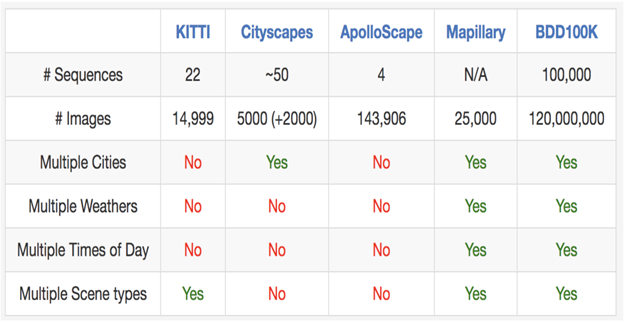
Camera Calibration essentially estimates the parameters that are present in the specific camera that is being used for the experiment. The goal of calibration is that we know for every pixel what is the precise direction vector from the origin that is our camera to the 3D world, that is, to establish a precise relationship between a 3D point in the real world and the 2D projection (pixel) in the image captured by that calibrated camera. To determine the projection of a 3D point onto the picture plane, we must first transform the point from world to camera coordinate systems using extrinsic parameters (Rotation R and Translation T). It also entails recovering the internal parameters of the camera lens. This includes the optical center, focal length, and the radial distortion coefficients of the lens.

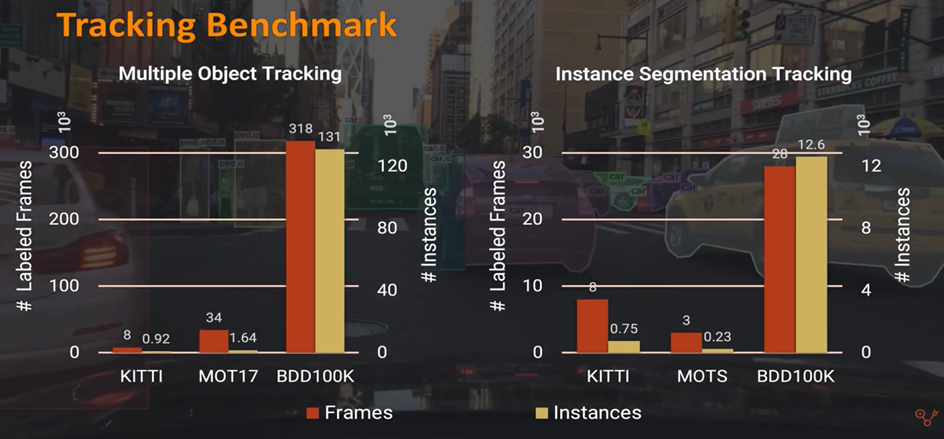


**Dataset**

The dataset we used was BDD-100K. It is a large dataset with around 100K videos and 10 tasks. Since data diversity is important for the robustness of perception algorithms, the dataset has a diverse geography, environment, and weather. The coverage of a variety of weather conditions especially gives it an edge.

Comparison between BDD100k and other datasets

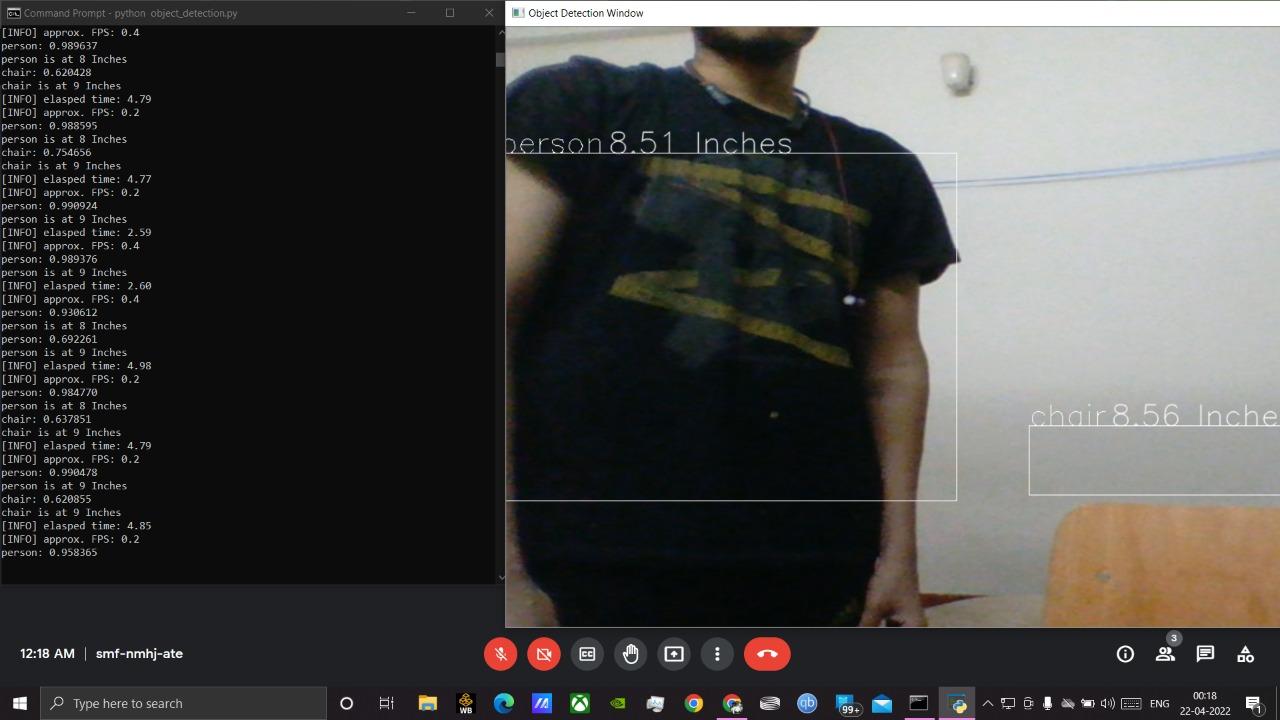
****



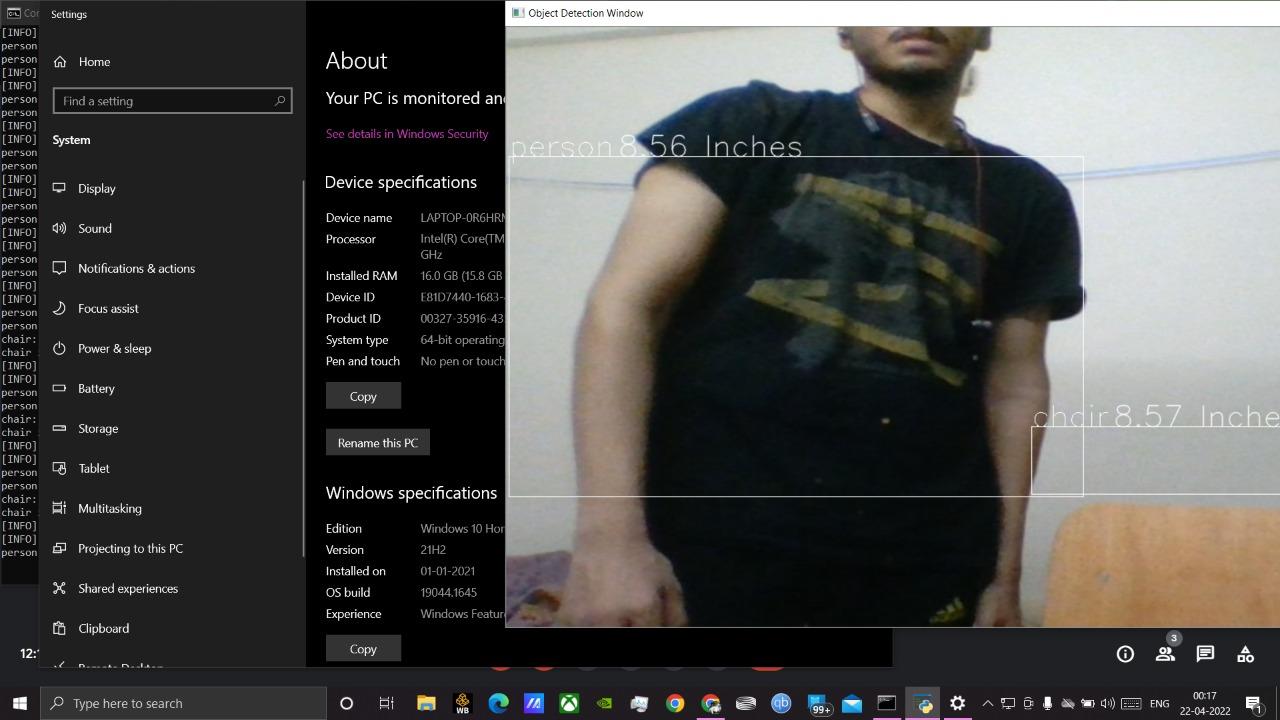
This dataset is divided in a 78:22 ratio into train and test directories. These directories shall be used to feed into our models.

**Results and Main findings**

Screenshots of the obtained results



The object detector is able to detect both the person and the chair, also presenting their respective dimensions in a single frame, where the left part of the picture indicates the uncertainties and probabilities involved.



We find the detector producing acceptable results for overlapping objects as well, where in the above case both the objects ( the person and the chair) are overlapping to a small extent.

Here is another example of the output obtained when the detector was run on a webcam.



**Accomplishments and Conclusions**

We were able to produce satisfactory results where the detector was tested for different distances and lengths ranging from 0 to 100 centimeters (or 1 meter). We are yet to test the same setup and model for larger distances and obtain accuracy and uncertainty results.

**Innovation**

Use the results obtained from this project to work on a distracted driver setup where the driver might be distracted because there are several ways for a driver to become distracted while driving and it is impractical to manually detect each instance of distraction. To avoid any mishappenings or any illegal driving practices, we can use this setup, where we can use the current setup to detect the distraction or the activity that the driver is performing and put the certain activity in a particular class, say for example Class C1 corresponds to a safe driving situation, Class C2- driver driving while drunk, Class C3- talking on the phone and so on. So the model can be fed with already present examples of the driver under multiple circumstances as mentioned above and train it on this dataset. Object detection can be done so as to classify the activity into a distracted or safe situation.

—------------------------------------------------------------------------------------------------------------------

**Group members and contributions**

| Ratnaker Gautam | 2019A4PS0990H | Coding and implementation, working with Roboflow |
| --- | --- | --- |
| Kanduri Saranya | 2019A3PS1318H | Research and analysis to make the model better,  Innovation and report |
| Haarika Manda | 2019AAPS1226H | Obtaining weights and Model training |
| B Sriharsh | 2019A7PS0165H | Getting the dataset, splitting and filtering |
| Chitrapu Sai Sudarsan | 2019A3PS0461H | Research to make the model better and analysis, report |
| Siddharth Dronamraju | 2019A4PS0978H | Working with YOLO architecture |

**Github link -** <https://github.com/RATNAKER555/OBJ_DET-DEPTH_EST>