BILKENT UNIVERSITY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Distance Map Extraction and Road Defect Detection Using Stereo Vision

Committee Meeting IV Report
20.05.2022

Group C2 Members

Barış Aşkın - 21702503
Şafak Çallıoğlu - 21702106
Emirhan İlhan - 21801662
Fatih Berkay Sarpkaya - 21702989
Muhammed Ustaömeroğlu - 21703863
Yavuz Yarıcı - 21703075

Academic Mentor

Prof. Dr. Levent Onural

Company Mentor

Talha Kaan Topuz

Teaching Assistant

Aslı Alpman

Nurol Makina ve Sanayi A.Ş (Nurol Makina) was founded in 1976 to address our country's requirements for turnkey industrial facilities, steel construction, and machinery production. It has completed numerous large-scale contracting projects and successfully submitted them. Nurol Makina has been engaged in the defense business since 1992, and its modern facilities in Ankara continue to produce 4x4 tactical wheeled armored vehicles.

Abstract

With the advancements in computer vision and the capability of our processors, real-time methods used in driving systems have become more important. Detecting the objects on the road and locating them is an open problem. In defense industry, especially for land vehicles, autonomous object detection and distance estimation are crucial. The purpose of this project is to detect potholes and speed bumps on the road and estimate their relative distances to the user by utilizing stereo vision data. To this end, a stereo camera was used for data acquisition, and a personal computer was used as a processing unit. The data from the stereo camera was first processed with a deep learning-based object detection method. Then, the disparity map of two cameras was extracted and used to estimate the distance from detected objects to the camera. The baseline algorithms existing in the current literature and the built-in algorithm in the camera were tried. After getting results from the object detection and distance estimation tasks, we matched their outputs on real-time frames, and we projected them onto a two-dimensional surface considering perspective shifts. We basically created a map on which the detected objects and their distances to the car are shown. YOLOv5 was used as an object detection deep network. The network was trained on Google Colab GPUs and tested on a personal computer. For simulation of our network, we tried our method on both saved video and real-time experiments. We mainly used PyTorch and OpenCV frameworks with Python programming language to train and test our network. We searched on the Internet for training datasets and collected an additional dataset in Ankara. After a literature survey, we implemented block matching (BM) and semi-global block matching (SGBM) in addition to the built-in algorithm of the camera for distance estimation. We worked on improving the accuracy and speed of all methods. We validated the results of each algorithm in a testing environment, the roads of Ankara, by measuring the distances of the test objects. Also, we provided users with controls on object detection region and confidence threshold through the interface. We used OpenCV and ZED SDK, Python package of the selected camera, to obtain the visual data and extract disparity information. The requirements and expected results are discussed in the report. Our aim is to achieve all of the requirements in the testing environment, which is the roads of Ankara. The end product is capable of detecting 80% of potholes and 85% of speed bumps. The final distance estimation error is 5% up to 10 meters and 12% from 10 to 20 meters. The complete project with all work packages including GUI works with 16 frame-per-second (FPS) in real-time tests. The outcomes of this project can be used in a single car or swarm systems with and without a driver inside.

Contents

List of Figures

1 Motivation and Novelty

Real-time driver assistance systems have been developed rapidly in recent years with the applications of new computer vision methods and improvements in processing units. Within this context, detecting the obstacles on the road and informing the driver of their positions is an important problem for the sake of safety. In defense industry, especially for land vehicles, autonomous detection of obstacles is crucial as these obstacles can potentially cause damage to the vehicles and risk to the drivers. Even on city roads, damages caused by potholes cost an average of $3 billion annually in the US [4]. Besides this high economic burden, the driver's late perception of the obstacles on the road causes traffic accidents, which poses a danger to the driver's health. As the economic and health issues are analyzed, it can be concluded that this is a national and global problem that affects drivers all around the world and has to be solved as soon as possible. Furthermore, in countries where roads have more frequent defects, the problem is more severe. The purpose of our project is to detect potholes and speed bumps on the city roads and estimate their relative distances to the user by utilizing stereo vision data. As a company that specializes in armored land vehicles, Nurol Makina is also investigating ways to utilize real-time methods that could provide assistance to drivers. Our project focuses on city roads and therefore it is not immediate use to Nurol Makina which manufactures off-road military vehicles. However, the results and findings of our project can give insights into the feasibility of the concept and can be applied to off-road vehicles of Nurol Makina with some adjustments in equipment and training data. After successfully implementing this project, the company expects a decrease in vehicle damage and accidents due to road obstacles. The target end-user of our product is car drivers, and our product does not have any significant cost compared to the vehicle's cost. The use of the product does not require any technical background. Drivers can easily see the type of obstacle and its distance to the vehicle from the in-vehicle display while driving.

There are some similar studies in the literature on pothole and speed-bump detection using deep learning with a camera input. However, these studies focus on either only the potholes [5–7] or only the speed bumps [8] and none of them combines the two classes of objects. In terms of the type of obstacles, the disadvantage of these projects is to detect just one type of obstacle. As for the functionality of the projects, these projects do not include distance estimation of the obstacles and only [8] combine object detection and distance estimation in order to provide the relative positions of the detected objects. However, as mentioned above, this project just detects the speed bumps on the road. Therefore, our project is the first in the literature to detect these two object classes and combine object detection with distance estimation. We are not planning to receive a patent upon completion of our project. After searching for patent solutions that address our problem, it is found that [9] is a solution to detect speed bumps. In this solution, an RFID tag is placed on the speed bumps and an RFID reader on the car detecting the speed bump on the road notifies the driver about

the detection. In this national patented solution, the aim is the same as our solution but the method used is different. Also, their product is not able to detect potholes. Moreover, an international solution is found. [10] uses a video recorder that includes a pothole detector. With this detector, the driver is informed about the potholes on the road. The aim and the solution technique are very similar to our solution.

2 Requirements

2.1 Functional Requirements

The aim of our project is to make real-time object detection and distance extraction. The target objects to be detected are chosen to be potholes and speed bumps. Our project aims at detecting and classifying potholes and speed bumps on the road and finding the distance of the car to them. These findings are illustrated in a 2D map and a real-time video is projected with these findings.

The company gave several requirements to satisfy all of the tasks in order to meet the needs of the company. We listed our requirements so that we can design our system accordingly.

#	Requirement	Justification	Comment
SFR-1	Potholes and speed bumps on the roads should be detected in the range of 20 meters	The type of camera allows 20-meter range and the company asked maximum this distance in this project	This distance could be extended with different camera setups.
SFR-2	70% of all objects should be detected and classified	The accuracy is proposed by company.	The accuracy could be increased by improving the training model.
SFR-3	The FPS (frame per second) is expected to be 18	18 FPS is acceptable for the company in this project.	FPS can be increased by using more powerful systems.
SFR-4	The error of depth estimation should be less than 1% up to 3 meters distance and less than 10% in the range of 3-20 meters	The error requirements are proposed by company regarding the user experience.	
SFR-5	Speed limit of the car is 32.4km/h	There is a speed limit of the car due to our systems FPS limit which is taken as 18.	
SFR-6	A user interface will show the detected objects, distances on camera's visual data and extracted map. User can control the detection confidence threshold.	User can control threshold by their preferences	

Figure 1: Functional requirements of the system.

1) The range of object detection: Depending on the type of camera, there is a constraint on the distance from which the objects are detected. For the camera we select, it is proposed that we

should detect potholes and speed bumps on the roads in the range of 20 meters.

2) Object detection accuracy: Our project needs to detect objects with pretty high accuracy. With IoU (Intersection over Union), which is calculated as dividing the area of overlap between the bounding boxes by the area of union, a threshold of 50%, the recall of object detection should be at least 70% which means that 70% of all objects are detected and classified correctly. And the precision of object detection should be at least 70%.

3) FPS: In our project, keeping track of the detected objects on a map is essential for the other drivers to avoid these obstacles. For the whole system, the FPS (frame per second) is expected to be 18.

4) Real-time depth estimation accuracy: Real-time depth estimation is important for the purpose of our project since it introduces huge errors if it is not properly taken care of, as it is explained in Section 3.2.3. Therefore, we need to seek high accuracy in real-time depth estimation. The error of depth estimation is selected to be less than 1% up to 3 meters distance and it is desired to be less than 10% in the range of 3-20 meters.

5) The speed of the car: In the end, our project will be used in a traveling car. However, we have a speed limit for the car due to our camera's FPS limit. Taking our FPS limit of 18, we calculated and determined our car speed limit $32.4km/h$. Our objective is to keep the displacement of objects between two consecutive frames below $0.5m$. We used Equation 1 for calculation:

$$v_{max} \leq \frac{0.5}{18} \times \frac{3600}{1000} = 32.4km/h. \tag{1}$$

6) GUI and User Control: A 2D map is expected to be created to show the distances of detected objects with 2D projection. A user interface shows the detected objects and distances on the camera's visual data and extracted map. The user is able to control the detection confidence threshold.

2.2 Non-Functional Requirements/Constraints

The project does not only consist of the functional requirements but also non-functional constraints. These non-functional requirements are crucial to be considered for the performance of the project, as it is effective in the accuracy of the functioning project. Our project possesses several non-functional requirements as given below. Our project does not have any safety, health, or size-related constraints.

#	Requirement	Justification	Comment
SNFR-1	The budget limit is 1250 dollars for this project	It is proposed by the company	
SNFR-2	Project should be work at daylight conditions	The camera we plan to use is sensitive to light conditions	
SNFR-3	Potholes and speed bumps in Ankara may differ from the ones in public datasets	Publicly available datasets are used. However, the testing environment is the roads in Ankara	In order to overcome this constraint, we extend our dataset with images captured on Ankara's roads.
SNFR-4	100W is needed for our system to work properly	ZED 2 works via USB 3.0 interface whose power is 1.9W. The average power of notebook computers is 100W	

Figure 2: Non-Functional requirements of the system.

1) Budget: In a project, the budget limits the amount of money one can spend to buy necessary equipment. The budget plays an important role in deciding the camera model we will use in our project. 1250$ is the budget we have.

2) The environmental conditions: The project is expected to work at daylight conditions which means it functions at the times the light in the environment is bright enough to detect objects and estimate distances because the camera we plan to use is sensitive to light conditions. The lack of bright light could decrease the accuracy of object detection and distance estimation, which is undesired for the purpose of our project. Moreover, the project is expected to work outside.

3) Geographical constraint: Our project includes an object detection task in which publicly available datasets are used. However, our testing environment is roads in Ankara, Turkey. In terms of the shape, frequency of occurring, and appearance, potholes and speed bumps in Ankara may differ from the ones in public datasets. This may be a constraint for our task but we plan to enrich our data with local photographs in case of this problem.

4) Power: The camera and personal computers are the devices consuming power in our project. Our camera, ZED 2 works via a USB 3.0 interface whose power is 1.9 W. The average power consumption of notebook computers is 100 W. Therefore, it may be concluded that we need approximately 100 W for our system to work properly.

3 Big Picture

The big picture of the project can be seen in Figure 3. ZED 2 stereo camera is used to take visual data from the road. The camera is directly connected to a personal computer via a USB 3.0 cable. Real-time video is transferred from camera to personal computer via USB 3.0. Video frames are separately processed on a personal computer, used as the main processor unit. Both object detection and stereo vision algorithms work on a personal computer. For object detection, YOLOv5 is used and training is done on Google Colab GPU (Tesla K80). Training images for object detection are collected from public datasets and our videos. They are all labeled with Roboflow software. Real-time object detection is done on a personal computer GPU. The stereo vision algorithm works on a personal computer as well. The personal computer has Intel Core i7 central processor and GTX 1650 Ti NVIDIA graphical card. First, a disparity map of left and right images is extracted. Then distance estimation algorithm, which is selected as ZED 2's built-in algorithm after research and comparison with the baselines in the literature, calculates the disparity map and corresponding distance map. The results of the stereo vision algorithm and object detection are combined on the real-time video and projected onto a 2D map and shown to the user. The user is able to control and see the program on the personal computer. On GUI, the user can see the left camera's output with marking bounding boxes and estimated distances of the target objects. Also, there is a 2D map showing the detected object's locations next to the camera output. The user is able to control the detection threshold and region of interest for the detection. Pytorch library is used for training and inference, and OpenCV framework is used to create the 2D map and user interface. The main power source of the project is the battery of the personal computer. The battery provides 100 Watt to the computer. The personal computer also provides 1.9 Watt to the ZED 2 stereo camera through the USB cable.

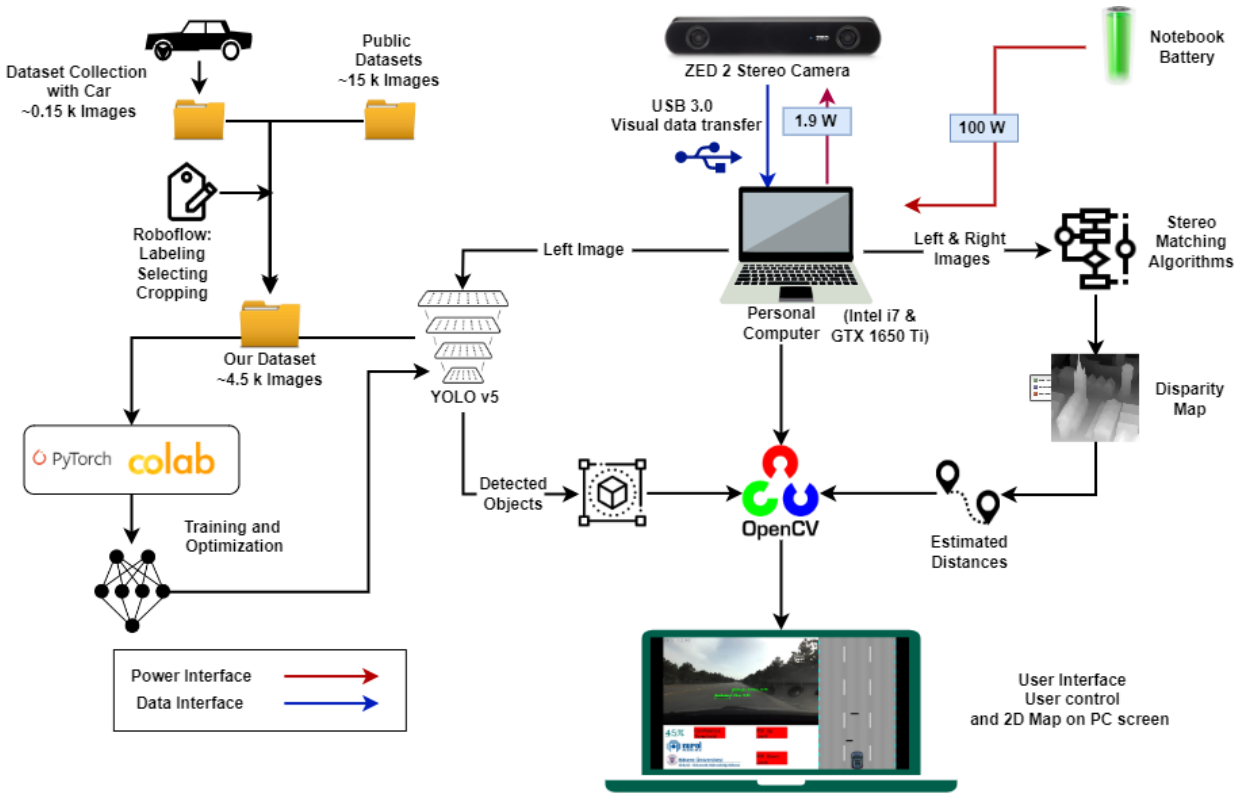

Figure 3: The Big Picture of the project

4 Methods and Implementation Details

4.1 Work Breakdown Structure and Project Plan

Three major tasks that need to be completed for the aim of the project are object detection, distance estimation, and real-time integration.

4.1.1 WBS with Responsible Team Members

Our work packages and the assigned group members for each work package are shown in Figure 4.

For the object detection task, we have divided into two teams for subtasks dataset collection, and implementation of algorithms. The search for available online datasets has been performed by Barış, Fatih, Muhammed, and Şafak, where each of them went over different datasets and analyzed their suitability for our use. Muhammed and Şafak have manually annotated the objects we were trying to detect within these datasets, using Roboflow. On the other hand, Barış and Fatih have preprocessed the annotated images so that they are ready for training and all of the label files are

in the right format.

The implementation of deep learning methods for object detection has been done by Emirhan, Yavuz, and Muhammed. Emirhan and Yavuz have performed research on the available methods in the literature. As a result of this research, they have decided on the models that we used. We trained different sizes of YOLOv5 -small, -medium, and -large. While making adjustments, Emirhan, Yavuz, and Muhammed have simultaneously trained separate models and made improvements depending on the results they obtain.

The distance estimation task consists of camera selection and implementation of algorithms. Şafak and Muhammed have performed a market search to find a stereo camera that can satisfy our functional requirements. After the arrival of the ordered stereo camera, Barış, Fatih, and Emirhan have implemented several different distance estimation algorithms. Also, ZED 2 camera's stereo vision algorithm performance has been investigated.

The last main task is to integrate the different packages of the project in real-time to obtain the final product. The combination of object detection and distance estimation has been done by Barış and Yavuz. The task of GUI design is assigned to Şafak, Yavuz, and Muhammed. They tried both OpenCV GUI and Kivy GUI packages and decided on OpenCV as it provides faster run-time.

In the distance estimation major task, the implementation of the camera's distance estimation algorithm has been added as an additional task and it is completed by Yavuz and Fatih. Moreover, in the real-time integration major task, a GUI design task has been added, and creating a 2D map has been transferred under this task. In addition, implementation of other utilities has been added as a subtask. These tasks are assigned to Muhammed and Şafak.

There is no change in the work breakdown structure since the CM3 report.

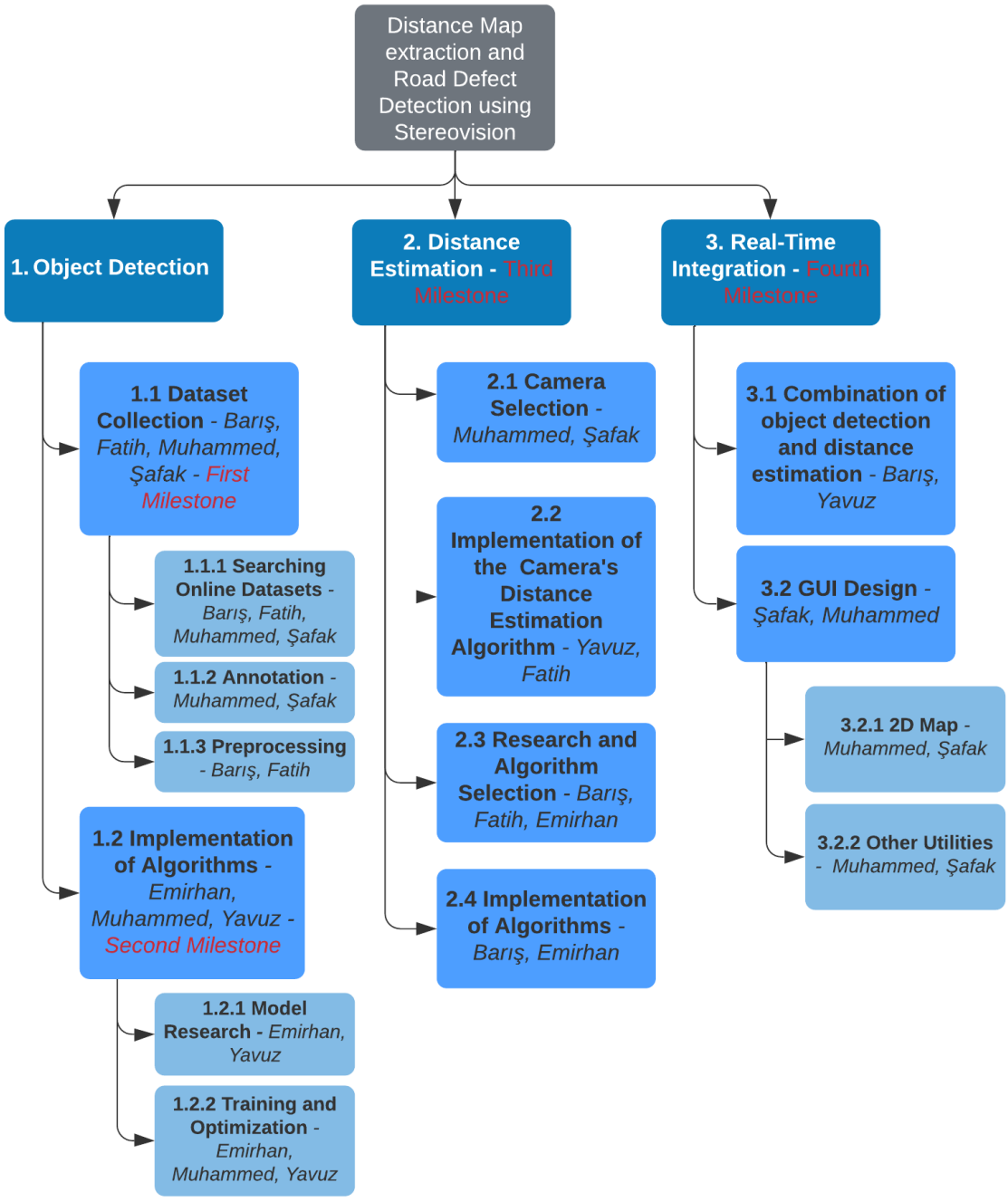

Figure 4: Work breakdown structure for the project (the milestones are achieved when the specified branch is finished with all of the subbranches).

Milestone	Criteria for Success
Completion of Dataset that is Ready to Use	Collection of a large enough dataset. All images must be annotated, labelled in the same format, pre-processed and be ready for training
Object Detection Implementation	Object detection part of the project must be completed. The potholes and the speed bumps must be detected and classified with the required accuracy. The model must be fast enough to satisfy FPS condition.
Distance Estimation Implementation	The algorithms that are extracting the distance map of the scene with the required accuracy condition must be implemented and working. Also, the algorithms must be implemented on ZED 2 stereo camera.
Graphical User Interface	The detected objects must be shown to the user with their estimated distances in 2D map that satisfies the FPS conditions.

Figure 5: Project milestones.

The four milestones for our project with their descriptions can be seen in Figure 5. The first milestone is dataset collection. In order for this milestone to be achieved, we need a large dataset in which every image is annotated, labeled in the same format, preprocessed, and ready for training. We collected 4500 total images 3700 of them are potholes and the remaining 800 images are speed bumps. The second milestone is completing the object detection model. For this milestone to be achieved, the resulting model must be able to detect the potholes and speed bumps with the required accuracy mentioned in the functional requirements part. Moreover, the architecture of the model and the total number of parameters must be adjusted so that the detection and classification times for the images satisfy the 18 FPS requirement. We have completed the model and trained it, so we have fulfilled the second milestone requirement. The third milestone is completing the distance estimation task. The distance estimation has been completed by utilizing the stereo camera. As a criterion for success, the stereo vision distance estimation method that we implement must satisfy the error conditions mentioned in the functional requirements. The last milestone is the completion of the GUI. In order for this milestone to be achieved, firstly, we have combined the distance estimation and object detection results to create a 2D map. The resulting 2D map that shows the detected objects with their respective distances must be displayed to the user with 18 FPS. The last milestone is achieved by completing GUI.

Figure 6: The project timeline.

The Gannt chart for our WBS with responsible team members can be seen in Figure 6. Implementation of distance estimation was shifted to the beginning of the second semester. Currently, it has been implemented. We had planned to complete the object detection task, which is completed, by the end of the first semester, and report our results in the second committee meeting. We started the online dataset search and literature survey for the object detection model on week three of October. These two tasks are currently completed. We have annotated and preprocessed the images in the online datasets we found. As our dataset gradually got larger, we coded the models that we used. Once our dataset have became large enough, we started training multiple models simultaneously. By December 13, we completed our dataset and in the remaining time until the end of the semester, we have done further improvements in our object detection model. Currently, the object detection task is completed and it satisfies all of the requirements.

For the distance estimation part, we started the market research for the stereo camera at the beginning of the first semester. After our research and the adjustments to our allocated budget, we ordered the ZED 2 stereo camera. We received the camera at the beginning of this semester. Firstly, we have implemented the built-in distance estimation algorithm of ZED 2 stereo camera. Also, we have implemented various stereo vision algorithms. Based on the results, we decided on the algorithm we use for distance estimation which is ZED 2 stereo camera's built-in algorithm. Afterward, we combined the distance estimation and object detection results, and currently, we are able to display these results in a real-time video as expected. Starting from CM3, we have focused on the implementation of GUI and completed this task successfully by showing the obstacles in real-time on a 2D map with a GUI that allows the user to change the detection threshold and confidence interval utilized in the machine learning procedure.

4.1.2 Object Detection

In our project, it is aimed that potholes and speed bumps are detected and classified with the accuracy mentioned in Section 2. The object detection part includes two sections which are dataset collection and implementation of algorithms.

4.1.2.1 Dataset Collection: For the part of object detection, first, we found available online datasets for our use. We looked for datasets that consist of road images that contain potholes and/or speed bumps. We went over each image in these datasets to complete the missing annotations and filtered out the images that are not suitable for training. In addition, we preprocessed these images and performed data augmentation (cropping, masking, etc.) to improve performance. Lastly, we went over the files that contain the labels so that they are all in the proper format. Furthermore, we have taken pictures of potholes and speed bumps and labeled them using Roboflow.

4.1.2.2 Implementation of Algorithms: For the implementation of algorithms, the first thing we have done is to make model research in the available literature. We looked for the state-of-the-art deep learning models that are used in object detection tasks. The advantages and disadvantages of these models are evaluated. Then, we implemented these deep learning models with the right adjustments. We implemented and trained YOLOv5 with different sizes and different parameters to compare results.

4.1.3 Distance Estimation

Distance estimation with the help of a camera consists of two major parts which are camera selection and implementation of algorithms.

4.1.3.1 Camera Selection: Considering the budget our project has, the list of possible cameras that could be utilized for our project was prepared. The project requirements are taken into account in order to decide the properties (baseline, range, working conditions, etc.) of the stereo camera. Finally, we chose Zed 2 stereo camera and we are currently utilizing it.

4.1.3.2 Implementation of Camera's Distance Estimation Algorithm: After receiving the camera, we set up the camera and installed the relative packages. We have implemented the built-in distance estimation algorithm of ZED 2 camera.

4.1.3.3 Research and Algorithm Selection: We have searched different algorithms for stereo vision. We have decided to try Block Matching (BM) and Semi Global Block Matching (SGBM) algorithms.

4.1.3.4 Implementation of Algorithms: We tried the Block Matching (BM) and Semi Global Block Matching (SGBM) algorithms for stereo vision distance estimation. Also, we checked the performance of the ZED 2 camera's stereo vision algorithm. At the end, we have decided on using Zed 2 stereo's distance estimation algorithm as the final algorithm after comparing the results.

4.1.4 Real-Time Integration

The real-time integration step includes two major components the combination of object detection and distance estimation, creating 2D Map.

4.1.4.1 Combination of Object Detection and Distance Estimation: We combined the distance map and the pixel coordinates of the detected objects in order to estimate the distance of potholes and speed bumps.

4.1.4.2 GUI: We have implemented a GUI using OpenCV and Kivy GUI. Since OpenCV Gui gives better results for FPS, we chose to use OpenCV Gui in our project for this user interface. **4.1.4.2.1 2D Map:** 2D Map that illustrates the objects according to their distances from the vehicle is created. It is provided in Fig. 7.

4.1.4.2.2 Other Utilities: The video indicating the potholes and speed bumps with corresponding distances is displayed. Moreover, the detection threshold is designed to be controlled by the user.

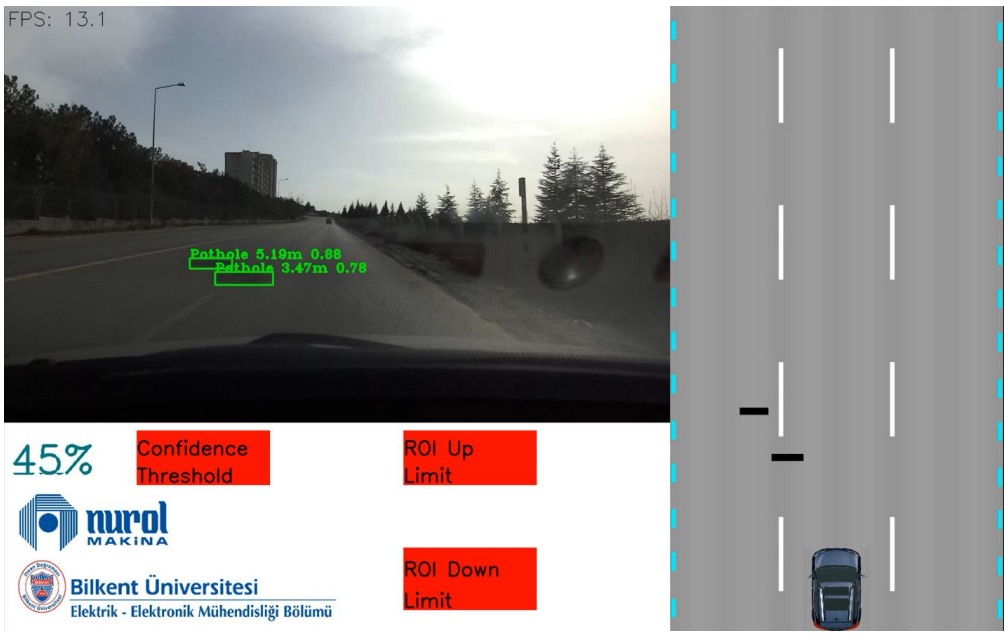

Figure 7: Final GUI.

4.2 Methods and Progress

The project's big picture is given in Figure 3. Our personal computer is used as a processing unit. Both object detection and distance estimation algorithms are implemented on a personal computer. The stereo camera (ZED 2) is connected to the computer. The visual data obtained from the left camera is processed to detect the target objects. If an object is found in the frame, the distance between the object and the camera is calculated using both left and right image frames with stereo vision algorithms. After that, the locations of detected objects is translated onto the 2D surface and the 2D map is updated in real-time. The computer we used has a GeForce GTX 1650 Ti NVIDIA graphical card with 4 GB memory and we run the object detection algorithm on this GPU. Also, the computer has an Intel Core i7 processor used for data acquisition, display, and running our code.

4.2.1 Distance Estimation

4.2.1.1 Camera Selection

For stereoscopic distance measurement, at least two cameras are required. The distance mapping requirements are 20 m range with less than 10% error and less than 1% error in 3 m range, which is directly related to the distance between the cameras, called baseline. Also, camera resolution is important for object detection accuracy. The camera is going to be used outdoors. The budget constraint is another requirement that is considered. To address the requirements, three different camera setups become prominent. They are (i) a dual camera, (ii) two separate cameras. At the beginning, the team had a budget of 1000$ Turkish Liras in the beginning. The candidate dual cameras that the budget was adequate for were VmodCAM - Stereo Camera Module [11], Astra [12], ELP960P2CAM-V90 [13], Intel RealSense D415 [14], HBV-1780-2 [15]. However, none of these dual-cameras has an effective range of 20m distance estimation because the distance between the cameras is short. Therefore, we focused on forming the setup by combining two cameras. Two cameras having master-slave operation property are required for synchronization. Seeing that the distance between cameras is up to our decision, we considered the range of the depth estimation according to the specifications. However, synchronization and stabilization of two cameras take a lot of time. We were going to implement this setup but the company does not want the team to focus on more mechanical works. Instead, they desired our focus to be mainly on algorithms and the improvement of the results. Therefore, they deter the team from two camera setups. The company increased the budget to 1250$ so that we can afford dual cameras. After a search, the ZED 2 stereo camera was selected. The company ordered the camera at a cost of 10000 Turkish Liras from a Turkish company in order for the arrival of the camera not to take much time. The camera has an effective range of up to 20 m and it works with less than 10% error at that range. In video mode, it has 3840x1080 output resolution and works at 30/15 fps. It can be used for both indoor and outdoor purposes. Thus, it satisfies all of the functional requirements [16].

Due to the internal buying processes of the company, the camera has been ordered in late November. However, the only local supplier of ZED 2 cameras stated that the camera will be available in the first days of January. This has forced us to make some adjustments in our plans. They are explained in Section 4.1.

4.2.1.2 Implementation of Camera's Distance Estimation Algorithms

Our visual data is obtained with ZED 2 stereo camera. The connection between the camera and the processor unit, our personal computers, is handled via a USB 3.0 interface. The power of the

camera is also obtained from the computer's power supply through this USB interface. Moreover, the product provides a package that enables us to obtain left and right rectified and unrectified images. We used the left camera's data for the object detection algorithm. For distance extraction, we used both left and right camera data. The technical parameters of ZED 2 camera such as focal length and baseline distance between two cameras are given precisely in its datasheet.

The ZED 2 camera has its own built-in distance estimation algorithm. We implemented and tested this algorithm with multiple image pairs. Afterward, we compared the results with other stereo vision distance estimation algorithms.

4.2.1.3 Research and Algorithm Selection

In the project, the distance between the camera and the object is estimated by using stereo vision algorithms. In this part, the working principle of stereo vision and the algorithm which is used for the distance calculation is explained in detail.

Conversion from 3D to 2D may not work properly by using the information coming from just one image. In this approach, there will likely be an "ambiguity" that causes confusion when perceiving the depth. An example of this ambiguity is shown in Figure 8.

Figure 8: Distance ambiguity example [1].

One could think that a woman holds the Pisa tower at first sight by just looking at the picture above. However, it is an obvious illusion because of the depth projection error in 3D to 2D conversion. This illusion can be fixed by adding new view angles to this picture. At this point, the knowledge of geometry about multiple views comes into play as a solution.

The geometrical explanation for 2-camera systems is called as Epipolar Geometry and its schematic is shown below.

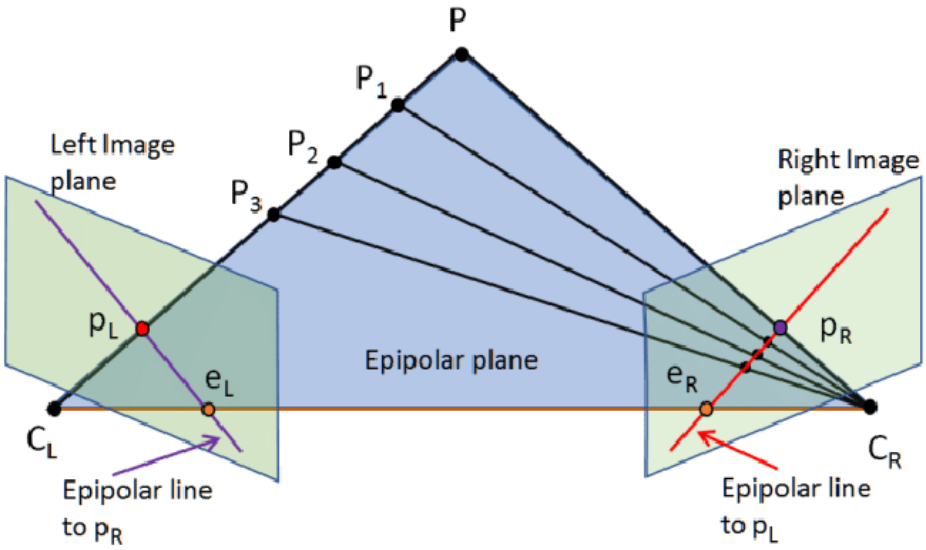

Figure 9: Epipolar geometry schematic [1].

As can be seen from Figure 9, there are two cameras that are positioned in parallel and their observation point is P which is a 3D point and its projection points to each image plane are pl and px respectively. The line between the cameras is called a baseline and the plane consists of camera centers and P is called an epipolar plane. The intersection points of baseline and image planes are called epipoles which are shown in the figure as e_L and e_R. The lines which are the intersection of image planes and the epipolar plane are called epipolar lines and they intersect the baseline at epipoles [1].

The 3D location of P is not known exactly in real-life but its projection to one of the image planes and some information about the cameras such as their locations, matrices, and orientations can be determined. Epipolar plane and epipolar lines can be defined by using this information.

The geometry behind the estimation of 3D model by using two cameras is explained briefly and some technical terms are defined. After that point, a stereo matching algorithm can be analyzed.

Stereo matching is simply based on finding the matched pixels of the selected two images and using their coordinates to estimate the 3D model. People also use this method to perceive depth. The baseline can be considered as the line between two eyes in this case. In the algorithm, two images that show the same scene from different perspectives are used. The aim is to estimate the depth for each pixel. The position of the cameras with respect to each other is important in this experiment. If it is fixed, it is enough to calibrate the cameras just once. Otherwise, more calibration may be needed. In the system, two cameras obtain the images at the same time and this could cause some matching problems because of the differences in the images. It is very important to find the part of the image that matches the other image. This problem is called a correspondence problem and it is considered a key point of stereo computation. The geometry behind this algorithm can be explained

as follows.

If two lines are sent to the 3D point from two camera centers, there will be a triangle as expected. A simple figure for that system is shown in the figure below.

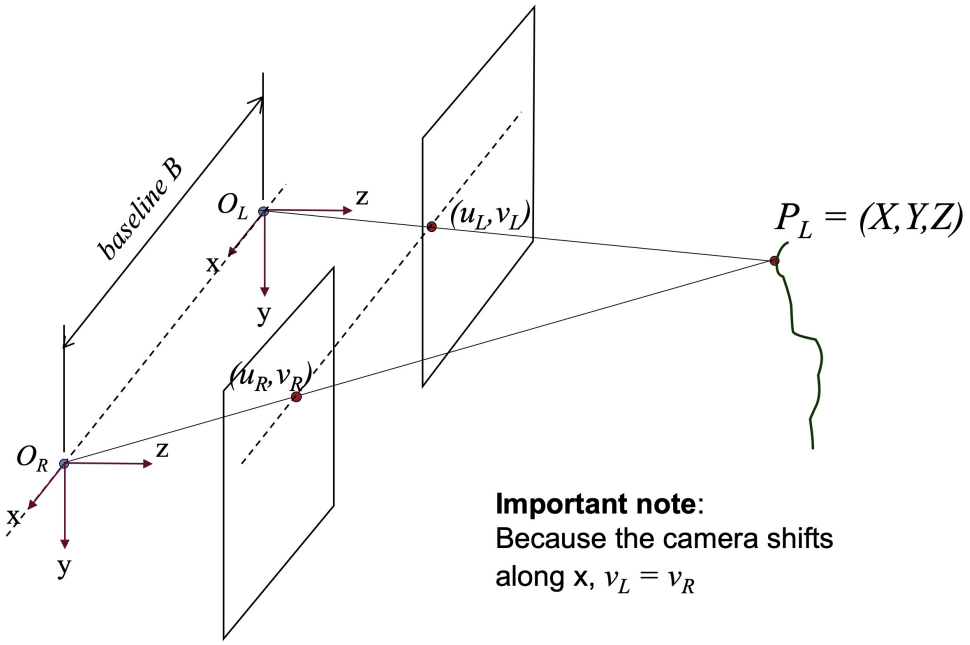

Figure 10: Stereo vision schematic [1].

In this system, there are two cameras that have parallel optic axes. Let's consider just x and z axes. The schematic is shown in the following figure.

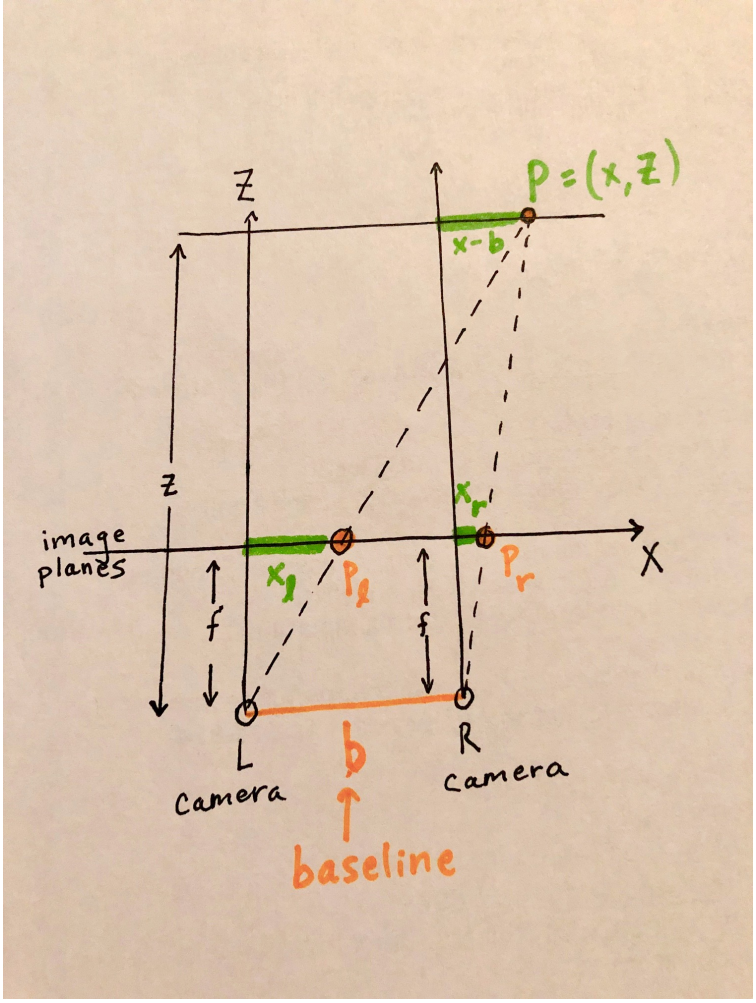

Figure 11: Stereo vision setup

In the figure, y-axis is perpendicular to that plane and f represents the focal length. The projection of P is p_l and p_r for left and right images respectively. There can be defined two similar triangles pairs. One of them is $(x, z) - (0, z) - (0, 0)$ and $(0, 0) - (x_l, f) - (0, f)$, and we can write,

$$\frac{x_l}{x} = \frac{f}{z}. \tag{2}$$

Another pair can be defined as $(b, 0) - (b, z) - (x, z)$ and $(b, f) - (b + x_r, f) - (b, 0)$. We can write,

$$\frac{x_r}{x - b} = \frac{f}{z}. \tag{3}$$

By using these equations, we can write the depth as,

$$z = f \times \frac{b}{x_l - x_r}, \tag{4}$$

where $x_l - x_r$ is defined as the disparity between the images. Therefore, it is understood that the depth depends on focal length, baseline, and the disparity of the images. Moreover, it is inversely proportional to the disparity. This conclusion makes sense because the distant object's movement is very small in the right and left images whereas the closer object moves more in the images. In the next steps, it is assumed that f and b are known.

In order to solve the correspondence problem which is described above some techniques are developed.

The classical approach is the Block Matching (BM) algorithm in which the matching problem is considered an optimization problem. The process starts with an initial guess and gradient descent or Newton's method is used to find the best solution. If the rectification process, which is making two images parallel, is applied to the images, all searching lines become epipolar lines and the epipolar lines converge at infinity. The images can be considered as matrices and the similarity between the matrices is found by calculating their inner products. The most common distance measures to calculate the similarity are the sum of squared difference (SSD) and the sum of absolute differences (SAD) [2].

The match and the disparity for each pixel are found by using a 1D search line. In this case, it is assumed that the images are rectified so that just a horizontal line is enough for searching. An example of this searching algorithm is shown below [2].

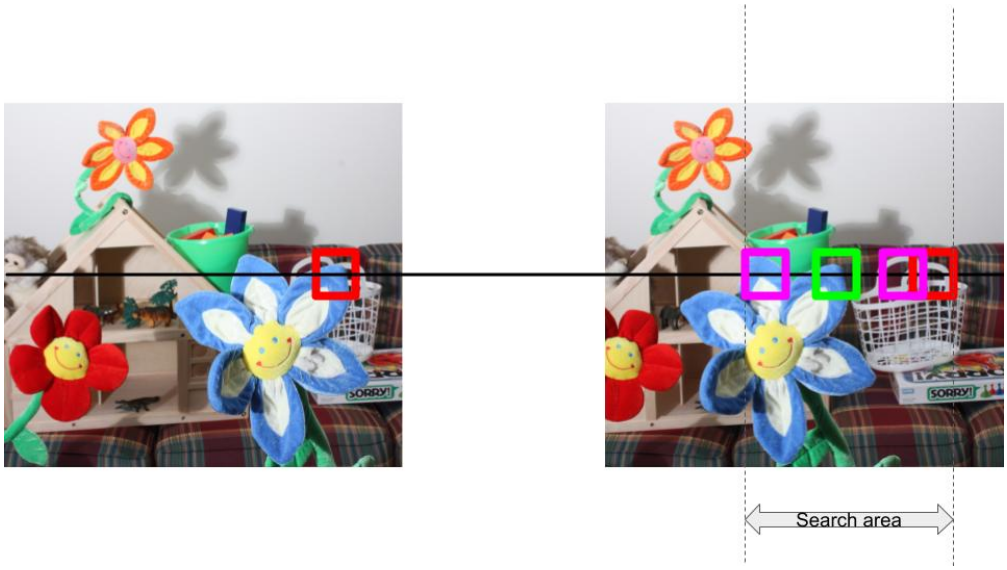

Figure 12: Searching algorithm example [2]

In this example, our aim is to find matching pair of the red patch in the left image. In order to accomplish this, this pair is searched on a line in the right image and the location of the initial patch is the same as the location of the red patch in the left image. The shift from the red patch to the green patch in the right image is the disparity for the center of the patch in the left image.

The disparity map is created by using the all disparity values for each pixel. An approach is to store the similarity values for all disparities of each pixel. This is called cost volume. For instance, $cost(x, y, d)$ represents the similarity value between the patch located at (x, y) in the left image and the patch which has a disparity d from that patch in the right image [2].

In this algorithm, smoothing could be a problem. The patches next to each other can have very different disparities because they are not smooth enough. In order to solve this issue, we can add a regularization term that penalizes jumps in a disparity between adjacent pixels, with a cost function in the form

$$E(\boldsymbol{d}) = \sum_{p} D(p, d_p) + \sum_{p,q \in \mathcal{N}} R(p, d_p, q, d_q) \tag{5}$$

where $D(p, d_p)$ is the pixel-wise dissimilarity cost at pixel p with disparity d_p, and $R(p, d_p, q, d_q)$ is the regularisation cost between pixels p and q with disparities d_p and d_q respectively, for all pairs of neighbouring pixels $\mathcal{N}$. For BM, only the left term in the cost function is optimized. The cost function with the additional regularisation term that penalizes jumps can be enforced with dynamical programming (e.g. Viterbi algorithm) on a per-scanline basis. However, in this case, the adjacent scanlines are optimized independently and streaking artifacts can emerge [17]. Hence, smoothing stays as an issue.

As a solution, Semi-Global Matching (SGM) idea can be used. The idea behind SGM is to perform line optimization along multiple directions and compute an aggregated cost by summing the costs to reach a pixel from each direction [18]. The number of directions can be increased up to 16. The cost is composed by a matching term $D(p, d)$ and a binary regularisation term $R(d_p, d_q)$. In SGM, mutual information is used as the dissimilarity measure for $D(p, d)$, and the regularization term has the form

$$R(d_p, d_q) = \begin{cases} 0 & d_p = d_q \\ P_1 & |d_p - d_q| = 1 \\ P_2 & |d_p - d_q| > 1 \end{cases} \tag{6}$$

where P_1 and P_2 are two constant parameters, with $P_1 < P_2$. The three-way comparison allows assigning a smaller penalty for unitary changes in disparity, thus allowing smooth transitions corresponding e.g. to slanted surfaces, and penalizing larger jumps while preserving discontinuities due to the constant penalty term.

Semi-Global Block Matching (SGBM) is a variation of SGM that aims higher speed at the expense of accuracy [19]. In SGBM, only 5 directions are considered instead of 8. Moreover, mutual information cost is not implemented and a simpler Birchfield-Tomasi sub-pixel metric [20] is used.

(a) Left image from the camera

(b) Right image from the camera

(c) Disparity map using BM

(d) Disparity map using SGBM

Figure 13: Example of disparity maps extracted using BM and SGBM algorithms.

After CM2, we started the implementation of the distance estimation algorithms upon the arrival of ZED 2 camera. After searching different algorithms in the literature, we decided to test Block Matching (BM) and Semi Global Block Matching (SGBM) algorithms for the distance estimation task. Then, using the visual data from ZED 2 camera, we implemented these algorithms. In addition, we tried the built-in distance estimation algorithm in ZED 2 camera and compared the results.

4.2.1.4 Implementation of Algorithms

In order to determine the distance algorithm that we used in real-time distance estimation, we implemented and tested BM, SGBM, and the built-in ZED 2 algorithms on Python. We compared the algorithms in terms of accuracy and speed. In order to determine accuracy, we calculated the distances of the three objects using the algorithms and compared the results with our measurements. These evaluations were performed in indoor conditions. Figure 13 shows an example of the disparity maps extracted from a pair of images taken from the camera.

In order to compare the accuracy of the three algorithms, we estimated the distances of objects at measured distances of 0 to 15 meters away from the camera. 10 measurements in the laboratory and 12 measurements outside have been done for each algorithm. Up to 10 meters, the average

error for BM algorithm is 19% and 11% for SGBM algorithm, and 5% for the built-in algorithm of ZED 2 camera. The best three sample results are shown in Table 1. According to our calculations, even though all algorithms perform adequately at close distances, as the distance of the object increases, the accuracy of BM and SGBM gradually decreases while ZED 2's built-in algorithm shows the best performance and stays within the acceptable error range. Moreover, we observed that BM and SGBM algorithms have difficulty in matching the objects as the distance from the camera increases. Especially with BM algorithm, most of the pixels could not be matched and a lot of unwanted artifacts were observed at object boundaries.

In order to improve the stereo matching performance of the BM and SGBM algorithms, we tried applying some preprocessing steps before matching the two images. Specifically, we applied Canny edge detection [21] on our images so that the object boundaries can be more clearly defined. This edge detection method includes a Gaussian filter to smooth the image, followed by the horizontal, vertical and diagonal Sobel filters that give the edge gradients of the image. Finally, the resulting values go through thresholding and the edges are detected. An example of the edge-detected version of a stereo road image pair is shown in Figure 14.

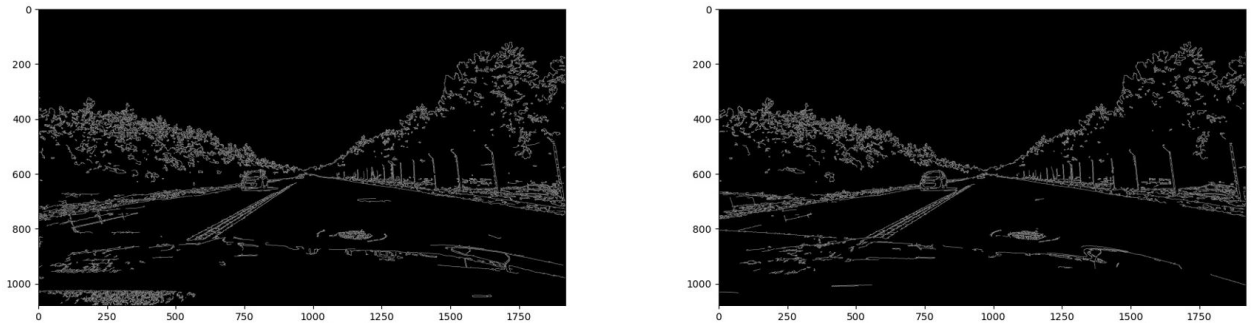

Figure 14: Edge detection for a road image pair.

The edge detection step failed to improve the performance of the distance estimation algorithms and the main cause can be observed in Figure 14. As seen in the figure, some objects could not be detected at all after the edge detection. Moreover, there is a discrepancy between the edges detected in the two images. Therefore, even if we reduce the threshold to detect some objects that were otherwise missed, the difference between the edges detected in the two images reduces the overall matching performance. As a result, we were not able to improve the performance of the algorithms by detecting the edges beforehand.

Table 2 shows the amount of time it took for different algorithms to extract the disparity map of a single frame. It can be seen that ZED 2 algorithm is the fastest while SGBM is the slowest by a large margin. Considering the extra time that the object detection task takes for each frame, only ZED 2 algorithm can satisfy our 18 FPS requirement. Therefore, for our real-time task, we decided

Measured Distance (m)	BM		SGBM		ZED 2	
	Estimated Distance (m)	Error (%)	Estimated Distance (m)	Error (%)	Estimated Distance (m)	Error (%)
1.35	1.30	3.7	1.30	3.7	1.36	**0.7**
2.35	2.22	5.5	2.30	2.1	2.34	**0.4**
5.35	6.03	12.7	5.52	3.2	5.39	**0.7**

Table 1: Comparison of best samples for the accuracy of distance estimation algorithms at different distances.

	BM	**SGBM**	**ZED 2**
Time Elapsed (s)	0.076	2.192	**0.023**

Table 2: Comparison of the speed of disparity map extraction of different algorithms for a 1920×1080 image.

to use the ZED 2 algorithm which also gives the best accuracy.

4.2.2 Object Detection

4.2.2.1 Dataset Collection

For training purpose, publicly available datasets are searched. During our search, we focused on either the intersection or one of the road datasets, pothole datasets, and speed bump datasets. In total, more than fifteen datasets are found, but some of them were eliminated due to improper distance of the camera while capturing the photograph. After the evaluation of all datasets, we found that none of them can be readily used for our project.

First, there is an inconsistency between the datasets' angles of view. Furthermore, inconsistency appears for the images from even the same dataset. Secondly, the annotations of available datasets are highly problematic. Most of them do not have annotated images. The remaining already annotated ones have only pothole annotations, only speed bump annotations or pothole and other objects' annotations except speed bumps. Moreover, some of the data is collected with the camera mounted inside the car while others are collected with the camera outside the car or mobile phone cameras. Although we had thought that it would be a problem for us before CM1, we after decided that we can utilize this inconsistency. In our project, ZED 2 camera is placed inside the car. Therefore, our model should learn that the region of a car in the image does not include any target objects.

To resolve these issues, we have looked at and modified all of the images one by one. We have resized images and annotated relevant objects in images. We call these steps "dataset modification". For resizing operation, to keep the aspect ratio of the image we first padded images with zero. For annotations, we used Roboflow annotation tool [22]. It has all the required features for dataset modification. We selected this tool among six candidate tools. Also, resizing operation is important because there is no common image size in our datasets. We used square-sized images for training since it enables us to use mosaic augmentation as proposed in [23].

The datasets used are explained in this section.

1. The KITTI Road/Lane Detection Dataset: The KITTI dataset contains the road images collected with a camera mounted at the top of the car [24].
2. Lisa Traffic Sign Dataset: The dataset is collected for traffic sign detection tasks. It includes video frames captured by a camera inside a car [25].
3. Small Obstacle Detection Dataset: The dataset is collected with the ZED 2 camera and small obstacles not belonging to the road are annotated [26]. However, they do not include potholes and speed bumps.
4. Pothole and Plain Road Images: This is a Kaggle dataset containing raw pothole and road images. No annotations are done and images are obtained from Google search.
5. Road Damage Dataset: This is a Kaggle dataset containing road images and crack & pothole annotations.
6. Speed Hump/Bump Dataset: The dataset includes road images with speed bumps [27]. It is collected for the speed bump detection and distance extraction task.
7. Cracks and Potholes in Road Images: The dataset includes road images with defects [28]. The segmentation maps of defects are given but the coordinates of bounding boxes, and annotations, are not readily given.
8. Pothole, Vehicle and Speed Breaker Detection Dataset: This is a Kaggle dataset that includes road images with annotations of three classes, potholes, vehicles and speed bumps. Photographs are captured with a camera inside a car, hence, the car regions should be cropped.
9. Road Pothole Images for Pothole Detection Dataset: This is a Kaggle dataset including road images with pothole annotations. The region of cars should be cropped before use.
10. Pothole Detection Dataset: This is a Kaggle dataset where images are scraped from Google search.
11. A Fully Annotated Image Dataset for Pothole Detection: The dataset includes road images with defects. Since images are captured from a closer distance than our purpose, we use some of them.

12. Pothole Image Dataset: The dataset consists of Google search images with potholes. The angle of view in some images is different from our use. We discard those images.
13. Pothole Detection Dataset: This is a Kaggle dataset with annotated potholes [29].

Overall, these datasets comprise nearly $15,000$ images. We have applied dataset modification steps and we have obtained nearly $4,500$ images ready to use for training. We have split our set into train, validation, and test sets in 75%, 15%, and 10% respectively.

In addition to public datasets, we have collected road videos in Ankara. This is a vital improvement since none of the public datasets were collected in Turkey. Road types and consecutively potholes and speed bump types change from country to country. Since our product is used in Turkey, more images from the country's roads will increase the accuracy. We have obtained nearly 1500 frames from recorded videos. We have applied the same dataset modification procedure and ended up with nearly 150 new images.

4.2.2.2 Model Research

Object detection is a broad term used to describe a group of related computer vision tasks related to the identification of objects in image data. Predicting the class of one object in an image is known as image classification. Identifying the location of one or more objects in a picture and creating a bounding box around their extent is known as object localization. Object detection combines these two tasks, locating and classifying one or more objects in an image. The ability to locate the object inside an image defines the performance of the algorithm used for detection. There are mainly two approaches to object detection:

1. Two-shot detection
2. Single-shot detection

In two-shot detection methods, there are two stages involved. The first one is the region proposal and the second one is the classification of those regions and refinement of the location prediction. One of the most known examples of two-shot detection is RCNN which runs a selective search on the image and selects the first 2000 region proposals from the results for classification. Although this algorithm is better than an exhaustive search it still takes around 47 seconds for each test image and it cannot be implemented in real-time [30]. There is an alternative method called Fast RCNN in which the input image is fed into the CNN, which generates a convolutions feature map. The convolution operation is done only once per image and a feature map is directly generated from it. Therefore, it works faster than RCNN. Similarly, in Faster RCNN the image is provided as an input to a convolutions network and it predicts the region proposals using a different network which

makes it faster. These are faster and more accurate than R-CNN, but they still fall short of real-time performance [31].

Single-shot detection, on the other hand, skips the region proposal stage and provides final localization and content prediction all at once. Although two-shot detection models perform better in general, single-shot detection is at the sweet spot of performance and speed/resources, making it more ideal for tasks such as recognizing objects in real-time like our project. One of the most popular examples of this is YOLO. Object recognition is reframed as a single regression task with YOLO, that goes straight from image pixels to bounding box coordinates and class probabilities. To predict what things are present and where they are, you only look once (YOLO) at an image. This makes YOLO extremely fast. There is no need for a complicated pipeline because detection is defined as a regression problem. At test time, it runs a simple neural network on a new image to predict defections. Secondly, YOLO sees the full image during training and validation, unlike sliding window and region proposal-based approaches. Thus, it encodes contextual information about classes, as well as their appearances. The number of background mistakes of YOLO is half of the Fast R-CNN [3].

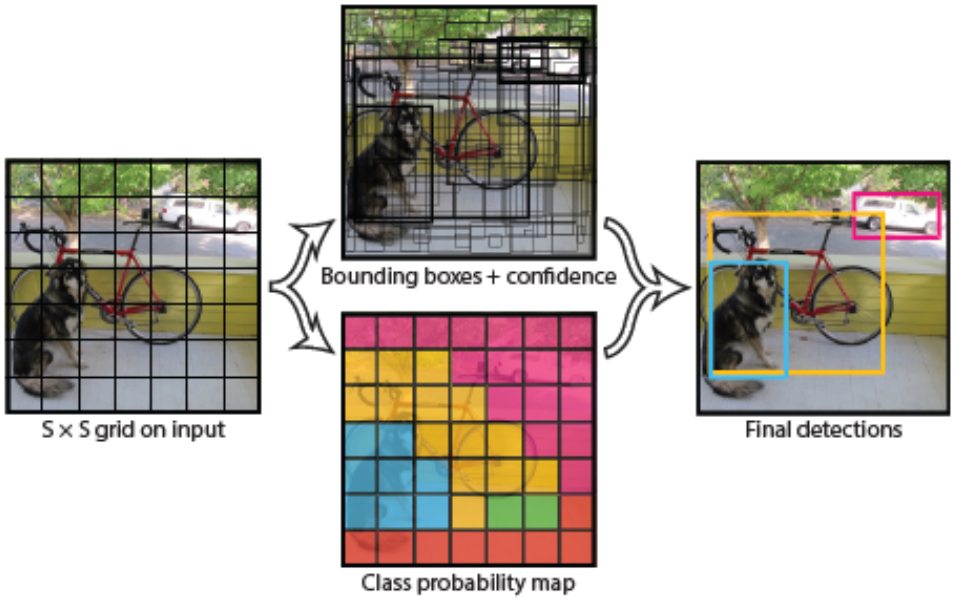

Figure 15: YOLO grids [3].

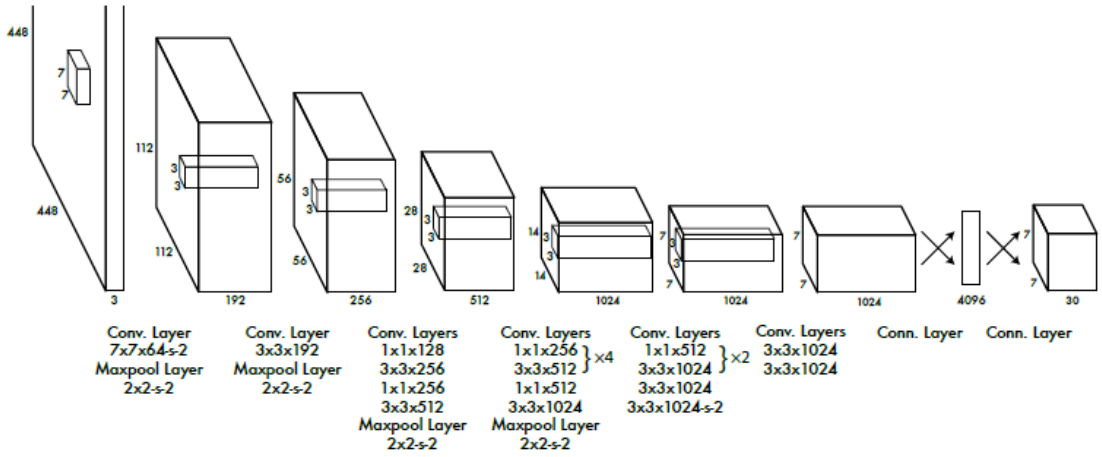

Figure 16: YOLO architecture [3].

In YOLO model, the given image is divided into a S x S grid as in Figure 16. If the object's center falls inside a grid cell, that grid cell is in charge of detecting the object. There are five predictions in each bounding box: x, y, w, h, and confidence. The (x; y) coordinates represent the center of the box with respect to the bounds of the grid cell. The width and height are predicted relative to the whole image. Since YOLO prediction results is in this form, our annotation for the dataset should be in the same format. The network architecture can be seen in Figure 15 . The YOLOv1 architecture can be seen in Figure 16. It has 24 convolution layers followed by 2 fully connected layers. The convolution layers extract features from the image and the fully connected layers predict the class probabilities and bounding boxes. This limits the number of nearby objects that the model can predict. Since each grid cell can only predict two boxes and only one class, YOLOv1 has limitations on bounding box predictions. Small objects that emerge in groups are difficult for the model to handle. These problems are solved and the performance increased with later versions [3]. YOLOv2 version improves the performance by introducing the concept of anchor boxes. Anchor boxes are simply predetermined areas inside a picture that depict the idealized position of the items to be recognized. Moreover, the model is arbitrarily resized throughout the training process to adapt to varied aspect ratios, which is known as multi-scale training [32]. YOLOv3 used 75 convolutional layers instead of fully linked or pooling layers, resulting in a much smaller model. It combined the best of both worlds by combining residual models (from the ResNet model) with feature pyramid networks (FPN) for multiple feature learning with minimal inference times [33]. And finally, the bag of freebies, which are approaches that improve model performance without raising inference cost, and the bag of specials, which are techniques that increase accuracy while increasing computing cost, were in YOLOv4 [23]. YOLOv5, which is similar to version 4 but implemented in PyTorch rather than Darknet, is introduced in 2020. YOLOv5 gives 56.0 mAP with

larger than 50 FPS for COCO dataset [34].

We have used YOLOv5 architecture in our project since it gives better performance with a high FPS criterion. Moreover, YOLOv5 can detect small images that are close to each other with high precision. This feature is quite helpful while detecting small potholes.

There are five different sizes of YOLOv5 architectures, which are nano, small, medium, large, and x-large. They all use the same structure but the depth and number of parameters of models are different. Hence, the number of operations (multiplication and addition) for one frame to pass through the model is different. Consecutively, their training and inference time vary with their sizes. Below, the performance summary of models is provided for an image with a size of 640×640 on COCO validation set.

Model	**mAPval 0.5**	**Inference Time (ms) (CPU)**	**Inference Time (ms) (GPU)**	**Number of parameters (M)**	**Number of FLOPs (B)**
YOLOv5n	46.0	45	0.6	1.9	4.5
YOLOv5s	56.0	98	0.9	7.2	16.5
YOLOv5m	63.9	224	1.7	21.2	49.0
YOLOv5l	67.2	430	2.7	46.5	109.1
YOLOv5x	68.9	766	4.8	86.7	205.7

Table 3: The performance comparison of various YOLOv5 models for 640×640 on COCO validation set [35]. Reported values are taken from [34] and based on their processors. FLOPs stand for floating-point operations per second. The number of parameter is the number of total trainable parameters within a model.

After consideration of the trade-off between the performance and the inference time, we have decided to train three models YOLOv5s with 640 and 960 pixels width and YOLOv5m with 640 pixels width. Also, we have considered the available GPU capacity of our computers for inference and Google Colab for training.

4.2.2.3 Training and Optimization We have trained YOLOv5 models on Google Colab GPU (Tesla K80 with 12 GB memory) using the dataset that we created. All of the models have been trained for 200 epochs and we looked at the precision-recall and mean average precision to make sure that the training process is converged. We have eliminated YOLOv5m since its inference time was long, and it did not satisfy our FPS criterion. Between YOLOv5s models, the model trained

with images whose width is 960 pixels is superior to the model trained with images whose width is 640 pixels. Therefore, we have chosen our main model as YOLOv5s trained with images with the size of 960×960. For training, the batch size is 16, the initial learning rate is 0.01, and the originally proposed learning rate scheduler is used. As an optimizer, Stochastic Gradient Descent with a momentum rate 0.937 is used and the loss function is the original YOLOv5's loss function. Mosaic data augmentation, which simply combines four images into one in different size ratios, is used as proposed in [23]. After the training is completed, we analyzed the results on the validation set. The training statistics can be seen in Figure 17. The precision and recall metrics are calculated using Equations 7 and 8. The mean average precision is calculated by taking the integral of the precision-recall curve using Equation 9 as below:

$$Precision = \frac{TP}{TP + FP}, \tag{7}$$

$$Recall = \frac{TP}{TP + FN}, \tag{8}$$

$$Mean\ average\ precision = \int_0^1 p(r)dr. \tag{9}$$

where $p(r)$ is the curve of precision with varying recall for different confidence thresholds; TP, FP and FN stand for the number of True-Positives, False-Positives and False-Negatives respectively. The precision-recall curve can be seen in Figure 19. This curve shows the trade-off between precision and recall with changing confidence threshold. For detection, we determined our confidence threshold, 0.45, by looking at the precision-recall curve such that it makes both precision and recall high at the same time. The confusion matrix of the YOLOv5 can be seen in Figure 18. As it can be seen in this figure, 73% of potholes are labeled as correctly and 93% of the speed bumps are labeled as correctly. Overall precision is 87%, overall recall is 77%, and mean average precision with IOU:0.5 is 81%. The examples of object detection can be seen in Figure 20.

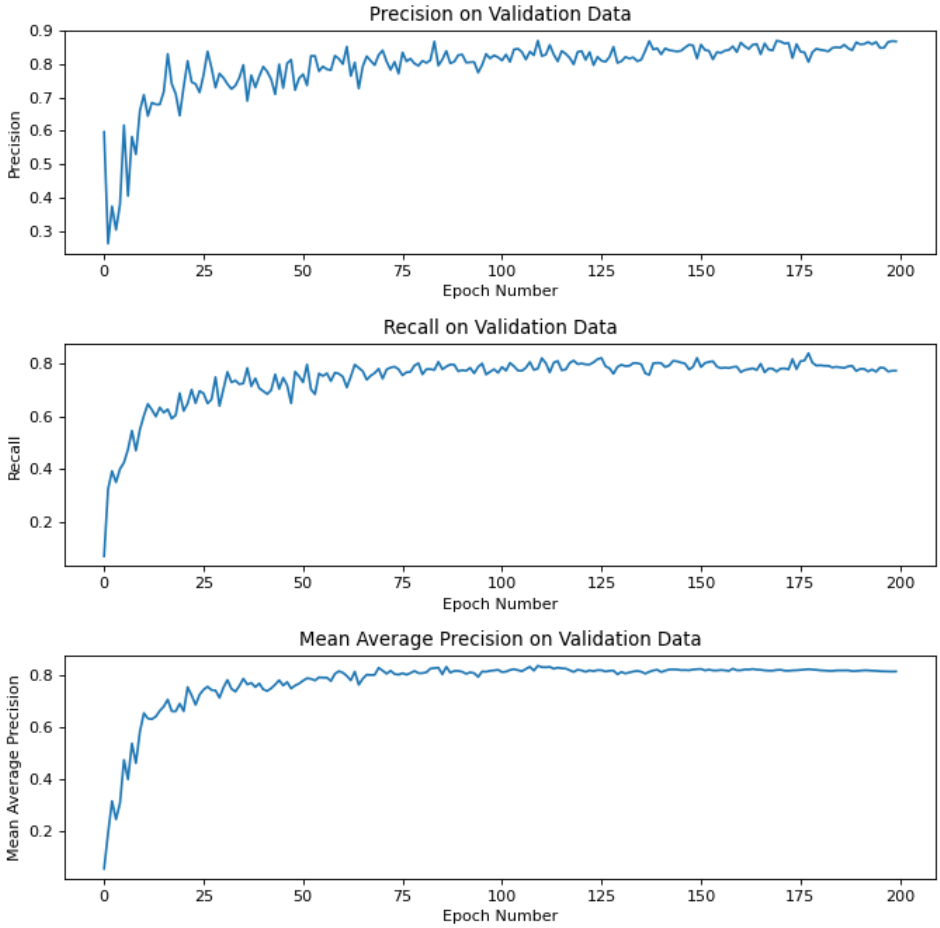

Figure 17: Training statistics.

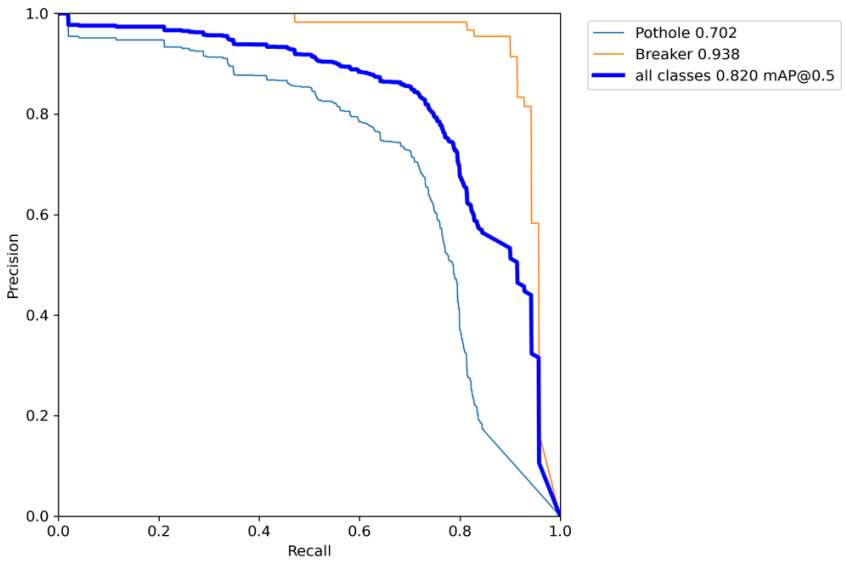

Figure 18: Precision & recall curve.

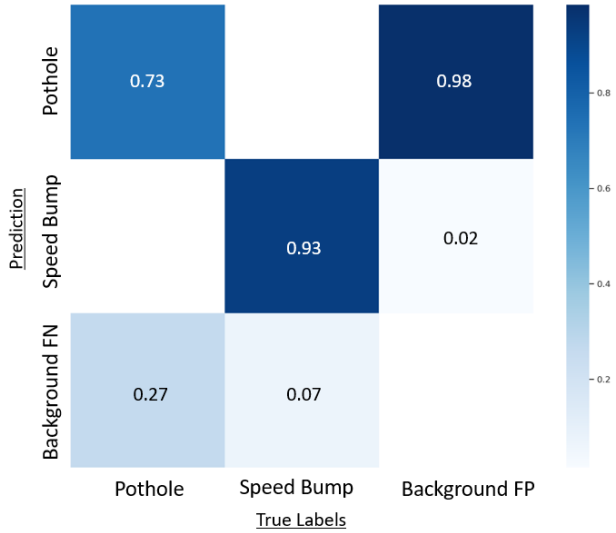

Figure 19: Confusion matrix.

Figure 20: Examples of the object detection.

4.2.3 Real-Time Integration

4.2.3.1 Combination of Object Detection and Distance Estimation

The object detection algorithm is implemented in real-time with the video input coming from the left camera of ZED 2. When an object is detected, we run the distance estimation algorithm and calculate the distance to that object acquired. Acquired distance and object boundaries are shown in figure 21

For a run-time improvement, the object detection algorithm has given only one of two frames when there are no detected objects. Moreover, object detection takes only regions restricted by region of interest boundaries. With these improvements, our algorithm works faster and higher FPS is achieved.

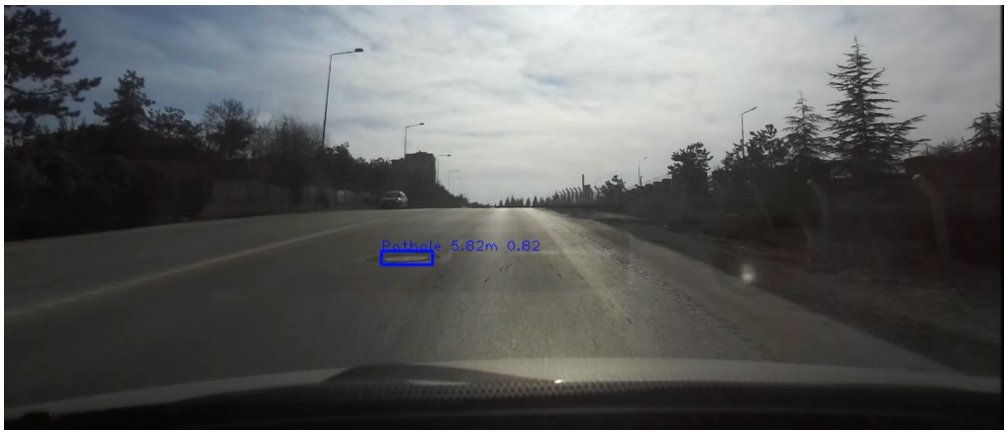

Figure 21: Combination of two modules.

4.2.3.2 Graphical User Interface

We integrated two main modules of our project: object detection and distance estimation. We created a 2D map that shows the positions of detected objects with distances relative to the car from birds-eye view. Objects' vertical positions are directly acquired from the depth estimation algorithm. To find the objects' horizontal position, we did a simple perspective calculation by utilizing the field of view angle of the camera. We also estimated objects' horizontal width and displayed it on 2D Map accordingly. An example of a 2D Map can be seen in Fig. 7.

For Graphical User Interface we used OpenCV interface. Our GUI consists of three main parts: a display that shows detected objects with their distances, 2D Map, and a user control panel. The user control panel allows users to adjust the confidence threshold and region of interest (ROI) for object detection. The upper and lower bounds of ROI can be adjusted as in Fig. 22. The object detection algorithm only takes the image between the upper bound and lower bound. Hence the algorithm works faster and an FPS increase is achieved.

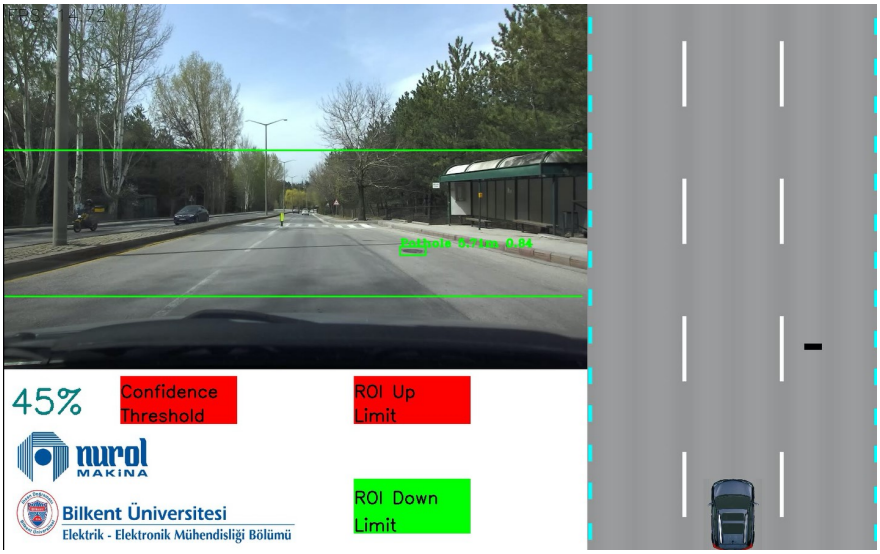

Figure 22: Region of Interest Adjustment.

4.2.4 Progress and Current State

We have added this section to explain our progress clearly. We have provided following table that shows all progress for each work package in a compact form.

Work Packages	Up To CM2	Between CM2 and CM3	Between CM3 and CM4
Camera Selection	Completed all work	Camera is delivered to us	_
Research and Algorithm Selection for Distance Estimation	Research is completed. Possible algorithms are selected	_	_
Implementation of Algorithms	_	Two different alghorithms are implemented and performances are compared	_
Dataset Collection	Dataset collection and labelling are completed	_	_
Model Research	The research is done. Three models with different sizes are trained. Best model is selected	_	_
Training and Optimization	Model is trained untl CM1 and further optimizations are completed up to CM2	_	_
Combination of Object detection and Distance Estimation	_	Distance estimation and Object detection are combined and the demo has been performed	Optimisation is done to increase FPS
Creatiing 2D Map	_		2D Map is created using output of distance estimation and object detection.
Graphical User Interface	_	Initial Research is done. Possible candidate softwares to create GUI are found	Graphical user interface is completed with all promised features.

Figure 23: Progress Table.

4.2.5 Risks and Precautions

The project had some potential risks and problems that would inhibit achieving the goals. The possible risks are analyzed and our solutions are explained.

1. **Insufficient CPU & GPU capacity:** For the first demo, our setup, described at the beginning of 4.2, is sufficient to satisfy our requirements until CM2. However, as we implement the remaining parts of the project including the distance estimation and 2D mapping, the burden on computational units is increase. This resulted in a limitation of the number of operations done in unit time for all work packages. Namely, it decreased our FPS performance.

   To reduce the performance risks, we switched to a better system that has a GTX 1650 Ti NVIDIA graphical card and a better generation processor. With the implementation of 2D Map and GUI, our FPS is decreased further.However, our FPS is still in acceptable range.
2. **Insufficient power supply when the PC is unplugged:** It is a challenge we first realized when we performed the first demo. In the beginning, we used Microsoft Windows 10 as an operating system. When we use our computer for the demo, it works using the stored energy in a battery. For better battery performance, Windows prohibits excess power use when it is unplugged. Although all settings related to power and battery use are done, it still has decreased the performance of all processing units in the computer. This is because Windows acts as a black box to adjust the power supply. This problem mainly affects the object detection part since most of the performance loss happens in GPU. For object detection, we have used a deep learning-based method whose real-time implementation heavily depends on GPU.

   We have proposed several solutions to this problem. First, the problem in Windows OS is searched and software solutions for the problem is tried. We found out that best way to deal with this problem is to run our project on Ubuntu operating system. With this update, the performance problem while unplugged is solved.
3. **Distance Matching Problem:** Distance estimation algorithms are based on disparity matching which works by matching two images coming from two cameras. Since the patterns on the roads are similar, distance estimation algorithms can fail to match disparities and cannot estimate the distances. To solve this problem, we tried to change the properties of the images such as contrast and brightness to get best performance. Moreover, we tried to apply edge detection before feeding the distance estimation algorithm with frames.

5 Results, Discussions, and Future Directions

The results, discussions, and future directions of the project are provided in this section.

5.1 Results

We have successfully implemented our project in real life by running our system while driving through the campus and Beytepe roads. As a result, it is seen that speed bumps and potholes are successfully detected in the range of 20 meters. Our project aims at detecting speed bumps and potholes, classifying them, and finding their distances to our car. For object detection, we looked at the ratio of detected objects and all desired objects on the road based on our visual evaluation. For the detection part, we successfully detected 80% of the potholes and 85% of the speed bumps. Our overall precision is 84%, and recall is 82,4% which is more than 70%, our minimum requirement. The results can be seen in Table 4.

Average Detection Results Based on Our Visual Evaluation

Estimation \ Groundtruth	True	False
True	42	9
False	8	-

Table 4: Object Detection Experimental Results.

For the distance estimation, we have tried 3 different methods and the results for these methods can be seen in Table 5. We used ZED 2 algorithm in our final design which gives the best results among these 3 techniques. With ZED 2, we get 5% errors up to 10 meters, and 12% errors from 10 to 20 meters. These results are very close to our requirements and approved by the company.

Average Estimation Errors (∼12 measurements for each method)			
Distance	**BM (%)**	**SGBM (%)**	**ZED 2 (%)**
Up to 10 m	∼19	∼11	∼5
From 10 to 20 m	Cannot match!	∼24	∼12

Table 5: Distance estimation experimental results.

We have combined our distance estimation and object detection modules. The results of these modules are shown to the user in Figure 24. The user can easily see the distance and type of object on the screen. It means that the requirement of the 2D map is satisfied. Moreover, the user is able to control the detection confidence threshold and region of interest. Our system works with 16 FPS. Considering that object detection, distance estimation, and 2D Map creation work simultaneously in the background, 16 FPS is quite satisfactory for our project and approved by the company.

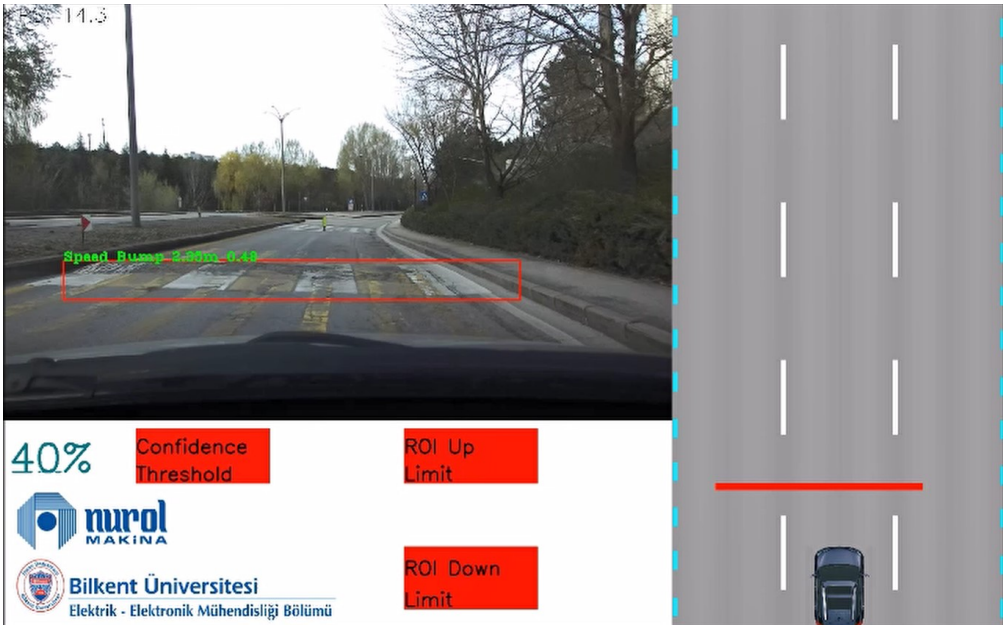

Figure 24: Sample object detection, distance estimation, and 2D map within GUI.

5.2 Discussion and Lessons Learned

The goal of our project was to design and create a product that can detect the potholes and speed bumps, classify them and inform the driver of their relative distances to the car. With the finished product, we managed to achieve our initial goals for the project and almost all of the design requirements were successfully satisfied.

During most of the project phases, especially prior to real-time tests on the field, most of our implementation went as we planned and lead to successful results. For the object detection part, our initial dataset was largely composed of images that we had compiled from other online datasets. We had anticipated that we would need to personally take some photos of city roads to expand our dataset but our initial testing results were even lower than expected and we needed to take more photos, especially for the speed bumps. Once we added enough images to our dataset, the deep learning model could easily meet performance specifications after training.

We decided to use ZED 2 stereo camera in our project after our market research. However, the order of the camera and its delivery took longer than expected and we obtained it in the second semester which was later than we planned. During the process, we learned that the order of this kind of equipment may take longer than expected, especially if there are issues related to the company and the budget. We would suggest others who might be involved in similar projects to plan and order their equipment as early as possible to avoid unexpected delays.

For the distance estimation part, we performed literature research and decided to implement two

algorithms (BM and SGBM) in addition to ZED 2's built-in algorithm and compare the results. Our plan was to see if we could improve the depth estimation performance of the camera with these other algorithms. However, neither of the two performed nearly as well as we hoped and they were far off the performance of the ZED2's built-in algorithm in terms of both speed and accuracy. After completing both the object detection and distance estimation modules, we combined them and provided the results on a graphical user interface. For the GUI, we learned to use two different frameworks and tried both of them to choose the one with better performance.

In the final stage, we started our real-time tests on a car by driving through the city roads. We ran into some unexpected incidents during these tests. For instance, we observed that FPS drops a lot when the testing video was being recorded. Moreover, we saw that the weather condition in cloudy and dark weather can affect our performance significantly because of the light balance of the camera. In better light conditions and when the video is not being recorded, we achieved the high accuracy performance that we had seen in our dataset.

5.3 Future Directions

The project is done in cooperation with Nurol Makina ve Sanayi A.Ş, which is a defense industry company. The outcomes of the project can be used in daily vehicles. Considering that smarter cars reduce accidents and car damages, either our product or its derivatives could be integrated with today's cars [36]. For the defense industry, identifying a terrain is crucial for both single and swarm vehicles. The ideas from our project may play an important role in future innovations in this industry. One possible future direction of our project is to utilize stereo data from more cameras placed at different angles to increase distance estimation accuracy. In our implementation, we have used only two cameras without an angle shift. Furthermore, if there is no constraint unlike in our project, active sensors such as LIDAR could be leveraged for enhanced accuracy. There is also room for increasing the computational power by using better hardware, which may boost the object detection performance by enabling processing frames with higher resolution.

6 Equipment List

The required equipment for our project is ZED 2 camera and computer as explained in Section 3. We used our computers as processor. The purchase of this camera is done by the company, Nurol Makina, and we borrowed the camera. We might have needed external power supply -Tuncmatik Digitech Pro 1000 VA- in case of the problem explained in Section 4.2.5, but we have already solved the problem without using an external supply. Remote GPU, Google Colab Tesla K80, is used to train the network without any charge.

Equipment	Cost	Acquired from
ZED 2 Camera	Borrowed ($1000)	Nurol Makina
MSI GF63 Thin 93CSR PC	-	Ours
Google Colab GPU (Tesla K80)	Free	Google

Table 6: The equipment list.

References

[1] K. Hata and S. Savarese, "Epipolar geometry," 2016, accessed on: Oct. 16, 2021. [Online]. Available: https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf

[2] J. Lambert, "Stereo and disparity," 2018, accessed on: Nov. 18, 2021. [Online]. Available: https://johnwlambert.github.io/stereo

[3] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015, accessed on: Nov. 18, 2021. [Online]. Available: http://arxiv.org/abs/1506.02640

[4] M. Dodds, "Pothole damage costs u.s. drivers $3 billion annually," Feb. 2016, accessed on: Dec. 26, 2021. [Online]. Available: https://info.oregon.aaa.com/pothole-damage-costs-u-s-drivers-3-billion-annually/

[5] Y. Yik, N. Alias, Y. Yusof, and S. Isaak, "A real-time pothole detection based on deep learning approach," *Journal of Physics: Conference Series*, vol. 1828, p. 012001, 02 2021.

[6] A. A. Shaghouri, R. Alkhatib, and S. Berjaoui, "Real-time pothole detection using deep learning," *CoRR*, vol. abs/2107.06356, 2021, accessed on: Nov. 18, 2021. [Online]. Available: https://arxiv.org/abs/2107.06356

[7] C. Chun and S.-K. Ryu, "Road surface damage detection using fully convolutional neural networks and semi-supervised learning," *Sensors*, vol. 19, no. 24, 2019, accessed on: Dec. 12, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/19/24/5501

[8] V. S. K. P. Varma, S. Adarsh, K. I. Ramachandran, and B. B. Nair, "Real time detection of speed hump/bump and distance estimation with deep learning using gpu and zed stereo camera," *Procedia Computer Science*, vol. 143, pp. 988–997, 2018, 8th International Conference on Advances in Computing Communications (ICACC-2018). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918320295

[9] T. T. A.Ş., "Kasis uyarı sistemi."

[10] S. K. Brett Hoye, "Pothole detection."

[11] "Digilent," accessed on: Nov. 18, 2021. [Online]. Available: https://www.elektrovadi.com/VmodCAM-Stereo-Camera-Module,PR-1368.html

[12] "Astra series," Mar 2021, accessed on: Nov. 18, 2021. [Online]. Available: https://orbbec3d.com/product-astra-pro/

[13] “80.09us $ 6% off: Senkronizasyon 960 p Çift lens usb kamera modülü mjpeg 60fps 2560x960 ov9750 cmos stereo video usb kamera 3d vr kamera: camera for: camera modulecameras camera - aliexpress,” accessed on: Nov. 18, 2021. [Online]. Available: https://tr.aliexpress.com/item/32839959377.html?src=google&memo1=freelisting&aff_fcid=04910ac013d740728bffe6dc221e324c-1634068849682-08940-UneMJZVf&aff_fsk=UneMJZVf&aff_platform=aaf&sk=UneMJZVf&aff_trace_key=04910ac013d740728bffe6dc221e324c-1634068849682-08940-UneMJZVf&terminal_id=483c9c59da8d4673bc2b77000d17696f

[14] “Intel realsense d415 stereo derinlik (depth) kamerası - robot sepeti,” accessed on: Nov. 18, 2021. [Online]. Available: https://www.robotsepeti.com/intel-realsense-depth-camera-d415?gclid=Cj0KCQjw5JSLBhCxARIsAHgO2SeNd_IOeJffl3blB6AFqPS31oYxtBK_9NRtaqzsdtBzB_cg3dYQOMwaAkZqEALw_wcB

[15] Banggood.com, “Hbv-1780-2 1mp ov9732 free driver binocular camera module for 3d depth detection of human eye,” accessed on: Nov. 18, 2021. [Online]. Available: https://www.banggood.com/HBV-1780-2-1MP-OV9732-Free-Driver-Binocular-Camera-Module-for-3D-Depth\-Detection-of-Human-Eye-p-1704210.html?cur_warehouse=CN&rmmds=search

[16] “Zed 2 - ai stereo camera,” accessed on: Nov. 18, 2021. [Online]. Available: https://www.stereolabs.com/zed-2/

[17] H. Hirschmüller, “Accurate and efficient stereo processing by semi-global matching and mutual information,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 807–814, 06 2005.

[18] H. Hirschmuller, “Stereo processing by semiglobal matching and mutual information,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[19] OpenCV, “Opencv cv::stereosgbm class reference,” accessed on: Mar. 26, 2022. [Online]. Available: https://docs.opencv.org/3.4/d2/d85/classcv_1_1StereoSGBM.html

[20] S. Birchfield and C. Tomasi, “A pixel dissimilarity measure that is insensitive to image sampling,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 401–406, 1998.

[21] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[22] Roboflow, “Roboflow - annotate,” accessed on: Nov. 18, 2021. [Online]. Available: https://docs.roboflow.com/annotate

[23] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020, accessed on: Nov. 18, 2021. [Online]. Available: https://arxiv.org/abs/2004.10934

[24] J. Fritsch, T. Kuehnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[25] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484–1497, 2012.

[26] A. Singh, A. Kamireddypalli, V. Gandhi, and K. M. Krishna, "Lidar guided small obstacle segmentation," *arXiv preprint arXiv:2003.05970*, 2020.

[27] V. S. K. P. VARMA, "Speed hump/bump dataset," 2018, accessed on: Nov. 18, 2021. [Online]. Available: https://data.mendeley.com/datasets/xt5bjdhy5g/1

[28] B. T. Passos, "Cracks and potholes in road images," 2020, accessed on: Nov. 18, 2021. [Online]. Available: https://data.mendeley.com/datasets/t576ydh9v8/4

[29] "Potholes dataset," accessed on: Nov. 18, 2021. [Online]. Available: https://makeml.app/datasets/potholes

[30] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013, accessed on: Nov. 18, 2021. [Online]. Available: http://arxiv.org/abs/1311.2524

[31] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015, accessed on: Nov. 18, 2021. [Online]. Available: http://arxiv.org/abs/1506.01497

[32] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016, accessed on: Nov. 18, 2021. [Online]. Available: http://arxiv.org/abs/1612.08242

[33] A. Farhadi and J. Redmon, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018, accessed on: Nov. 18, 2021. [Online]. Available: http://arxiv.org/abs/1804.02767

[34] G. Jocher, A. Stoken, A. Chaurasia, J. Borovec, NanoCode012, TaoXie, Y. Kwon, K. Michael, L. Changyu, J. Fang, A. V, Laughing, tkianai, yxNONG, P. Skalski, A. Hogan,

J. Nadar, imyhxy, L. Mammana, AlexWang1900, C. Fati, D. Montes, J. Hajek, L. Diaconu, M. T. Minh, Marc, albinxavi, fatih, oleg, and wanghaoyang0106, "ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support," Oct. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5563715

[35] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[36] L. Hulsey, "Smarter cars reduce accidents, but dumb cars are still the norm," 2017, accessed on: Apr. 14, 2022. [Online]. Available: https://www.govtech.com/fs/transportation/smarter-cars-reduce-accidents-but-dumb-cars-are-still-the-norm.html