

# **Project Report**

**Entitled**

## **“POTHOLE DETECTION ROBOT USING ROS AND OpenCV/ DEEP LEARNING”**

*Submitted to the Department of Electronics Engineering  
In Partial Fulfillment of the Requirement for the Degree of*

### **Bachelor of Technology**

**( ELECTRONICS & COMMUNICATION )**

**: Presented & Submitted By :**

Mr Pankaj Kumar Vijayvergiya

(Roll No. U17EC122)

Mr Prakash Saini

(Roll No. U17EC151)

Mr Sarvesh Dubey

(Roll No. U17EC152)

Mr Dhruvil Parikh

(Roll No. U17EC153)

B. TECH. IV (EC), 8<sup>th</sup> Semester

**: Guided By :**

Prof. A. H. Lalluwadia

Associate Professor, ECED.



**(Year: 2020-21)**

**DEPARTMENT OF ELECTRONICS ENGINEERING**

Sardar Vallabhbhai National Institute of Technology

Surat-395007, Gujarat, INDIA.

# **Sardar Vallabhbhai National Institute of Technology**

Surat-395 007, Gujarat, INDIA.

## **ELECTRONICS ENGINEERING DEPARTMENT**



### **CERTIFICATE**

This is to certify that the **PROJECT REPORT** entitled **“POTHOLE DETECTION ROBOT USING ROS AND OpenCV/ DEEP LEARNING”** is presented & submitted by Candidate **Mr. Pankaj Kumar Vijayvergiya, Mr. Prakash Saini, Mr. Sarvesh Dubey** and **Mr. Dhruvil Parikh**, bearing **Roll No. U17EC122, U17EC151, U17EC152** and **U17EC153**, of **B.Tech. IV, 8<sup>th</sup> Semester** in the partial fulfillment of the requirement for the award of **B.Tech** Degree in **Electronics & Communication Engineering** for academic year 2020-21.

They have successfully and satisfactorily completed their **Project Exam** in all respect. We, certify that the work is comprehensive, complete and fit for evaluation.

*A. H. Lalluwadia*

Prof. A.H. Lalluwadia  
Associate Professor &  
Project Guide

### **PROJECT EXAMINERS:**

#### **Name of Examiner**

#### **Signature with date**

1. Dr. J. N. Sarvaiya

2. Dr. K. P. UPLA

3. Dr. Suman Deb

4. Dr. Raghavendra Pal

Dr. P. N. Patel

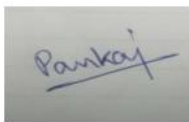
Associate Professor &  
Head, ECED, SVNIT.

**DEPARTMENT SEAL**  
(May 2021)

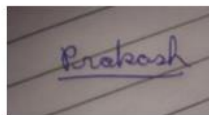
# ACKNOWLEDGEMENT

It is a privilege to express our sincerest regards to the Project guide, **Prof. A. H. Lalluwadia** (Associate Professor) for his valuable inputs, able guidance, encouragement, whole-hearted cooperation, and constructive criticism throughout the duration of this project.

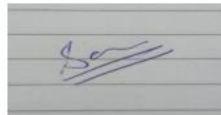
We would like to extend our sincere thanks to the Head of the Department Prof. P.N. Patel for encouraging and allowing us to present the project “Pothole Detection Robot using ROS and OpenCV/Deep Learning” at our department premises for the partial fulfillment of the requirements leading to the award of B-Tech degree. We would also like to take this opportunity to thank all our lecturers who have directly or indirectly helped us by imparting the necessary knowledge of the field. We would also like to pay our deepest respects and love to our family members and friends for their love and encouragement.




Mr. Pankaj Kumar  
Vijayvergiya  
(U17EC122)



Mr. Prakash Saini  
(U17EC151)



Mr. Sarvesh Dubey  
(U17EC152)



Mr. Dhruvil Parikh  
(U17EC153)

# ABSTRACT

## **PROJECT TITLE: POTHOLE DETECTION ROBOT USING ROS AND OpenCV/ DEEP LEARNING**

The main objective of the project is to design and fabricate a Semi Automated differential drive Robot, which will detect the Pothole on the road with the help of Deep Learning algorithms and taking the help of OpenCV image processing library.

The pothole detection bot, in the Gazebo World (Simulation Environment) will survey the roads and provide live feed which can also be recorded and saved on the hard-drive for further review by the authorities. This differential drive bot has a camera mounted on the top to facilitate the purpose.

The robot is also equipped with path-planning algorithms to enable efficient navigation along the road. With the collected data, we hope to alert the concerned department and the users to take necessary precautions while travelling through a particular area or to avoid the areas where major damage has occurred. As a result of the accurate data, it will prove to be economic for the government department as they will know exactly the amount of resources needed.

Mr. Pankaj Kumar Vijayvergiya (U17EC122)

Mr. Prakash Saini (U17EC151)

Mr. Sarvesh Dubey (U17EC152)

Mr. Dhruvil Parikh (U17EC153)

Guide Name: Prof. A. H. Lalluwadia

Date of Exam: 12/05/2021

Timeslot of Seminar Exam: 10:00AM to 1:00PM

Examiner Name: Dr. J.N. Sarvaiya, Dr. K.P. Upla, Dr. Suman Deb & Dr. Raghavendra Pal

# TABLE OF CONTENT

CHAPTER 1 .....	9
INTRODUCTION .....	9
1.1 Background.....	9
1.2 Objective.....	12
1.3 Project Outline .....	13
CHAPTER 2 .....	14
LITERATURE SURVEY.....	14
2.1 Public Reporting .....	14
2.2 Vibration-Based Methods.....	14
2.3 2D-Vision-Based Methods .....	15
2.4 3D Scene Reconstruction-Based Methods .....	17
2.5 Learning-Based Methods.....	21
2.6 Modern Based Pothole Detection Method.....	22
2.6.1 Tensorflow Object-detection API.....	22
2.6.2 Transfer Learning .....	23
2.6.3 F-RCNN (Faster Region-based Convolutional Neural Network) .....	23
2.6.4 Inception – V2 .....	24
CHAPTER 3 .....	25
MECHANICAL MODULE .....	25
3.1 Design Overview .....	25
3.2 Robot Structure DesignUsing 3D CAD (Solidworks).....	25
3.3 The Description Model .....	26
3.4 The TF Tree .....	26

3.5 Motion Analysis For Joint Torque Validation.....	27
3.6 Selection of Optimistic Drive System .....	27
3.6.1 Differential Drive System.....	27
3.6.2 Ackermann Steering System .....	29
3.6.3 Omnidirectional System .....	30
3.7 Remark .....	31
CHAPTER 4.....	32
MODELLING AND SIMULATION.....	32
4.4.2 Physical Model Description .....	35
4.5 Vision Camera .....	37
CHAPTER 5.....	39
DETECTION MODULE.....	39
5.2 Proposed Approach .....	39
5.3 ResNet .....	40
5.3.1 Background.....	40
5.3.2 Motivation .....	40
5.3.3 What problems ResNets solve? .....	40
5.3.4 Architecture .....	41
5.3.5 Convolution .....	43
5.3.6 ResNet Layers .....	45
5.4. Comparison: .....	49
Table 5.3: Comparison between the ResNet34 and InceptionV3.....	49
CHAPTER 6.....	50
RESULTS.....	50
6.1 Teleoperation.....	50

6.2 Model Accuracy & Loss:.....	50
6.3 Prediction:.....	52
CHAPTER 7 .....	54
SUMMARY AND CONCLUSION .....	54
CHAPTER 8.....	55
FUTURE SCOPE .....	55
REFERENCES .....	56

# LIST OF FIGURES

Figure 1.1 Potholes.....	9
Figure 1.2 Number of vehicles vs usage of the vehicle.....	10
Figure 2.1 Recorded road scene with transparent color-encoded.....	17
Figure 2.2 Reconstructed 3D point cloud using calculated disparities.....	17
Figure 2.3 Obtained elevation-difference map using best-line fit in y-disparity space....	21
Figure 3.1 CAD (Solidwork Model).....	24
Figure 3.2 Sample URDF model.....	25
Figure 3.3 Kinematic Chain of robot generated by URDF.....	27
Figure 3.4 Differential drive system.....	28
Figure 3.5 Ackermann Steering.....	29
Figure 3.6 Four wheeled omni robot.....	29
Figure 4.1 General structure of Gazebo World.....	33
Figure 4.2 A simulated environment in Gazebo.....	34
Figure 4.3 Pothole detection bot in Gazebo.....	35
Figure 4.4 Visualisation of simulated environment.....	37
Figure 5.1 Model Architecture.....	38
Figure 5.2 Comparison of ResNet vs VGG.....	41
Figure 5.3 Another look at ResNet34.....	42
Figure 5.4 Convolution.....	43
Figure 5.5 Max Pooling.....	43
Figure 5.6 Layer 1, block 1, operation 1.....	44
Figure 5.7 Layer 1, block 1.....	45
Figure 5.8 Layer 1.....	45
Figure 5.9 Layer 2, block 1, operation 1.....	46
Figure 5.10 Projection Shortcut.....	47



Figure 5.11 Layer 2, block 1.....47

Figure 5.12 Layer 2.....48

Figure 6.1 Road traversing through Teleoperation.....49

Figure 6.2 Model Accuracy.....50

Figure 6.3 Model Loss.....50

Figure 6.4 Prediction on test images.....51-52

# LIST OF TABLES

Table 2.1 Public reporting.....	13
Table 2.2 Vibration Based Methods.....	14
Table 2.3 Examples of 3D Reconstruction-based methods and used sensors.....	18
Table 2.4 Examples of CNNs for Image Segmentation, and Used Data Sets.....	19
Table 4.1 Comparison between general specifications of simulated software.....	31
Table 5.1 Parameter Table.....	40
Table 5.2 ResNet Architectures.....	48
Table 5.3 Comparison between ResNet34 and InceptionV3.....	48

# CHAPTER 1

## INTRODUCTION

### 1.1 Background

Roads make a crucial contribution to economic development and bring important social benefits. They are of vital importance in order to make a nation grow and develop. Roads open up more areas and stimulate economic and social development. For those reasons, road infrastructure is the most important of all public assets. But due to repeated loading and weathering on roads, a pothole may be caused which may affect the human life very badly.

The number of vehicles drastically increases every year, and the number of accidents proportionally does too. The condition of the road surface affects our safety directly. The American Automobile Association estimated in the five years prior to 2016 that 16 million drivers in the United States had suffered damage from potholes to their vehicle with a cost of 3 billion USD a year [1]. In India, 3,000 people per year are killed in accidents involving potholes. Britain has estimated that the cost of fixing all roads with potholes in the country would cost 12 billion. According to the World Health Organization, road traffic injuries caused an estimated 1.25 million deaths worldwide in the year 2010. That is, one person is killed every 25 seconds. Only 28 countries, representing 449 million people (seven percent of the world's population), have adequate laws that address all five risk factors (speed, drunk driving, helmets, seat-belts and child restraints) [3]. By the way, there is a close relationship between the accident and road conditions including a pothole. According to Austroads [4], road accidents occur as the result of one, or more than one of the following factors: human factors, vehicle factors, road and environment factors. Vogel and Bester [5] introduced risk factors (human, vehicle and environment factors) for 14 accident types that can be used as a reference point to determine the likely cause of an accident of a specific type. A research had been done from a little bit different point of view, where the researchers proposed a cost-effective solution to identify the potholes and humps on roads and

provide timely alerts to drivers to avoid accidents or vehicle damages. Ultrasonic sensors are used to identify the potholes and humps [6].

A pothole is a structural failure in a road surface, caused by failure primarily in asphalt pavement due to the presence of water in the underlying soil structure and the presence of traffic passing over the affected area.



Fig.1.1: Potholes [5]

India has the second largest population in the world with one of the fastest growing economies. As the population rises, the need for well-maintained roads is also increasing as road transport is one of the most primary means of transport in the country. However, most of the roads in the country are narrow, congested and of very poor quality. Dangerous road surface conditions are a major distraction for safe and comfortable transportation. Driving a vehicle in India is potentially life taking.

Pothole is a depression in the road surface caused by wear or subsidence. Potholes are formed because of substantial downpours and poor drainage framework in urban areas. Research for the Department for Transport has shown that the public's main concern on roads is potholes. Potholes stand out above all other defects as the most unacceptable of all conditions. Nearly everyone has this at the top of their scale. Potholed roads are a common sight across rural and urban India especially during and after monsoons. Every year crores and crores of rupees are spent by the highway agencies in extensive pot-hole patch repairs. Because of adverse media

coverage these agencies do patch repairs of main streets in urban areas and main highways in rural areas after the monsoon season is over. By lanes in towns and cities and some secondary roads in rural areas usually remain neglected for years. This ritual is repeated year after year even though several lakhs of people are involved in accidents due to potholes causing serious injuries and in many cases fatalities. Both drivers and road maintainers are interested in fixing the potholes as soon as possible. However, they must be identified first.

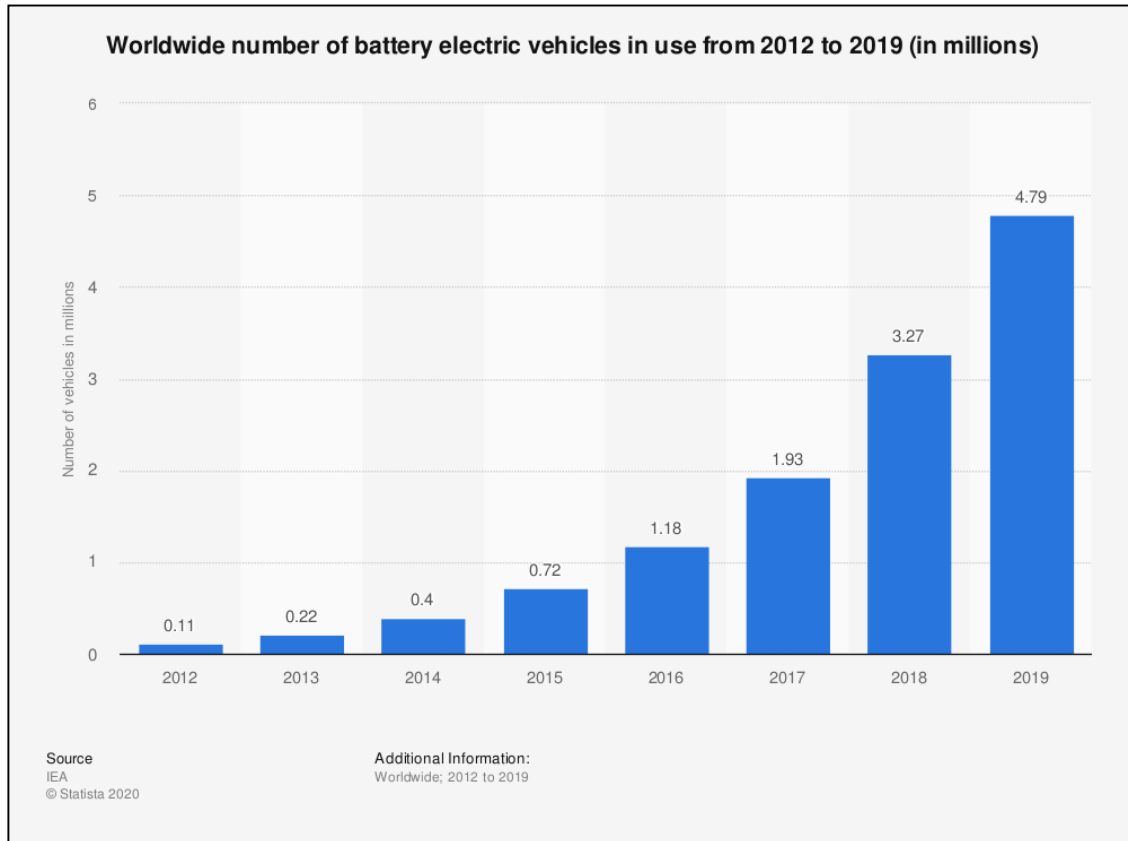


Fig.1.2: Number of vehicles vs Usage of the vehicle [10]

Street Surface Monitoring (RSM) is the way toward identifying the misery on cleared or unpaved street surfaces. The essential point of this procedure is to distinguish any pain, (for example, street surface splits) at early stages so as to apply upkeep on time. Early recognition of street surface breaks can help support before the repair costs turn out to be too high. What's more, measuring the condition and execution of streets are ceaselessly grown with additional time with the new strategies and upgrades.

The potholes happen because of overwhelming precipitation or because of the substantial vehicles out and about. These potholes turn into the principle purpose behind the street mischance. On major the greater part of the mishaps are because of the potholes and the lessening in the weight of the tyre.

According to the survey report “Road Accidents in India,2011”, by the ministry of road transport and highways, a total of 1,42,485 people had lost their lives due to fatal road accidents. Of these, nearly 1.5 per cent or nearly 2,200 fatalities were due to poor condition of roads. According to most recent figures by a few state governments, potholes across the nation have claimed 3,597 lives in 2017, an over half ascent in the toll than a year ago. Around 30 deaths happen day by day on streets because of potholes.

## **1.2 Objective**

Our project is to make a robot which helps the society in promoting road safety and to reduce the difficulties in detecting the pothole.

We designed a Semi-Automatic Robot which will detect the pothole on the road, and hence the accidents that occur due to the pothole may be reduced.

The simplest method to detect potholes is collecting photos of road damage and hazards taken by the participants and up-loading them to a central server. However, it requires strong participation from the users. An automated approach to detect potholes with little or no human interaction is more promising. This would ensure more comprehensive survey data with fewer errors caused by human factors than generated by mere enthusiasm of the participants. Some pothole detection methods have been proposed previously and they can be classified into two groups: Image recognition method and mobile sensing method. Image recognition methods include image processing techniques to collect and analyze the road information to detect potholes. In mobile sensing methods, accelerometers and gyro sensors are incorporated and the data processing centre analyses these data to check whether the data exceed the thresholds for pothole detection.

The identification and estimation of the asphalt breaking gives important data to the nearby experts out and about system condition and decreases support costs. Huge advances have been

made as of late in utilizing an assortment of methods for evaluating the asphalt street surface. For proficient gathering of asphalt condition information, distinctive methodologies have been proposed, and different robotized frameworks have been created world broadly since the 1980s. Past ways to deal with asphalt conditions included a work escalated, tedious, and unsafe procedure of information accumulation. A brisk, simple to utilize, and financially savvy strategy for building up a Raspberry Pi based street location technique is proposed in this paper.

### **1.3 Project Outline**

The layout of the remainder of the report is as per the following:

Chapter 1: Brief information about the pothole situation in India and accidents caused by them.

Chapter 2: Detailing on the previous works done in this field and an overview of the technology used for the said purpose.

Chapter 3: The mechanical model of the Robot.

Chapter 4: Deals with Gazebo (Simulation Environment), integrated with ROS and the virtual world in which the bot is deployed.

Chapter 5: Pothole detection with ResNet Architecture.

Chapter 6: Overview of how different components of the software are integrated.

Chapter 7: Summary and conclusion.

Chapter 8: Deals with Future Scope of the Project

## CHAPTER 2

# LITERATURE SURVEY

This survey contributes to the motivation of developing automated road-surface anomaly detection systems for various real-world environments. There have been many advancements in this technical era recently.

### 2.1 Public Reporting

This type of system enhances civic engagements by government, and facilitates the participation by citizens of the country. These systems use the public as sensors [10]. The main advantage of this method is that there is no need for costly hardware or software. Citizens can report a pothole by capturing its picture with their mobile devices and later by uploading or sending to a website or application, or by merely sending information about a pothole's location. Some reported systems are listed in Table 2.1.

App/Website	Countries
FixMyStreet	UK, New Zealand
SeeClickFix	US
Citizens-Connect	Netherlands, Canada
PDX Reporter	Portland
Report a Pothole	London
BBMP	Bangalore, India
Citizen Hotline 1999	Taoyuan, Taiwan

Table 2.1: Public Reporting; Listed names on the left identify the websites of those application (for example [www.fixmystreet.com](http://www.fixmystreet.com)). Citizen hotline 1999 is the name of an innovative open data platform used in Taiwan. The related research publication was in 2017 [3]

### 2.2 Vibration-Based Methods

Vibration-based methods include approaches of collecting abnormal vibrations [17] caused in the vehicles while driving over road anomalies. Vibrations of the vehicle are collected using an accelerometer; see Table 2.2. The main drawback of the vibration-based methods is that the



vehicle has to drive over the pothole in order to measure the vibrations caused by the pothole on the road. M. Ghadge et al. [18] used an accelerometer and GPS to analyze the conditions of roads to detect locations of potholes and bumps using a machine learning approach, defined by K-means clustering on training data and a random-forest classifier for testing data. The data is divided first into two clusters of “pothole” or “non-pothole”, and then a random forest classifier is used to validate the proposed result provided by the clustering algorithm. It is reported that clustering does not perform well when clusters of different size and severity are involved; size and severity of a pothole are the major properties considered in the system. F. Seraj et al. [19] used a support vector machine (SVM) for a machine-learning approach to classify road anomalies. The proposed system uses accelerometer, gyroscope and a Samsung galaxy as sensors for data collection; data labeling is performed manually (by a human) and then a high-pass filter is used to remove the low-frequency components caused due to turns and accelerations. Ren et al. [20] used K-means clustering to detect potholes based on data collected by using an accelerometer and GPS. The proposed system lacks accuracy regarding the isolation of potholes from other road anomalies.

Authors	Years	Sensors
Jintin Ren et al.	2017	Accelerometer, GPS
Fatjon Seraj et al.	2016	Accelerometer, gyroscope and GPS
Marcin Badurowicz et al.	2016	Smartphone, Accelerometer
Chih-Wei et al.	2015	Accelerometer, Smartphone
Manjusha Ghadge et al.	2015	Smartphone, Accelerometer,
Mohamed Fekry et al.	2014	Accelerometer and GPS
Artis Mednis et al.	2011	Accelerometer, Smartphone, Laptop
Thegaran Naidoo et al.	2011	Accelerometer and GPS

Table 2.2: Vibration-Based Methods [5]

## **2.3 2D-Vision-Based Methods**

Vision-based methods use 2-dimensional (2D) image or video data, captured using a digital camera, and process this data using 2D images or video processing techniques [21], [22]. The choice of the applied image processing techniques is highly dependent on the application for which 2D images are being processed. Koch and Brilakis [8] proposed a method aiming at a separation of defect and non-defect regions in an image using histogram shape based

threshold. The authors consider the shape of a pothole as being approximately elliptical based on a perspective view. The authors emphasize on using machine learning in future work, and claim that the proposed work already results in 86% Accuracy along with 86% Recall and 82% Precision, with the common definitions of

$$\text{Precision} = \frac{TP}{TP + FP} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

where TP is the number of true positives, FP of false positives, TN of true negatives, and FN of false negatives. Tedeschi and Benedetto [10] recently suggested a system for automatic pavement distress recognition (ADPR) which is able to perform in real time by identifying road distress including fatigue cracks, longitudinal and transverse cracks, and potholes. The authors used a combination of technologies of the OpenCV library and for the classification of the three different types of road distresses, three classifiers have been used based on local binary pattern (LBP) features; they achieved more than 70% for Precision, Recall, and the F1-measure. Authors discussed difficulties of defining the severity of considered kinds of road distresses. For texture classification the authors used Haralick's features [23] based on gray level co-occurrence matrices (GLCMs) and then classified image regions using a tool from [24]. Ryu et al. [26] proposed a method to detect potholes both for asphalt or concrete road surfaces using 2D images collected by a mounted optical device on a survey vehicle. The system mainly works in three steps of image segmentation, candidate region extraction and decision. The system fails to detect potholes in darker images (image regions) due to shadows (e.g. of trees or cars) present in real-world road recordings. Powell and Satheesh Kumar [27] present a method for the detection of potholes by segmenting images into defected or non-defective regions. After extracting the texture information from defected regions, this texture information is compared with texture information obtained from non-defective regions.

The proposed system considers shadow effects on the road and aims to remove those effects of shadows using a shadow removal algorithm. The system is unable to perform in rainy weather. The authors concluded that the system should be further extended to perform also on video data as the system was only tested on 2D images collected using an iPhone camera with 5 megapixel image resolution. Bashkar and Manohar [44] propose a methodology of detecting pothole's mean depth by using SURF features on uncalibrated stereo pairs of images (without employing disparity images). A particular methodology has been developed for this purpose, but appears to suffer from uncalibrated stereo rectification; it is far from providing good results. Ying et al. [46] proposed a system which can detect road surface based on a feature detector which is shadow-occurrence optimized. This system uses a connected-component-analysis algorithm and other morphological algorithms and is demonstrated on images of datasets provided by KITTI [47] and ROMA [48]. Thekkethala et al. [25] used two (stereoscopic) cameras and applied stereo matching to estimate depth of a pothole on asphalt pavement surface. After performing binarization and morphological operations, a skeleton of a pothole is estimated. The system is tested on 24 images and no estimates of depth have been provided. The system can detect skeletons of potholes of great depression. Authors did not estimate the road manifold.

## **2.4 3D Scene Reconstruction-Based Methods**

3D scene reconstruction is the method of capturing the shape, depth, and appearance of objects in the real world; it relies on 3D surface reconstruction which typically demands more computations than 2D vision. Rendering of surface elevations helps to understand accuracy during the design of 3D vision systems. 3D scene reconstruction can be based on using various types of sensors, such as Kinect [28], stereo vision cameras, or a 3D laser. Kinect sensors are mainly used in fields of (indoor) robotics or gaming. 3D lasers define an advanced road-survey technology; compared to camera-based systems it still comes with higher costs; [30], [31] report survey cycles of (usually) once in four years. A 3D laser uses a laser source to illuminate the surface and a scan camera for capturing the created light patterns. [32] applied the common laser-line projection; the recorded laser line deforms when it strikes an obstacle (and supports thus the 3D reconstruction), but does not work well, e.g., on wet roads or potholes filled with water. Stereo-vision cameras are considered to be cost-effective as

compared to other sensors. Stereo vision aims at effective and accurate disparities, calculated from left-and-right image pairs, to be used for estimating depth or distance; see, for example, [33]. Commonly, the canonical left-right calibrated stereo camera setup is used while aiming at a reconstruction of dense 3D surfaces.



Fig.2.1: Recorded road scene with transparent color-encoded disparity map calculated using stereo global matching algorithm [29]. The used color key is shown on the right; disparity 250 encodes a distance very close to the host vehicle and 0 encodes very far away [9]

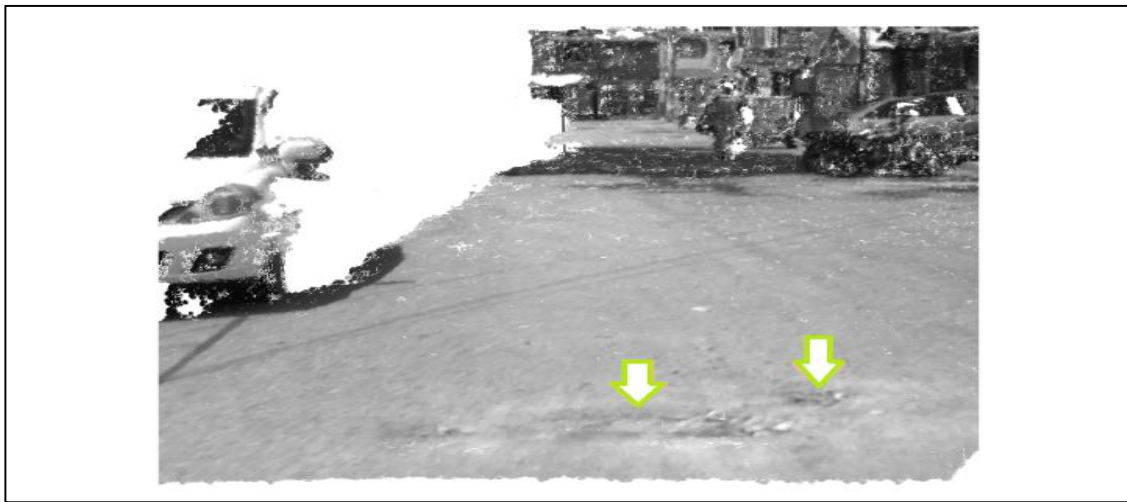


Fig.2.2: Reconstructed (and texture-mapped) 3D point cloud using calculated disparities. Green arrows indicate locations of two potholes [9]

For briefly introducing the stereo-vision notation (later needed in this paper), consider coordinates  $(x, y) \in$  of a pixel in an image, where it denotes the image domain of the left

image. A disparity map  $D: \rightarrow \mathbb{R}^+ \cup \{0\}$  defines the translation of image coordinates in the left image into those of a detected corresponding pixel  $(x - D(x, y), y)$  in the right image.

A 3D point  $(X, Y, Z)$  in the 3D scene is mapped into an image pixel at  $(x, y)$  following a perspective projection

$$x = f_x \cdot \frac{X}{Z} + x_c, \quad y = f_y \cdot \frac{Y}{Z} + y_c \quad (5)$$

where  $f_x$  and  $f_y$  are the focal lengths in  $x$  and  $y$  coordinate direction, and  $(x_c, y_c)$  is the principal point in the image plane. Given a calibrated and rectified stereo camera pair, let  $d = D(x, y)$  be the disparity value assigned to a left-image pixel location  $(x, y)$ , The  $Z$ -coordinate can be triangulated as follows:

$$Z = f_x \cdot \frac{b}{d} \quad (6)$$

Here,  $b$  is the length of the baseline which is the distance between left and right camera optical centers. This allows us that  $X$  and  $Y$  can also be recovered; the whole process is called triangulation.

Authors	Years	Sensors
Tomasz Garbowski et al.	2017	Stereo-vision cameras
Vijaya Bashkar et al.	2016	Stereo-vision cameras
Aliaksei Mikhailiuk et al.	2016	MicroController TMS320C6678 DSP
A. Rasheed et al.	2015	Kinect sensors
Marcin Staniek	2015	Stereo-vision cameras, GPS and vibration sensor
Kiran Kumar Vupparaboina et al.	2015	Laser, camera
Zheng Zhang et al.	2014	Stereo-vision PointGrey Flea 3 cameras
He Youguan et al.	2011	Stereo-vision cameras and LED
X.Yu et al.	2011	Laser, camera

Table 2.3: Examples of 3D Reconstruction-Based Methods and Used Sensors [7]

Table 2.3 summarizes a few 3D reconstruction-based methods for detecting road distress. Garbowski and Gajewski [41] presented a semi-automatic pavement failure detection system (PFDS) which is a part of the FEMat [42] road package (UDPhoto toolbox). It allows a user

to inspect the condition of road pavement based on calculated clouds of 3D points.. Shen et al. [43] propose the use of Takata's stereo-vision system for performing a road surface preview along the host vehicle. Video data recorded with the used compact stereo vision sensor (with a baseline of 16.5 cm) is analyzed in an embedded system, already tested in various vehicles, also in combination with various driver assistance systems such as forward collision warning (FCW), automatic emergency braking (AEB), or lane departure warning (LDW). Authors state that the proposed system achieves satisfactory accuracy; they also state that it does not perform well when there is glare on the road surface. Calculated disparities within detected road-surface image segments support the estimation of a manifold, approximating the road-surface. Commonly the road surface is assumed to be planar (i.e., the manifold is thus a plane). But this planarity assumption is often not corresponding to actual uneven road surfaces. To simplify, the road manifold is often modeled in driving direction by a profile, i.e. a curve whose parallel translation left-to-right creates the road manifold. A line creates a plane, and a quadratic polynomial profile creates a quadratic road manifold. Quadratic road manifolds are discussed by Ai et al. [6]. For a consideration of twisting and bending surfaces of roads, see Mikhailiuk and Dahnoun [55]; their algorithm has been implemented on a Texas Instrument C6678 multi-core SoC digital signal processor. Zhang et al. [34] proposed an efficient algorithm to estimate the size, depth, position and severity of potholes by modeling the road surface as a quadratic manifold by using a random sample census (RANSAC) approach.

CNN	Year	Used datasets
<i>FCN</i>	2014	PASCAL VOC
<i>SEGN</i>	2015	CamVid
<i>DilatedConvolutions</i>	2015	VOC2012, COCO
<i>DeepLab</i>	2014-2017	PASCAL 2012, CityScapes
<i>RefineNet</i>	2016	PASCAL 2012
<i>PSPNET</i>	2016	PASCAL 2012, CityScapes
<i>LargeKernelMatters</i>	2017	PASCAL 2012, CityScapes
<i>MaskR – CNN</i>	2017	COCO

Table 2.4: Examples of CNNs for Image Segmentation, and Used Data Sets [16]

Pothole detection and segmentation are achieved by using a connected-component labeling (CCL) algorithm. Proposed methods may also follow a multi-sensor approach [11]– [15].

Tseng et al. [25] developed an automated survey robot which performed in simulated test-field environments to detect five types of distress, namely alligator cracks, small patches, potholes, rectangular, and circular manhole covers.

## **2.5 Learning-Based Methods**

For identifying objects in image data, various convolutional neural networks (CNNs) have been proposed, see Table 2.4, such as Chen et al. [56], RefineNet [57], PSPNet [58], or the “large-kernel-matters” proposal in [59]. For identifying an object at pixel level, fully convolutional neural networks (FCNs) have been proposed; for example, see Long et al. [60], or SegNet by Badri Narayanan et al. [61]. Regarding road damage detection, Zhang et al. [67] propose a CrackNet to predict class scores for all the pixels in a considered damage. Song et al. [62] use a CNN approach to detect potholes. The authors used a smartphone as a sensor to acquire movement information and the InceptionV3 [63] classifier; they adapted the final fully-connected layer in the CNN to the given task. Maeda et al. [64] used a CNN, trained by using a vast dataset of road images collected in Japan, to detect road surface damage. Some authors used SSD Inception V2 [65] or SSD MobileNet [66] to identify different sorts of road damages. Detected road damages are identified by generating bounding boxes (i.e., not at pixel level). Staniek [49] uses stereo vision cameras for the acquisition of road surfaces in the form of clouds of 3D points. The author emphasizes on solving stereo matching by using a Hopfield neural network, which is a special case of a recurrent artificial neural network. The author achieved 66% accuracy when evaluating matching pixels (for 50 image pairs) using a CoVar method [54] for evaluation. The authors [50] have proposed a model to detect potholes based on YOLOv2 architecture. However, their reported architecture differs from our proposed model in LM2. Also the tested frames basically show not much more than potholes, while the real road scene is much more complex. The CNN model proposed by the authors [51] to detect potholes has been trained on a CPU and experiments show that CNN based model performs better than Conventional SVM based approach.



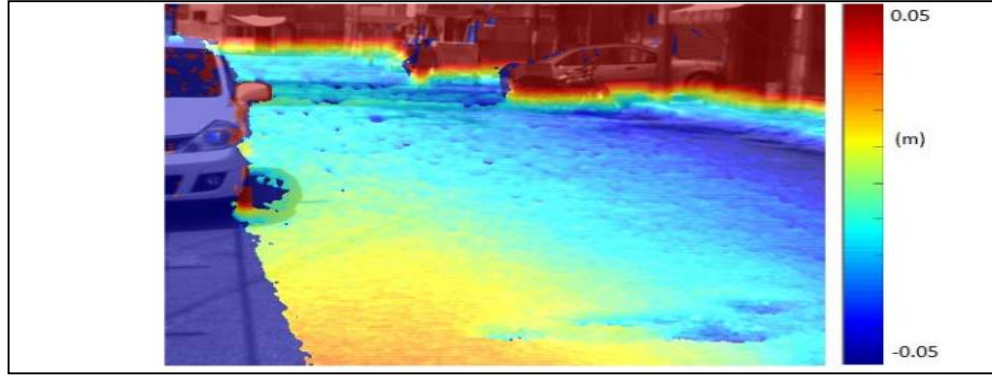


Fig.2.3: Obtained elevation-difference map using a best-line fit in y-disparity space. In the applied color key, blue corresponds to points being 5 cm or more below the road surface, and red to points being 5 cm or more above the road surface [19]

However, the system is not able to detect potholes under varying illumination conditions. The authors [20] have proposed a CNN based model mainly to classify a region on a road as pothole or non-pothole. The author has collected the dataset using a smartphone camera mounted on the front windshield of the vehicle and the authors have used preprocessed cropped frames with ROI to train the proposed model. The authors [20] have developed CNN based models using thermal images to classify whether an image has a pothole or not. The thermal images are recorded using a thermal camera.

## **2.6 Modern Based Pothole Detection Method**

The proposed modern based pothole detection method using the transfer learning technique detects potholes in videos/images in real-time. The method uses common techniques like “use of FRCNN”, “inception v2 model” which are described below. The main advantage of this method is small training time with an easy training process and higher accuracy.

### **2.6.1 Tensorflow Object-detection API**

Tensorflow’s object detection API is a very powerful tool that can quickly enable anyone to build and deploy a powerful image recognition system. It provides many pre-trained models (trained on different datasets) which can be used to build customized classifiers/detectors/recognizers after fine-tuning. We’ve selected the model named “F-RCNN inception v2”.



### **2.6.2 Transfer Learning**

Transfer Learning makes use of knowledge gained while solving one problem and applies it to a different but related problem. With this technique, we can save a lot of time. In this technique below first, we select any pre-trained model (In which all the parameters are trained), then we perform fine tuning. We fetch the new dataset to fine-tune the pre-trained CNN. If the new dataset is similar to the original dataset (with which the model has been trained), the same weights can be used for extracting the features from the new dataset. In our case, the dataset is very different from the original dataset. The Earlier layers of CNN contain more generic features (edge detector, colour blob detectors), but the later layers of CNN become progressively more specific to the details of the classes contained in the original dataset. So, earlier layers can help to extract the features of the new data. We fixed the earlier layers and Re-train the rest of the layers (because of the small amount of data).

### **2.6.3 F-RCNN (Faster Region-based Convolutional Neural Network)**

It stands for Faster Region-based Convolutional Neural Network. Before talking about Faster RCNN we should know about Fast R-CNN. Fast RCNN is a detector that uses an external proposal or external selective search. It consists of External selective search, CNN with max pooling, ROI (Region of Interest) pooling layer, fully connected layers, and output layers. Fast R-CNN takes an input image and then with the help of CNN & max pooling layers, a convolutional feature map is extracted from the image. The ROI pooling layer performs a very important task here. We know that fully connected layers can accept only certain sizes, So ROI pooling layers convert the output of the CNN into certain fixed sizes. When we put Fast RCNN with the RPN (Region proposal network), it becomes Faster R-CNN. So, basically, the difference between Fast R-CNN and Faster R-CNN is the Region proposal. In Fast R-CNN, there is an external selective search whereas in Faster R-CNN RPN is combined with the Architecture. RPN is the Architecture that makes Fast R-CNN a Faster R-CNN. In order to reduce the computational complexity and requirement, RPN decides where to look in the image. It scans the image and gives k output boxes each with 2 scores indicating the probability of availability of an object. Different size and different aspect ratio of boxes are selected in order to accommodate different types of objects.

In order to get a trained working Faster R-CNN architecture, we need to perform the following steps:

1. Training of RPN – first of all, we need to Train the RPN architecture with the dataset so that it can propose the expected region.
2. Training of Fast R-CNN – As we know, Faster R-CNN is a combination of RPN and Fast R-CNN. So, we've to train a Fast R-CNN with the proposals obtained by RPN (after training) in order to make a Faster R-CNN.
3. Fixing Convolutional layers, fine-tuning unique layers to RPN.
4. Fixing Convolutional layers, fine-tuning fully connected layers of Fast R-CNN.

#### **2.6.4 Inception – V2**

Inception v2 is an upgraded version in the inception network series. It is a complex architecture of CNNs. Inception v1 is the first version of this series. Inception v1 performs convolution on input, with 3 different sizes of filters (1x1, 3x3, 5x5). Additionally, max pooling is also performed. The Outputs are concatenated and sent to the next inception module. Using this inception module, GoogLeNet neural network architecture (having 9 such inception modules) was built. Furthermore, to improve computational speed inception v2 architecture is introduced as shown in figure 03. In this architecture 5x5 convolution is factorized to two 3x3 convolution operations. A 5x5 convolution is 2.78 times more expensive than a 3x3 convolution. Moreover, a new method is introduced to improve performance. In the new method, the convolution of filter size  $n \times n$  is factorized to a combination of  $1 \times n$  and  $n \times 1$  convolutions. This method is 33% cheaper than the single  $3 \times 3$  convolution.

## CHAPTER 3

# MECHANICAL MODULE

### 3.1 Design Overview

In regards to the robot mechanical design, several design criteria need to be considered such as body weight and size, manufacturability, manufacturing time and cost. In the initial design process of this robot, it was very important to have a modular system for easy assembly and further maintenance. The driving mechanism consists of a DC motor, leg structure, bearings and output shaft which drives the wheel. For this robot, four of these wheel modules are attached to the chassis. This design approach allows initial analysis, tests and validation on one-wheel modules and also decreases the production time and cost for all four-wheel modules. The chassis structure is very simple and it can be built from an aluminum box section.

### 3.2 Robot Structure Design Using 3D CAD (Solidworks)

All the robot parts have been designed using 3D CAD to enable further stress analysis. For this aim, the educational version of SolidWorks 2016 software has been used to design the 3D model of the robot.

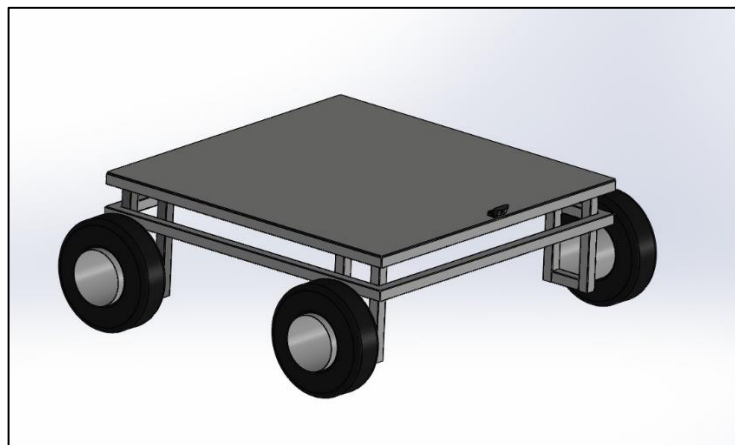


Fig.3.1: CAD (Solidworks Model)

SolidWorks is a CAD modelling and CAE software program which runs on Microsoft Windows. This design includes the parts which should be manufactured and also the other parts supplied from off the shelf. In designing the robot's body parts, assimilability with other parts such as motors and cameras has been considered. This 3D design allows us to study and analyze the mechanical interference between each part as joints and links.

### **3.3 The Description Model**

The description model is a file that describes the geometry, physics, and relative relationships of the various components of the robot/machine. This is achieved through the Unified Robot Description Format, or simply the URDF model of the robot, as shown in below the figure.

```
<joint name="1"
  type="fixed">
  <origin
    xyz="0.0569659303003536 -0.544982974215764 0.465904192054767"
    rpy="3.14159265358979 -6.12303176911189E-17 -1.5707963267949" />
  <parent
    link="base_link" />
  <child
    link="L1" />
  <axis
    xyz="0 0 0" />
</joint>
```

Fig.3.2: Sample URDF model [59]

The URDF is an XML file that describes the components of the machine; links, joints, grippers, sensors, etc. as well as their geometries, surface structures, colours, masses, inertias, joint/link types, motion types and ranges, etc. This XML file can then be visualized using ROS's RVIZ package, or any other open-source visualization platform, such as Moveit! or Gazebo, which ROS supports and is integrated smoothly with.

### **3.4 The TF Tree**

The TF tree describes the various frames of the robot; such as the base frame, joint frames, arm frame, sensor frame, etc. and their relative relationships. The TF tree is needed to perform

the complex robot kinematics operations; by combining the TF tree with the URDF model of the machine, all kinematics operations, forward or inverse, are then performed by ROS's TF package under the hood, saving developers from the burden of performing these complex mathematics operations manually.

### **3.5 Motion Analysis For Joint Torque Validation**

The kinematic chain of the robot generated by the URDF file is shown. In URDF terminology multiple links can be connected to one link by joints and defined parent and child links. Driving joints are defined as continuous. This robot definition allows us to use Gazebo for dynamic simulation and RVIZ for visualization. This robot has four DOFs in total on a robot local frame which provides three DOFs motion ( $V_x$ ,  $V_y$ , and  $\omega$ ) in the world frame.

### **3.6 Selection of Optimistic Drive System**

Generally, the locomotion control system of mobile robots can be divided into differential drive system, Ackermann driving system and omnidirectional system. Selection of the appropriate drive system depends on the type of application and the level of accuracy which is to be expected.

#### **3.6.1 Differential Drive System**

This is the most common control mechanism for robot builders. Velocity difference between two motors drive the robot in any required path and direction. Hence the name differential drive. Differential wheeled robots have two independently driven wheels fixed on a common horizontal axis.

There are three fundamental cases which can happen in a differential wheeled robot:

1. If the angular velocities are identical in terms of both values and direction, i.e. if both the wheels are driven at the same speed and same direction (either clockwise or anticlockwise) then the robot is more likely to follow a linear path, either forward or backward based on the motors spin.

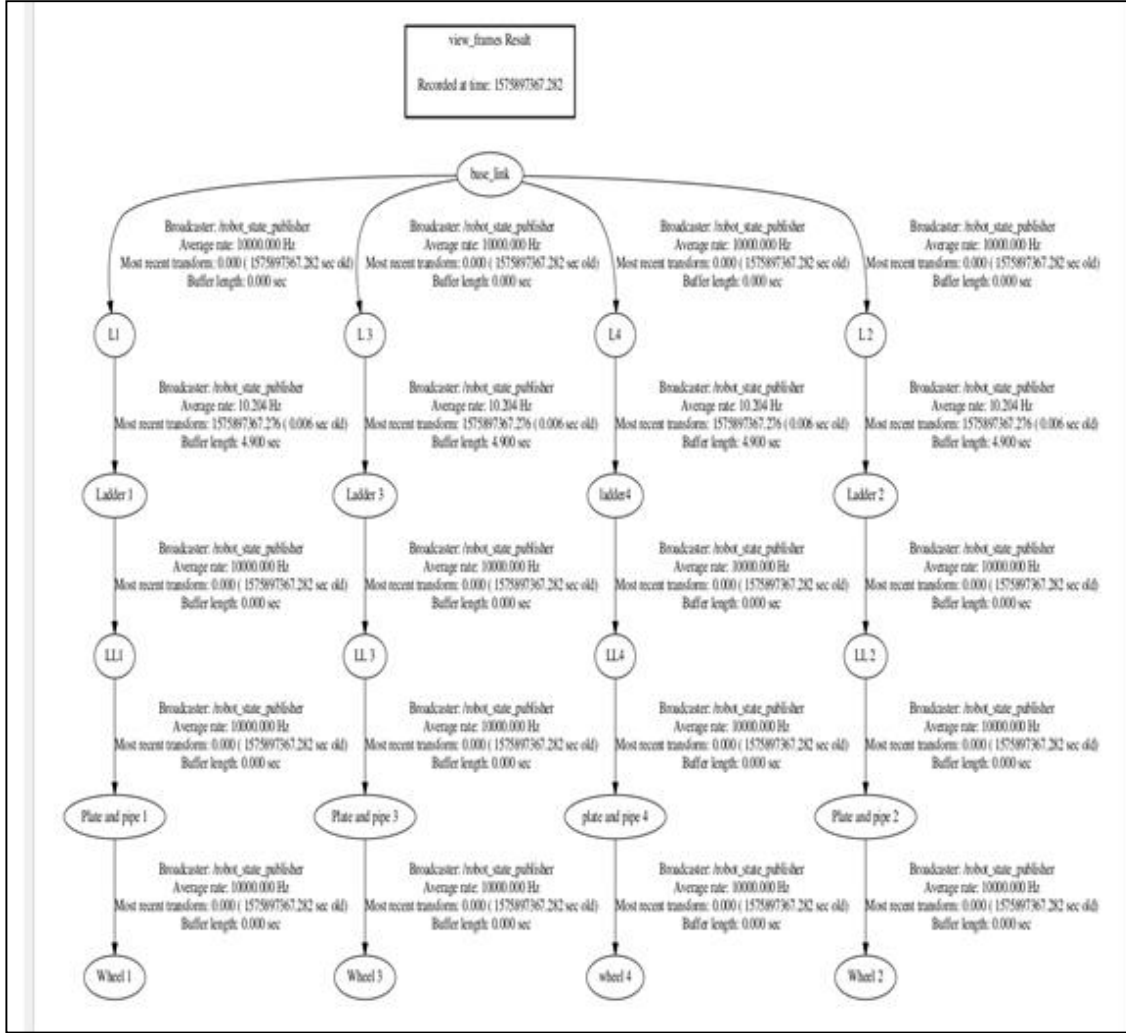


Fig.3.3: Kinematic chain of Robot Generated by URDF [49]

2. If the angular velocities are identical in terms of values and opposite in direction, i.e. if both the wheels are driven in the same speed but in the opposite direction (One clockwise and other anticlockwise) then the robot tends to spin around its vertical axis. This complete turn capability is one of the greatest advantages of a differentially driven robot (i.e. zero radius turn).
3. If the angular velocities are different in terms of values (same or different direction), i.e. if the wheels are driven at different speeds in the same direction or opposite direction, then the robot makes a curve motion. Lastly, if one of the wheels rotate and the other stays still then the robot almost makes a  $90^\circ$  turn.

One of the major disadvantages of this control is that the robot does not drive as expected. It neither drives along a straight line nor turns exactly at expected angles, especially when DC motors are used. This is due to the difference in the number of rotations of each wheel in a given amount of time.

To overcome this problem dual-differential drive can be used which can mechanically guarantee straight line motion. In this approach, each wheel has mechanical differentials and differentials combine the forces from shafts and drive the wheels. In other words, two wheels are connected to two motors where one motor controls the rotation of both wheels while the other controls the direction.

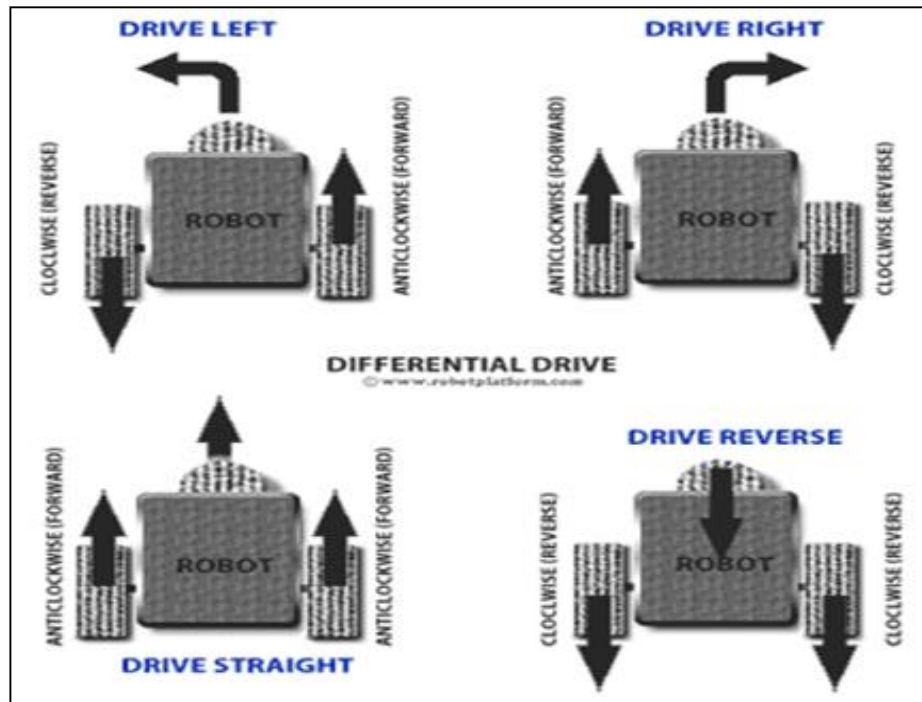


Fig.3.4: Differential Drive System [57]

### **3.6.2 Ackermann Steering System**

The Ackerman driving system mechanically coordinates the angle of two front wheels which are fixed on a common axle used for steering and two rear wheels fixed on another axle for driving. The advantage in this design is increased control, better stability and maneuverability

on road, less slippage and less power consumption. Ackerman steering is designed in such a way that when there is a turn, the inner tire turns with a greater angle than the outer tire and avoids tire slippage. This approach can be generally used for fast outdoor robots which require excellent ground clearance and traction. The downside is additional parts required; no zero radius turn and increased complexity in design and variable wheel tracking is not possible.



Fig.3.5: Ackermann Steering [57]

### **3.6.3 Omnidirectional System**

Omnidirectional robots are built using Omni wheels and/or castors. Since Omni wheels have smaller wheels attached perpendicular to the circumference of another bigger wheel, they

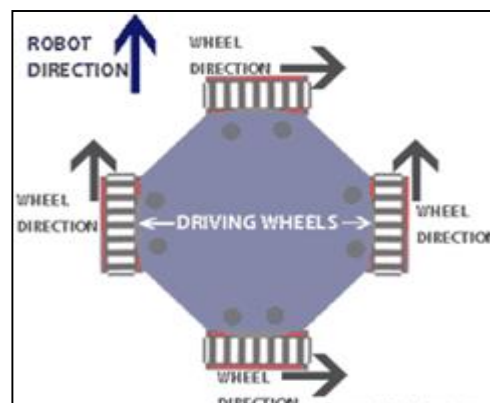


Fig.3.6: Four Wheeled Omni Robot [57]



wheels to move in any direction instantly. The major advantage is that they do not need to rotate or turn to move in any direction unlike other designs. In other words, they are “Holonomic robots” and can move in any direction without changing the orientation.

The drawback of a four-wheeled Omni robot is that it does not balance on irregular terrain and also not all four wheels are guaranteed to stay on the same plane. Apart from this, Omni-wheels are expensive. They are less efficient since not all wheels are fully utilized for driving and controlling the robot. Since, Omni wheels are a combination of many wheels / rollers into one, there is a greater resistance to rotation which leads to greater loss of energy; i.e. loss due to friction. Omni-wheeled robots work on the principle of slippage, position control is difficult.

### **3.7 Remark**

The field terrain would be uneven and sometimes variable wheel tracking and zero radius turn might be required for the robot. So based on these constraints, using a differential drive system will meet the requirements of the robot.

# CHAPTER 4

## MODELLING AND SIMULATION

### 4.1 Introduction

Nowadays, modelling & simulation are the major parts of scientific and engineering processes, especially in industries wherein heavy amount of resource might get wasted due to faulty systems. Modelling and simulation play an important role in the performance analysis and in the development of advanced control algorithms for robotic systems. Design, testing, and validation of robotic systems ranging from indoor mobile robots to outdoor mobile robots, articulated industrial manipulators, underwater robotic systems, and humanoid robots would have never existed without proper modelling and simulation tools. Furthermore, simulation can be used as a tool to develop virtual environments for training operators as well as an educational tool for teaching and learning the basic concepts of robotic systems.

Simulation software	Developer	Physics engine	Supported operating systems	Prog language	CAD files support	API support	ROS support
Webots	Cyberbotics Ltd	Proprietary based on ODE	Linux, Mac OS, Windows	C++	WBT, VRML, X3D	C, C++, Python, Java, Matlab, ROS	Yes
Gazebo	Open Source Robotics Foundation	ODE, Bullet, Simbody, DART	Linux	C++	SDF/URDF, OBJ, STL, Collada	C++	Yes
Actin	Energid Technologies	Proprietary	Windows, Mac OS, Linux, VxWorks, and RTOS-32. (RTX and QNX Planned)	C++	SLDPRT, SLDASM, STEP, OBJ, STL, 3DS, Collada, VRML, URDF, XML, ECD, ECP, ECW, ECX, ECZ,	Not known	Yes
RoboDK	RoboDK	None	Linux, macOS, Windows, Android	Python	STEP, IGES, STL, WRML	C/C++, Python, Matlab	No
Morse	Academic community	Bullet	Linux, BSD*, Mac OS	Python	Unknown	Python	Yes
OpenRAVE	OpenRAVE Community	ODE, Bullet	Linux, Mac OS, Windows	C++, Python	XML, VRML, OBJ, Collada	C/C++, Python, Matlab	Yes
OpenHRP3	AIST	ODE, Internal	Linux, Windows	C++	VRML	C/C++, Python, Java	No
ARGoS	Swarmanoid project	Multiple-physics engines	Linux and Mac OSX	C++	Does not support	C++	Yes
V-REP	Coppelia Robotics	ODE, Bullet, Vortex, Newton	Linux, Mac OS, Windows	LUA	OBJ, STL, DXF, 3DS, Collada, URDF	C/C++, Python, Java, Urbi, Matlab/Octave	Yes

Table 4.1: Comparison between general specifications of the selected simulation software [25]

Additionally, simulation provides low cost means of testing and experimentation, and makes controlling disturbances much easier compared with using real robotic systems [26].

Along with the advancement in powerful and affordable computing technologies in the last two decades, numerous proprietary and open source robotic modelling and simulation software have been developed for robotics application. Examples of such software are Open Dynamics Engine (ODE), Robotic Toolbox for MATLAB, Microsoft Robotics Developer Studio (MRDS), Webots, Virtual Robot Experimentation Platform V-Rep and Gazebo [25]. In Table 4.1 a comparison of different software has been provided.

## **4.2 Gazebo & ROS**

Gazebo is a multi-robot simulation tool which has the capability of accurate and efficient simulation of a population of robots, sensors and objects in a 3-dimensional world. Gazebo [51] generates realistic sensor feedback and has a robust physics engine to generate interactions between objects, robots and environment through URDF scripts. Furthermore, Gazebo provides high-quality graphics, and suitable programmatic and graphical user interfaces. Gazebo is offered freely as a stand-alone software, but has also been packaged along with Robot Operating System (ROS) as the simulation tool. ROS was originally developed by the Stanford Artificial Intelligence Laboratory in support of the Stanford AI Robot (STAIR) project. ROS is an open source robotic middleware that provides libraries and tools to help software developers produce robot programs. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing system, package management, and more user-friendly interfaces.

Along with these points, Gazebo fails at several points. While it can import 3D meshes, there are no editing options, which makes it difficult to alter and optimize models. Moreover, Gazebo interface has a number of issues and fails to follow established conventions. Several difficulties were also noted when installing dependencies for Gazebo and for many of its third-party models. It needs high power graphics in the computer too. While not necessarily severe by themselves, these issues together could have a negative impact on a research project. We have experimented our design through Gazebo simulator because of its wide acceptance in the industries and better open-source support.

## **4.3 Simulation Environment**

The Gazebo simulator has provided dynamics simulation by considering gravity, friction, and contact forces provided by the included physics engines Open Dynamics Engine – ODE. Different types of Gazebo plugins have been enabled to develop control interfaces and sensing systems for pothole detection bot. Further sections are divided as:

1. World environment model description
2. Physical model description
3. ROS/GAZEBO plugins to model the sensors and control/hardware interfaces.

Fig. 4.1 shows the general structure and the required components in Gazebo to model a robotic system.

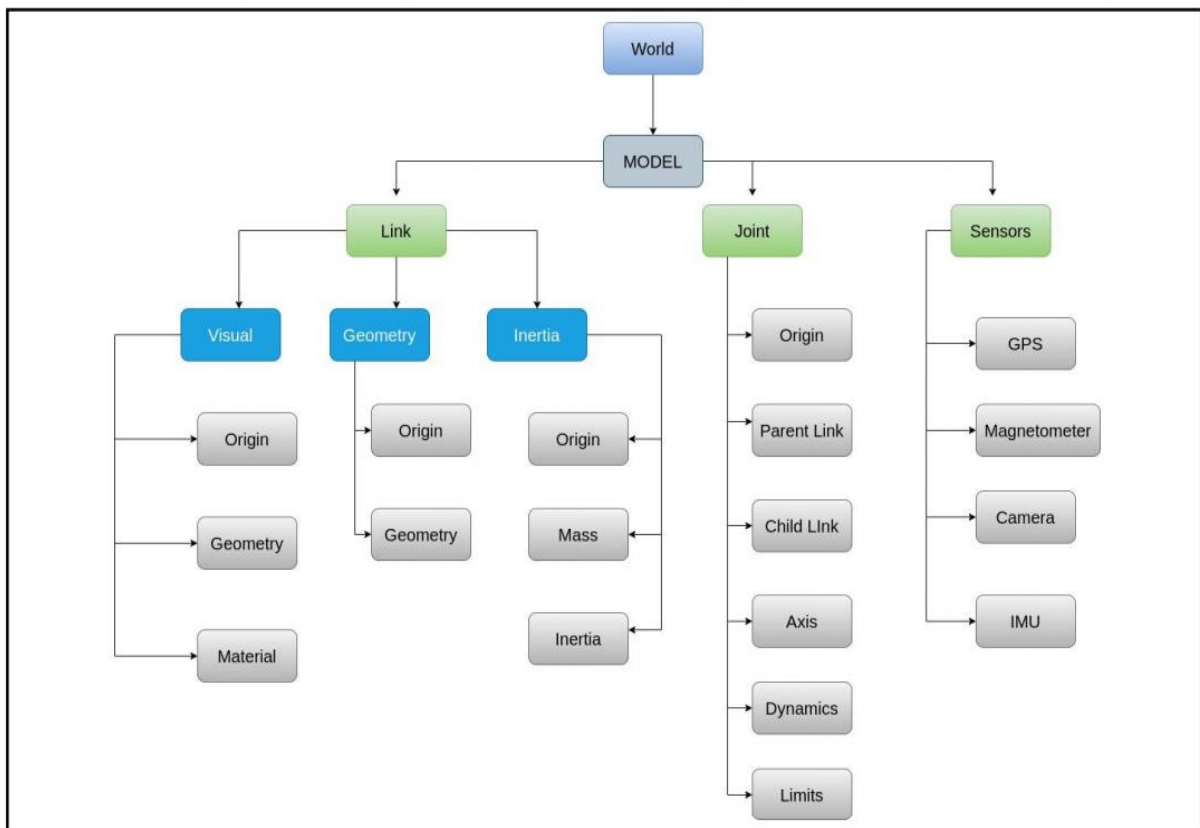


Fig.4.1: General structure and the required components in Gazebo to model a robotic system [25]

### **4.4.1 World Description**

World is a general term to describe objects, global parameters [28] and physics properties. By default, a world is defined by Gazebo with default required parameters. The objects in the world can be static or dynamic. Static objects such as building, lights or walls are defined by their visual and collision geometry. Dynamic objects such as robots are defined not only by visual and collision geometry but also by their inertia information. Objects can be created using standard geometric shapes, or inserted from the model database, or created and by any 3D modelling tools and imported to the Gazebo simulator environment. Scripts can also be added in sdf format for advance modelling. Fig.4.2 shows a simulated environment in Gazebo with the set physics parameters such as gravity. These objects were created using the sdf scripts and models were integrated from GAZEBO library.

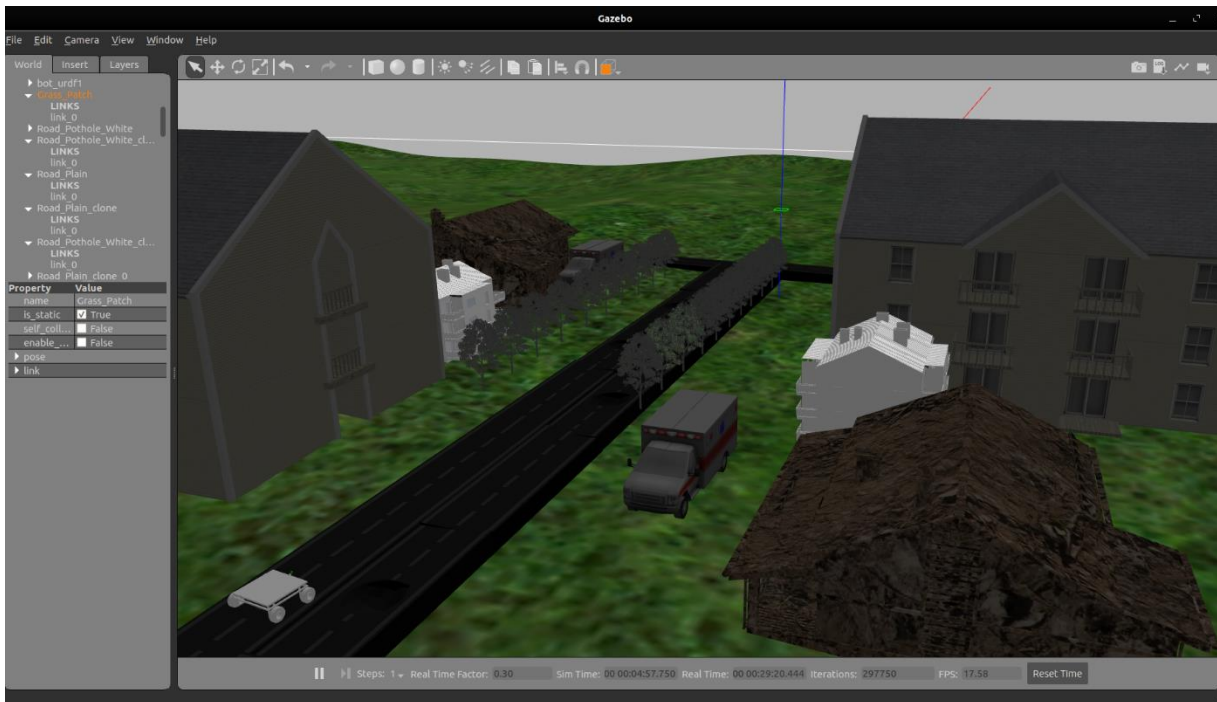


Fig.4.2: A simulated environment in Gazebo

### **4.4.2 Physical Model Description**

A robot, as a dynamic object in the world, consists of links that are connected to each other by joints. A link in Gazebo describes the kinematic and dynamic properties of a physical link in the form of visual/collision geometry and inertia information. A joint models kinematic and dynamic properties of a joint such as joint type, motion axes, and joint safety limits. All this

information is described in the Universal Robotic Description Format - URDF file format so that we can frame design constraints. URDF is an XML file format used by Gazebo and ROS to model all the components of a robot. To model pothole detection bot as a 4WDS platform, 10 links and 9 joints need to be defined as described earlier. The base\_link describes the chassis and four links are connected to it by four fixed joints as driving links. Each of the driving links are connected to a wheel by a continuous joint through motor. Each sensor also should be defined as a physical link attached with a fixed type joint to the base\_link. With no attached sensor. A virtual link called base\_footprint is also needed by some of the of the ROS third party software such as navigation which is placed on the ground and it is sized as the footprint of the platform.

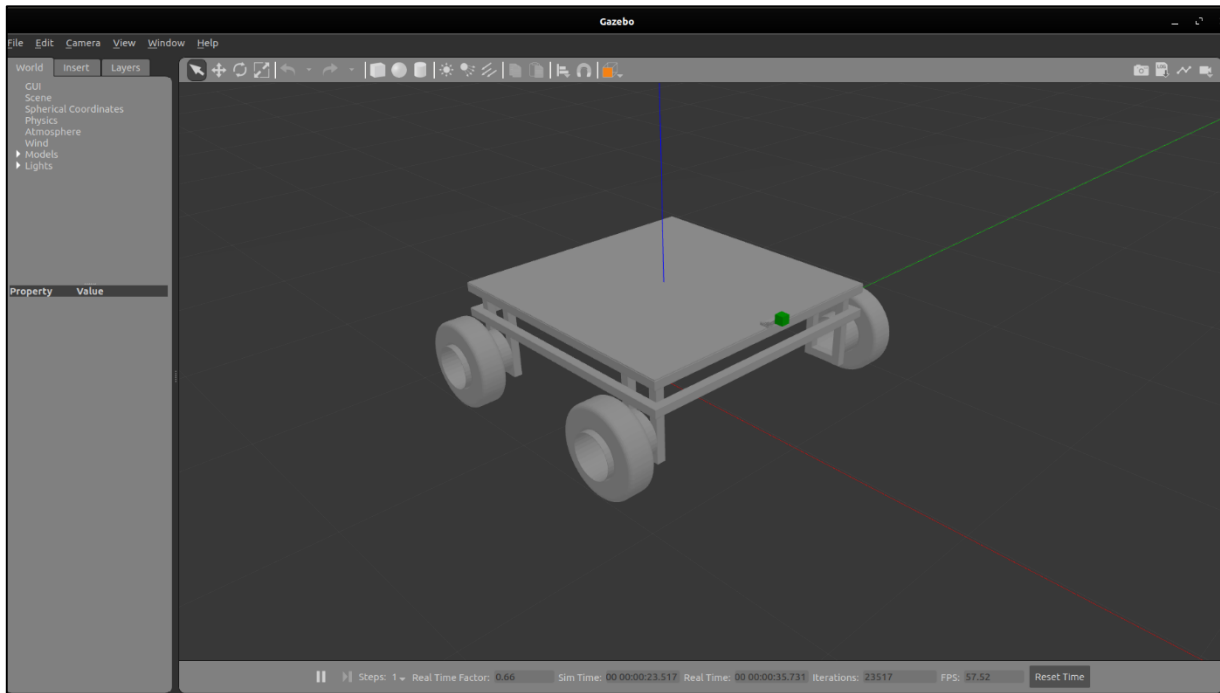


Fig.4.3: Pothole Detection Bot in Gazebo

### **4.4.3 Control Plugin**

To be able to control the motion of each joint in one degree of freedom, a Gazebo control plugin is required. Gazebo also needs to be interfaced with a robot middleware such as ROS to control each joint. A meta package called gazebo\_ros\_pkgs [27] provides a set of ROS packages to interface with Gazebo. A set of packages called ros\_control provide controllers,

hardware interfaces and toolboxes to control joint actuators. Common controllers such as effort, position and velocity are provided by `ros_control`. These controllers (typically PID type) take joint states and set points as inputs and output effort, position or velocity. Each joint is interfaced with the relevant controller by the relevant hardware interface. The same developed software control system is used to control the physical robot, so that the control messages are written to the physical speed/position controllers to actuate control the actuation of the joints. The speed and position feedbacks are read from the encoders back to the control system. By launching the ROS package containing the URDF file of “Pothole detection bot”, the simulated model is opened in the virtual world of Gazebo. This also loads the control interface and waits for all the controllers to be loaded.

The transmission tag in the URDF file defines the type of command interface and the relationship between the joint and actuator. Controller manager provides the infrastructure to load, start, stop and unload the controllers in a real-time manner. A YAML configuration file is also needed which includes information of controllers such as joint controller type and parameters for PID. Hardware\_interface provides position, velocity and effort interfaces between ROS and Gazebo. Our system has 5 controllers of which one provides joint states, four are effort controllers to control each of the driving joints with required torque.

## **4.5 Vision Camera**

Vision camera is a vision system to simulate the real physical world. Camera gives the ability to generate 3D images. Typically, a camera that is placed at the center of robot to capture images of field. Through several pre-processing steps, images [50] are being converted into video and DL model has been implemented to classify between pothole and plain road. All the required parameters such as frame rate, image width and height, output format, base line distance and noise parameters can be defined within Gazebo plugin through URDF. Fig.4.5 shows the visualization of simulated camera by showing the image of an environment in Gazebo. Parameters for plugin has adopted from hector project to model the simulated camera.



Fig.4.4: Visualization of simulated camera by showing the image of an environment in Gazebo



# CHAPTER 5

## DETECTION MODULE

### 5.1 Introduction and Related Works

Deep Learning has been really remarkable in the field of detection, some works related to detection of potholes have been done. Earlier a low cost model for analyzing 3D pavement images was proposed, which utilizes a low cost Kinect sensor which gives the direct depth measurements, thereby reducing computing costs. Lin and Liu have proposed a method for pothole detection based on SVM (Support Vector Machine). This method distinguishes potholes from other defects such as cracks [32]. The images are segmented by using partial differential equations. In order to detect potholes, the method trains the SVM with a set of pavement images. However, the training model fails to detect the pavement defects if the images are not properly illuminated.

### 5.2 Proposed Approach

In the previous approach we tried the InceptionV3 model with some tweaking in the last layers and addition of some layers to the model. This time we tried something very different from previous and not following any previously written papers.

```
Model: "sequential_1"
-----
Layer (type)                Output Shape              Param #
-----
resnet50 (Model)            (None, 2048)              23587712
dropout_1 (Dropout)         (None, 2048)              0
dense_1 (Dense)             (None, 2048)              4196352
dense_2 (Dense)             (None, 1024)              2098176
dense_3 (Dense)             (None, 512)               524800
dense_4 (Dense)             (None, 2)                 1026
-----
Total params: 30,408,066
Trainable params: 30,354,946
Non-trainable params: 53,120
-----
```

Fig.5.1 Model Architecture [36]

Trying out ResNet50 and ResNet101 with some tweaks in the last layers of the model.

Before moving to some stats or results obtained after applying our custom model to the problem, let's first dive more into the model structure of ResNet and discover some key features that set it out from other types of architectures.

## **5.3 ResNet**

### **5.3.1 Background**

Researchers observed that it makes sense to affirm that “*the deeper the better*” when it comes to convolutional neural networks. This makes sense, since the models should be more capable (their flexibility to adapt to any space increase because they have a bigger parameter space [37] to explore). However, it has been noticed that after some depth, the performance degrades.

This was one of the bottlenecks of VGG. They couldn't go as deep as `wanted, because they started to lose generalization capability.

### **5.3.2 Motivation**

Since neural networks are good function approximators [33], they should be able to easily solve the identify function, where the output of a function becomes the input itself.

$$f(x) = x \tag{1}$$

Following the same logic, if we bypass the input to the first layer of the model to be the output of the last layer of the model, the network should be able to predict whatever function it was learning before with the input added to it. The intuition is that learning  $f(x) = 0$  has to be easy for the network.

$$f(x) + x = h(x) \tag{2}$$

### **5.3.3 What problems ResNets solve?**

One of the problems ResNets solve is the famous vanishing gradient. This is because when the network is too deep, the gradients from where the loss function is calculated easily shrink to

zero after several applications of the chain rule. This result on the weights never updating its values and therefore, no learning is being performed.

With ResNets, the gradients can flow directly through the skip connections backwards from later layers to initial filters.

### 5.3.4 Architecture

Since ResNets can have variable sizes, depending on how big each of the layers of the model are, and how many layers it has, we will follow the described by the authors in the paper [39] — ResNet 34 — in order to explain the structure after these networks. If you have taken a look at the paper, you will have probably seen some figures and tables like the following ones, that you are struggling to follow. Let's depict those figures by going into the detail of every step.

In here we can see that the ResNet (the one on the right) consists on one convolution and pooling step (on orange) followed by 4 layers of similar behavior.

Each of the layers follow the same pattern. They perform 3x3 convolution with a fixed feature map dimension (F) [64, 128, 256, 512] respectively, bypassing the input every 2 convolutions. Furthermore, the width (W) and height (H) dimensions remain constant during the entire layer.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10 <sup>9</sup>	3.6×10 <sup>9</sup>	3.8×10 <sup>9</sup>	7.6×10 <sup>9</sup>	11.3×10 <sup>9</sup>

Table 5.1: Parameter Table [33]

The dotted line is there, precisely because there has been a change in the dimension of the input volume (of course a reduction because of the convolution). Note that this reduction

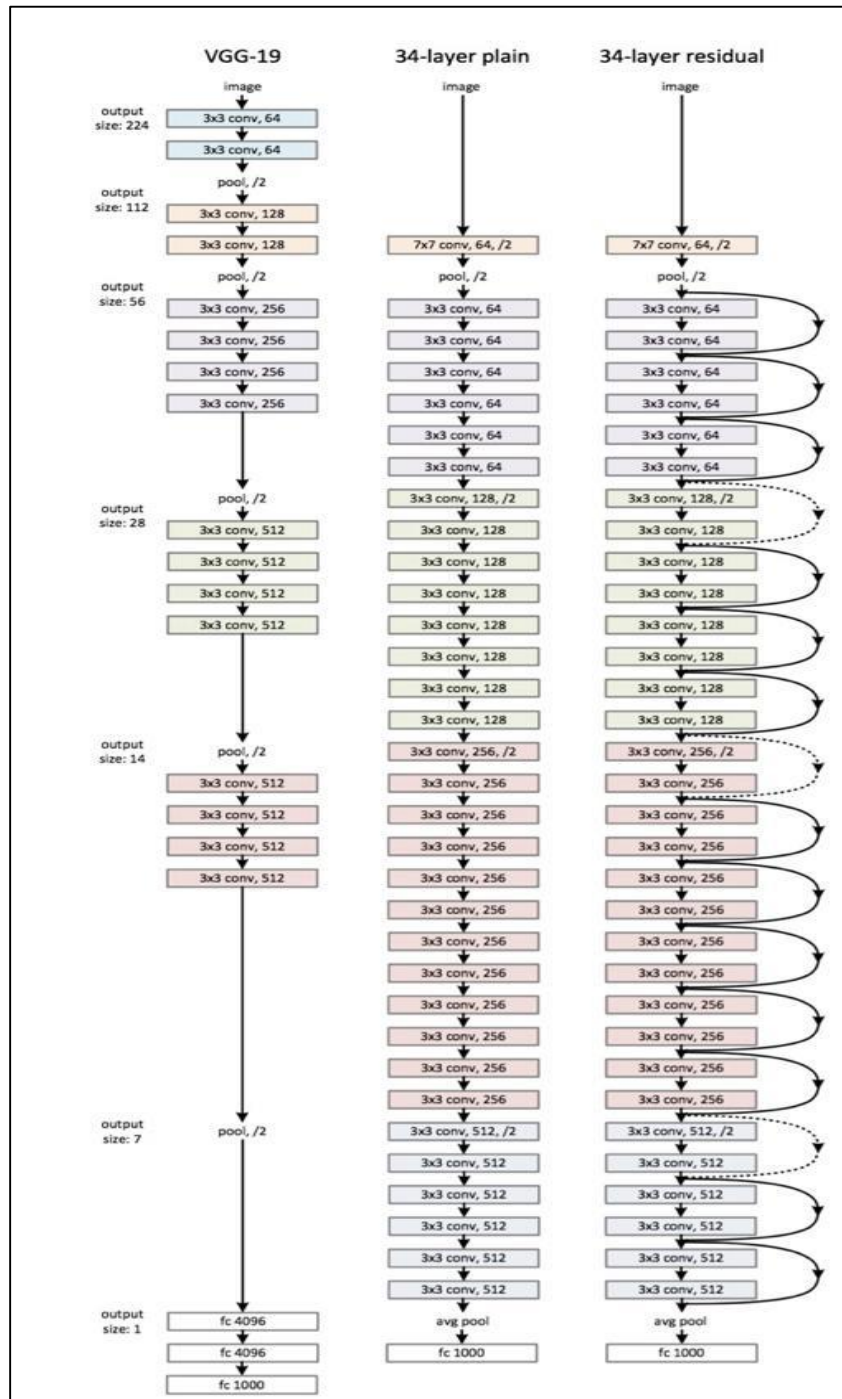


Fig.5.2 Comparison of ReSNet (plain and residual) vs VGG [33]

between layers is achieved by an increase on the stride, from 1 to 2, at the first convolution of each layer; instead of by a pooling [40] operation, which we are used to see as down samplers.

In the Table 5.1, there is a summary of the output size at every layer and the dimension of the convolutional kernels at every point in the structure.

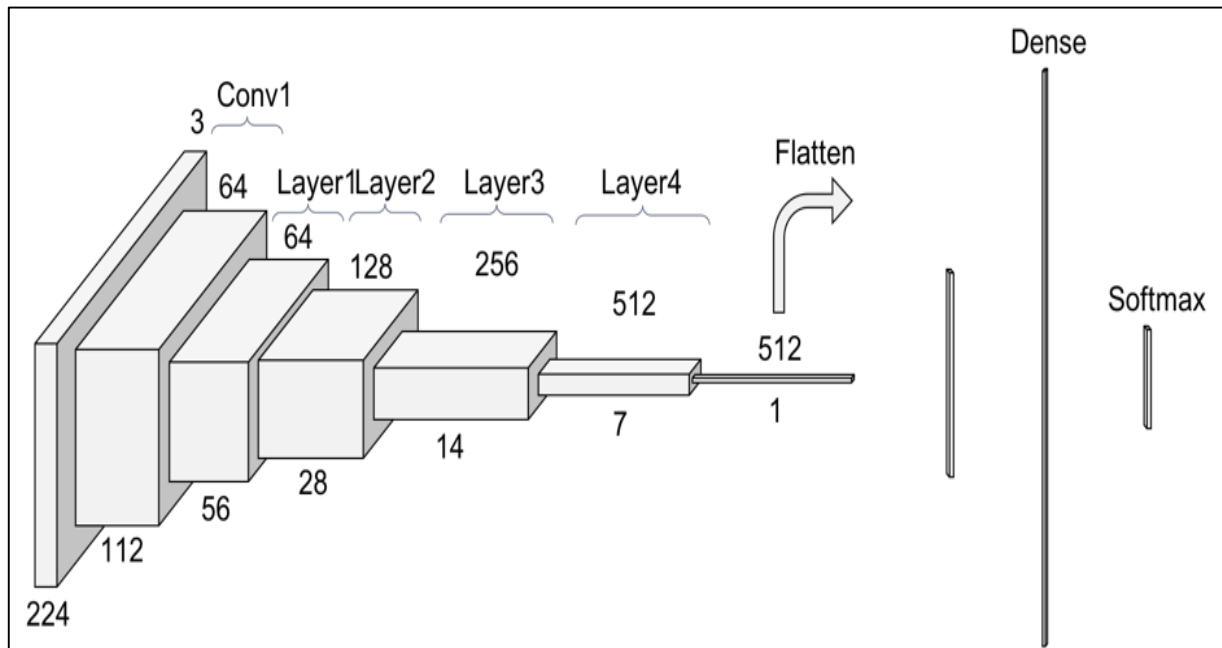


Fig.5.3: Another look at ResNet 34 [40]

### 5.3.5 Convolution

The first step on the ResNet before entering the common layer behavior is a block — called here Conv1 — consisting of a convolution + batch normalization + max pooling operation.

So, first there is a convolution operation. In Fig.5.1 we can see that they use a kernel size of 7, and a feature map size of 64. We need to infer that they have padded [44] with zeros 3 times on each dimension — and check it on the PyTorch documentation. Taking this into account, it can be seen in Fig.5.4 that the output size of that operation will be a (112x122) volume. Since each convolution filter (of the 64) is providing one channel in the output volume, we end up with a (112x112x64) output volume — note this is free of the batch dimension to simplify the explanation.

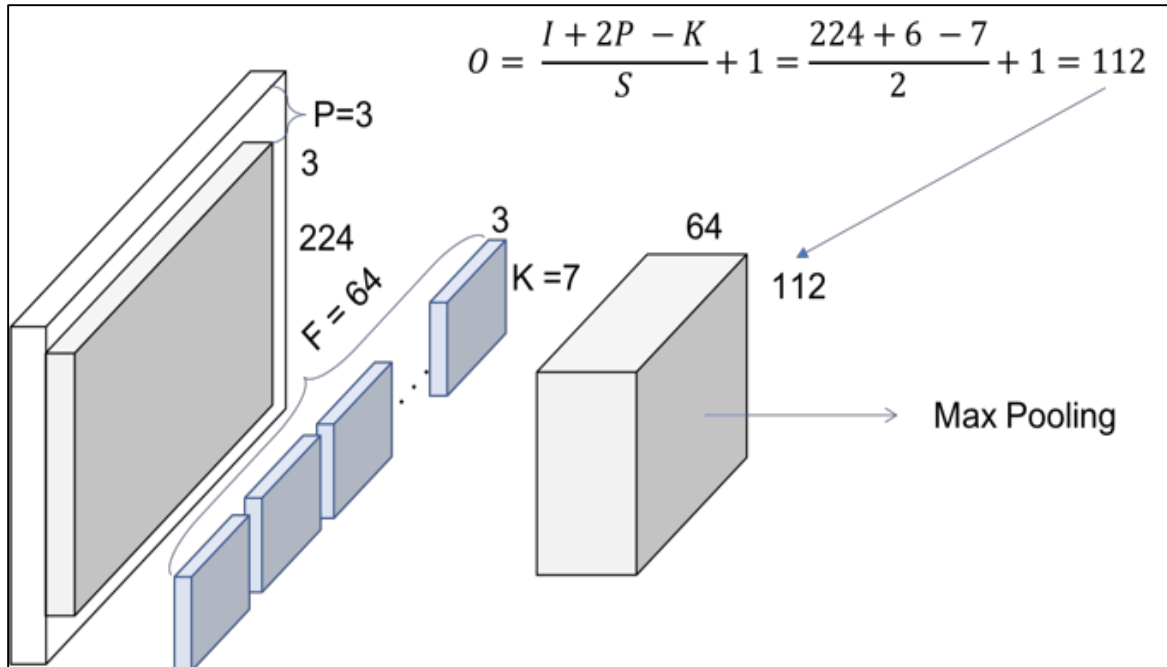


Fig.5.4: Conv1 — Convolution [40]

The next step is the batch normalization, which is an element-wise operation and therefore, it does not change the size of our volume. Finally, we have the (3x3) Max Pooling operation with a stride of 2. We can also infer that they first pad the input volume, so the final volume has the desired dimensions.

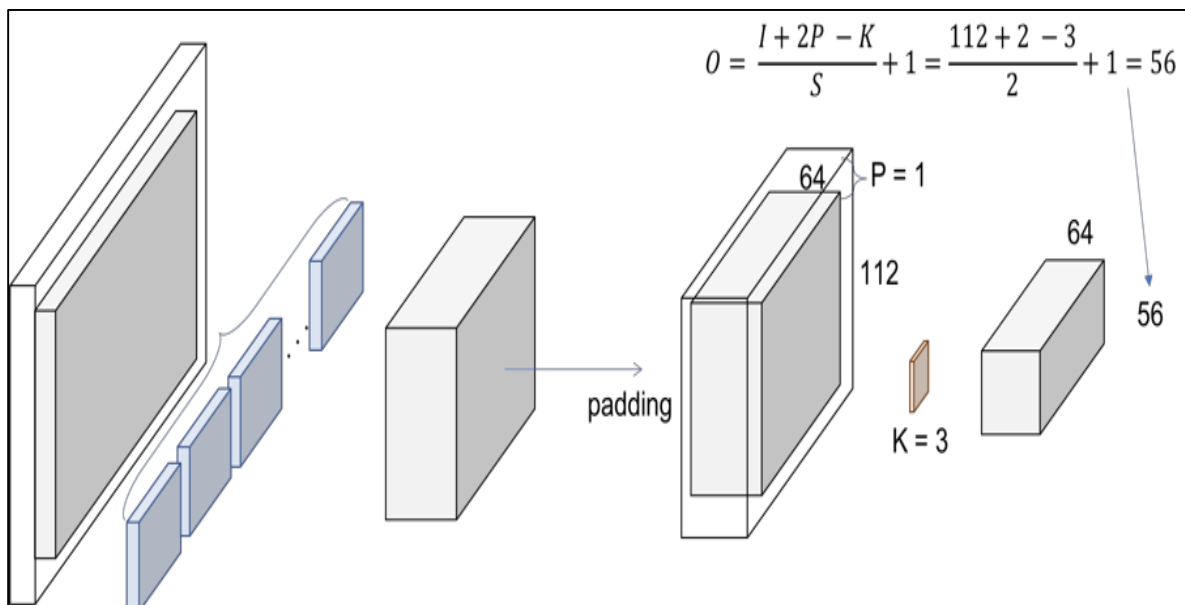


Fig.5.5: Conv1 — Max Pooling [40]

### 5.3.6 ResNet Layers

So, let's explain this repeating name, block. Every layer of a ResNet is composed of several blocks. This is because when ResNets go deeper, they normally do it by increasing the number of operations within a block, but the number of total layers remains the same — 4. An operation here refers to a convolution of a batch normalization and a ReLU activation to an input, except the last operation of a block, that does not have the ReLU.

Therefore, in the PyTorch implementation they distinguish between the blocks that include 2 operations — Basic Block — and the blocks that include [42] 3 operations — Bottleneck Block. Note that normally each of these operations is called layer, but we are using layer already for a group of blocks.

We are facing a basic one now. The input volume is the last output volume from Conv1. Fig.5.6 shows what is happening inside this block.

We are replicating the simplified operation for every layer on the paper.

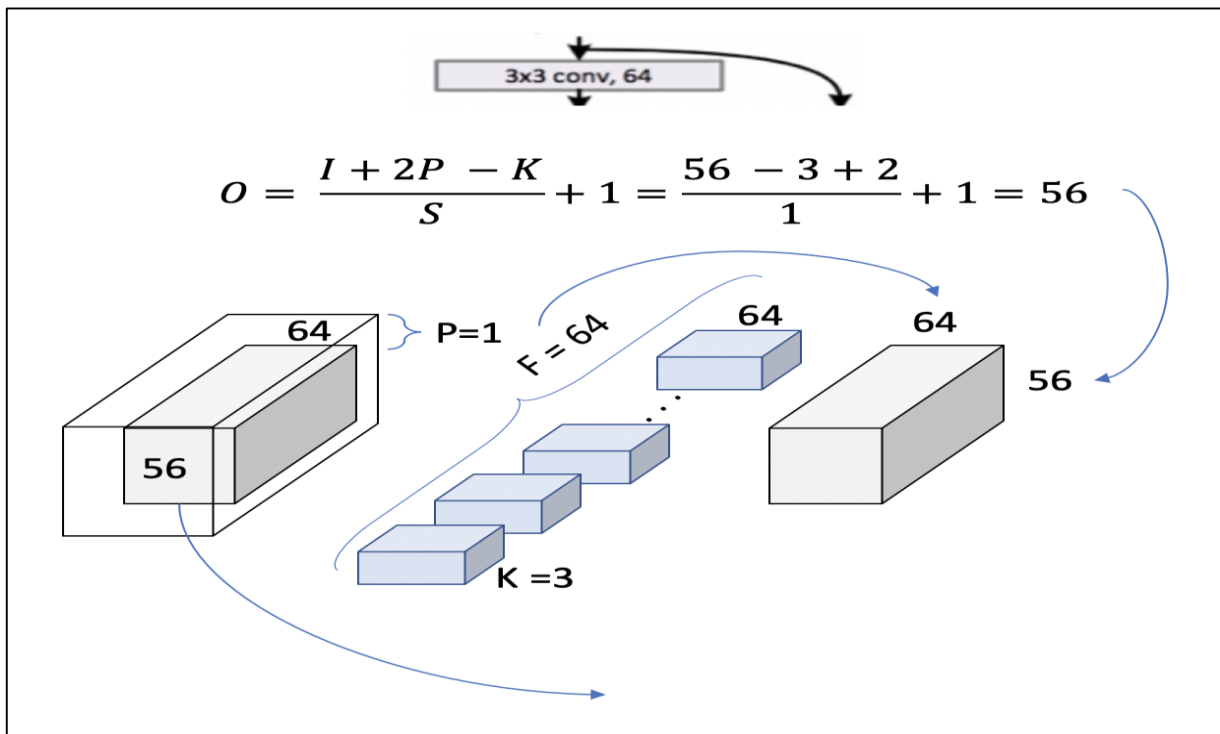


Fig.5.6: Layer 1, block 1, operation 1 [40]

We can double check now in the table from the paper we are using  $[3 \times 3, 64]$  kernel and the output size is  $[56 \times 56]$ . We can see how, as we mentioned previously, the size of the volume does not change within a block. This is because a padding = 1 is used and a stride of also 1. Let's see how this extends to an entire block, to cover the 2  $[3 \times 3, 64]$  that appears in the table.

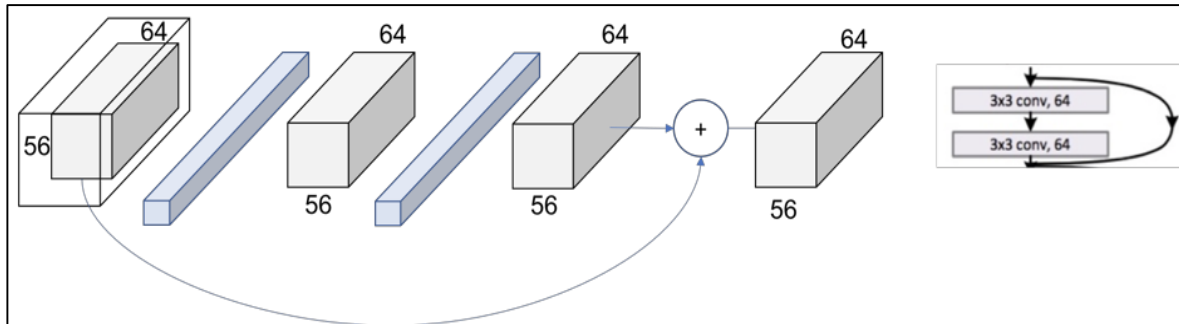


Fig.5.7: Layer 1, block 1 [40]

The same procedure can be expanded to the entire layer then as in Fig.5.7. Now, we can completely read the whole cell of the table (just recap we are in the 34 layers ResNet at Conv2\_x layer).

We can see how we have the  $[3 \times 3, 64] \times 3$  times within the layer.

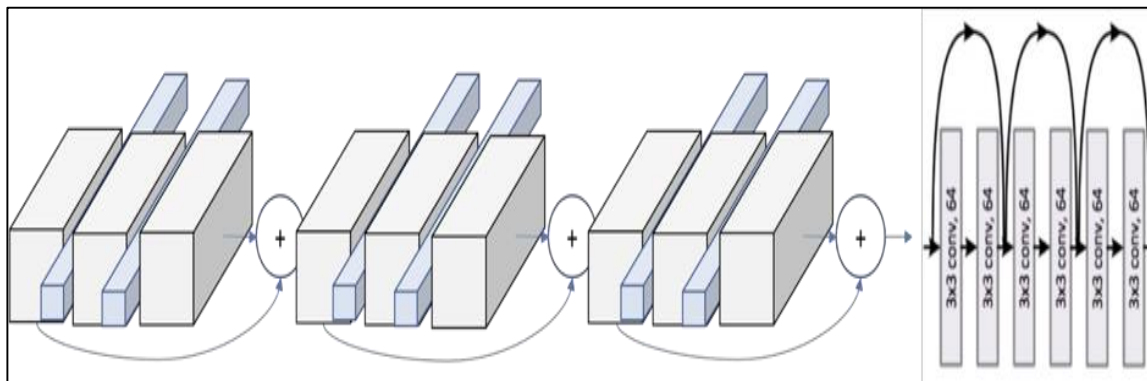


Fig.5.8: Layer 1 [40]

The next step is to escalate from the entire block to the entire layer. In Fig.5.1 we can see how the layers are differentiable by colors. However, if we look at the first operation of each layer, we see that the stride used at that first one is 2, instead of 1 like for the rest of them.

This means that the down sampling of the volume though the network is achieved by increasing the stride instead of a pooling operation like normally CNNs do. In fact, only one



max pooling operation is performed in our Conv1 layer, and one average pooling layer at the end of the ResNet, right before the fully connected dense layer in Fig.5.1.

We can also see another repeating pattern over the layers of the ResNet, the dot layer representing the change of the dimensionality. This agrees with what we just said. The first operation of each layer is reducing the dimension, so we also need to resize the volume that goes through the skip connection, so we could add them like we did in Fig.5.6.

This difference on the skip connections are the so called in the paper as Identity Shortcut and Projection Shortcut. The identity shortcut is the one we have already discussed, simply bypassing the input volume to the addition operator. The projection shortcut performs a convolution operation to ensure the volumes at this addition operation are the same size. From the paper we can see that there are 2 options for matching the output size. Either padding the input volume or perform 1x1 convolutions. Here, this second option is shown.

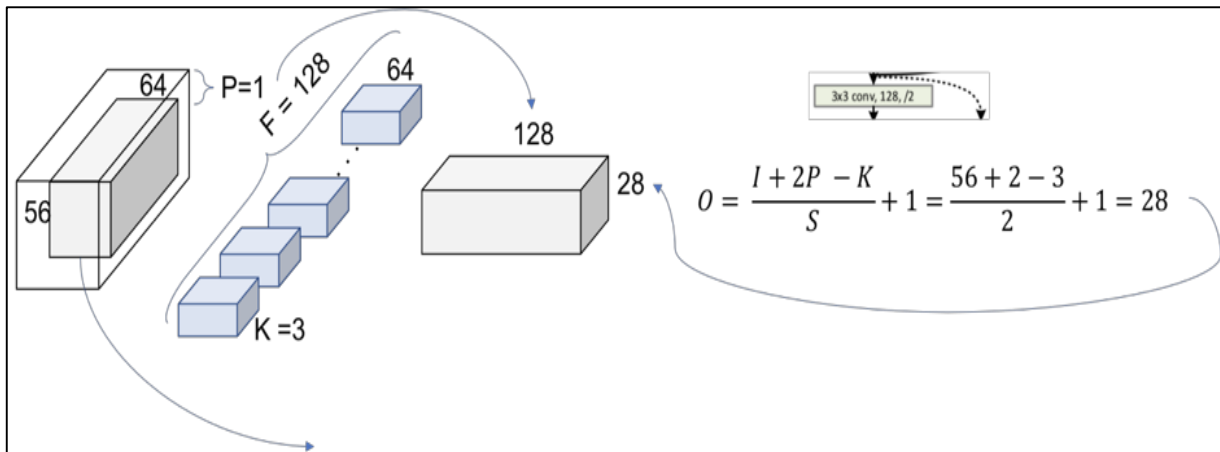


Fig.5.9: Layer2, Block 1, operation 1 [43]

The number of filters is duplicated in an attempt to preserve the time complexity for every operation ( $56*64 = 28*128$ ). Also, note that now the addition operation [45] cannot be performed since the volume got modified. In the shortcut we need to apply one of our down sampling strategies. The 1x1 convolution approach is shown in Fig.5.9.

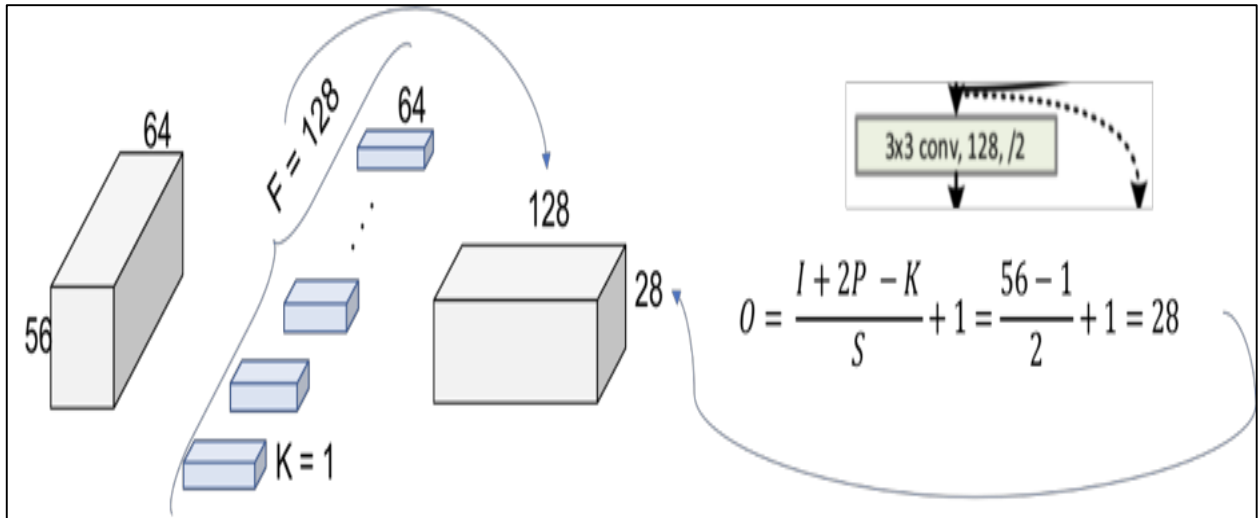


Fig.5.10: Projection Shortcut [40]

The final picture looks then like in Fig.5.10 where now the 2 output volumes of each thread has the same size and can be added.

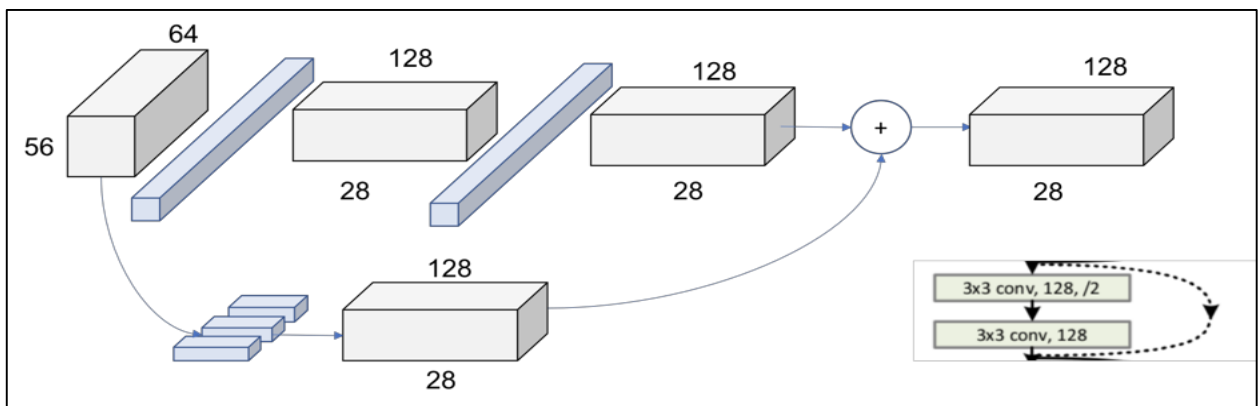


Fig.5.11: Layer 2, Block 1 [40]

The behaviour is exactly the same for the following layers 3 and 4, changing only the dimensions of the incoming volumes.

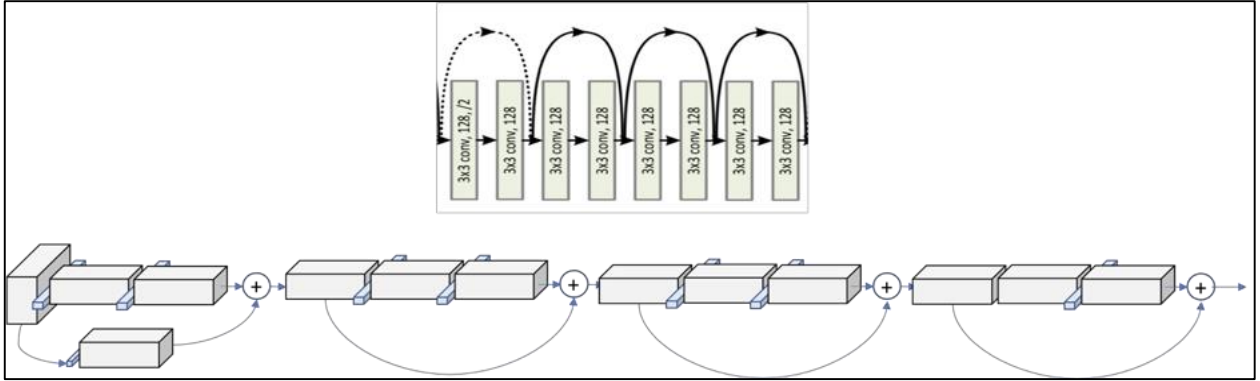


Fig.5.12: Layer 2 [40]

Number of Layers	Number of Parameters
ResNet 18	11.174M
ResNet 34	21.282M
ResNet 50	23.521M
ResNet 101	42.513M
ResNet 152	58.157M

Table 5.2: ResNet architectures for ImageNet [33]

## 5.4. Comparison:

Model	Epochs	Training Accuracy	Testing/Validation Accuracy
ResNet34	50	98%	95%-97%
InceptionV3	50	95%	89%

Table 5.3: Comparison between the ResNet34 and InceptionV3

# CHAPTER 6

## RESULTS

This section provides the experimental results and performance of our robot through ROS third party software and custom open source tools. For some scenarios we have given input velocity from the keyboard to ensure functionality of the robot. The results from the experiment were recorded in a log file for further visualization with live performance.

### 6.1 Teleoperation

As from the below Fig 6.1, it can be seen that it is able to navigate through the road through remote or teleoperation with the help of camera vision.

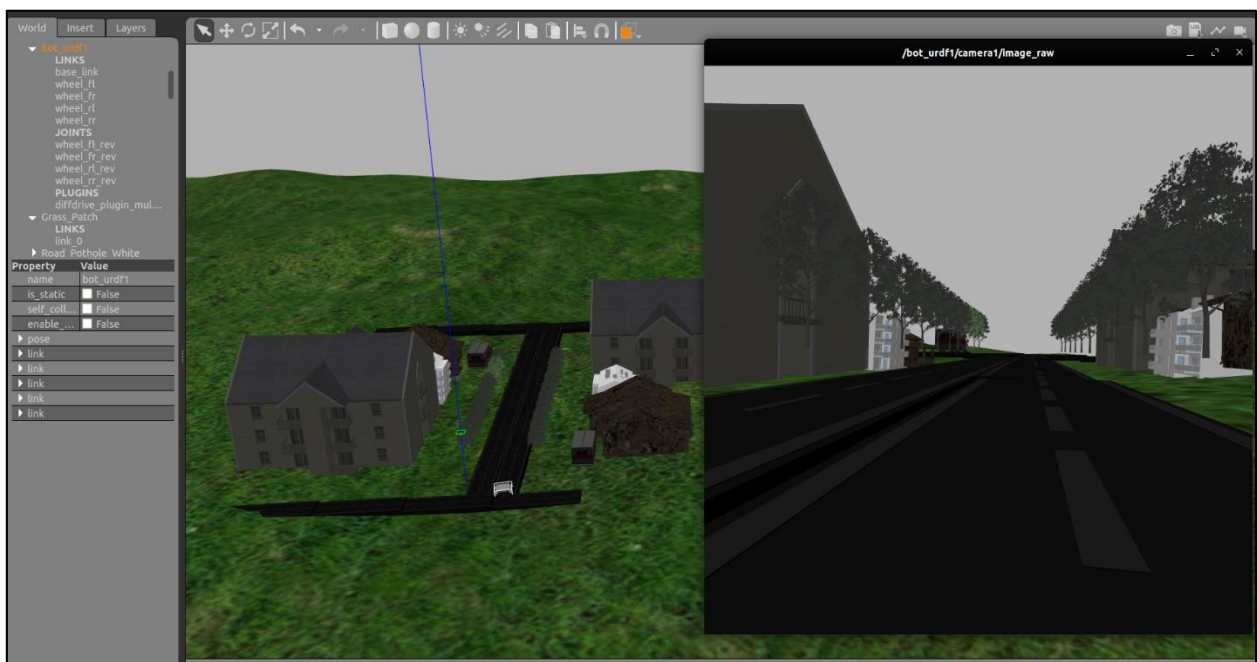


Fig.6.1: Road Traversing through Teleoperation

### 6.2 Model Accuracy & Loss:

The below Fig.6.2 shows the model accuracy for both training as well as validation. This validates the nature of the CNN Layers learning intricate features as we do the training and

saturate over once it has been achieved. We did 50 epochs with ResNet34 and got this output validation and shows that the validation is nearly 95-97% accurate. The model precisely contains the base as the ResNet model and 3-4 Dense Layers after the Base model.

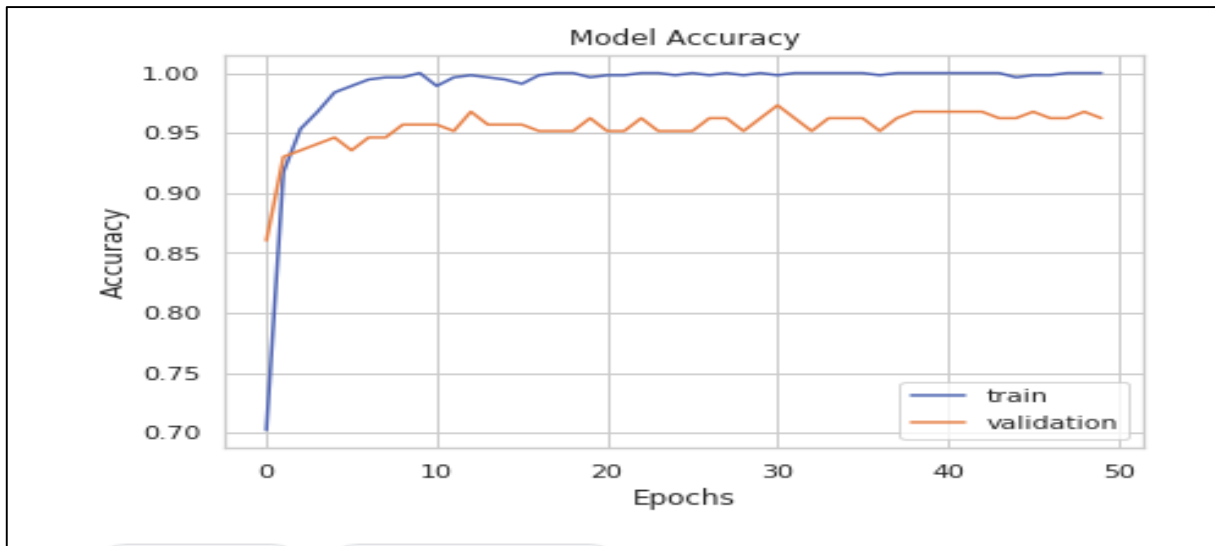


Fig.6.2: Model Accuracy

Fig.6.3 shows the model loss for both training as well as validation. This validates lower loss as more training is done and also the saturation once peak accuracy has been achieved.

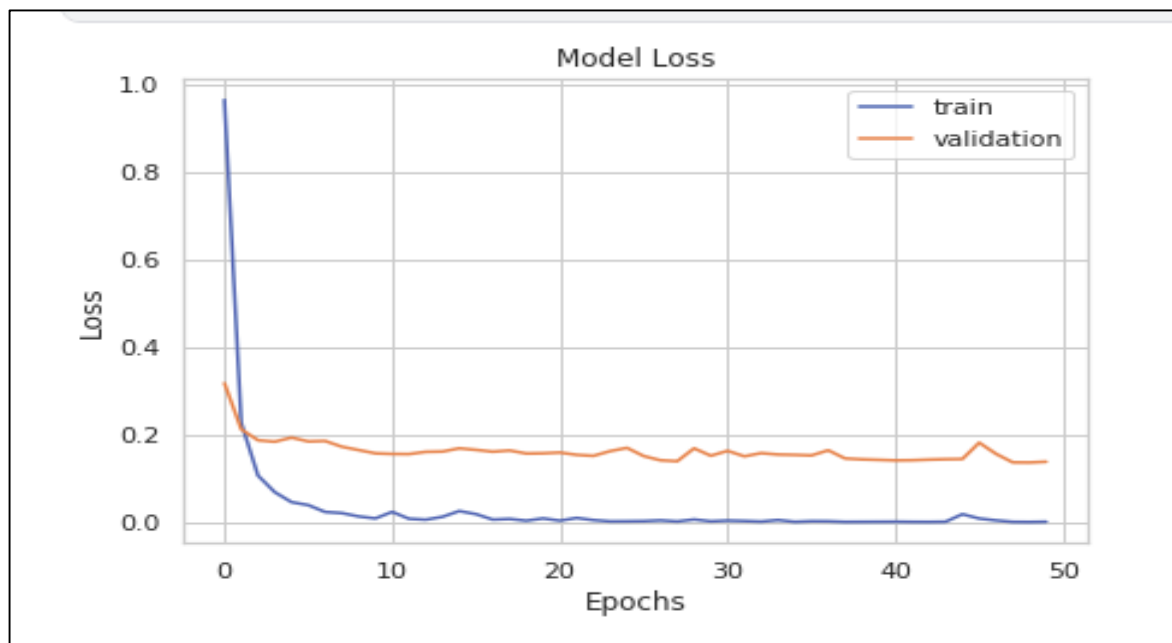


Fig.6.3: Model Loss

## 6.3 Prediction:





Fig.6.4: Prediction on test images

From the Fig.6.4, it can be concluded that the model has now correctly learned to recognize the potholes. The accuracy is approximately 97% at time of testing and is calculated using “Mean Squared Error”.

## **CHAPTER 7**

### **SUMMARY AND CONCLUSION**

This project presents the idea of the making of a robot which is used to detect the potholes on roads. It begins with a glimpse of the current situation of Indian roads and how robotic techniques can be utilized for detecting potholes. As a result of the pandemic, the whole project is shifted to the virtual world of gazebo where it is correctly analyzed with the help of its physics engine (ODE). The solidworks model of the differential drive robot is imported into gazebo and interfaced with sensors. The data from different sensors has been collected. A ResNet34 Model was trained with the help of a custom dataset and the model achieves better accuracy than the previous Inception V3 model that was used. Whole of the software system is based on the ROS framework which is helpful in managing packages and simplifies data communication.

The Robot is operated remotely and can be used for different kinds of surveys to collect data about potholes. It is found that the ResNet34 model has achieved an accuracy upto 97%. That is why it was preferred over the InceptionV3. Remote navigation, collection of data and accurate classification of potholes in simulation environment was the goal of this project and it was achieved. This project would be a huge contribution for increasing the economic benefits, continuous management of roads and passenger safety.



## **CHAPTER 8**

### **FUTURE SCOPE**

The future scope of work in this project is vast. It can be programmed to achieve autonomous navigation based on path-planning and image processing techniques, it can be used to identify the depth of the pothole and it can also be facilitated to determine factors like whether it's easily repairable or the what approximate amount of concrete and other materials required to repair it. For autonomous navigation and making the trajectory of the bot smoother, various other techniques such DGPS (Differential GPS), implementation of predictive filters and control techniques can be tested. For minimizing the errors, data from camera can be fused and used for precise navigation of the robot. For the machine vision part, the performance can be greatly improved by providing more images and their corresponding ground truths from different real scenarios to the classification model where the robot will be deployed.

## REFERENCES

- [1] Pothole Damage Costs U.S. Drivers \$3 Billion Annually, <https://www.oregon.aaa.com/2016/02/pothole-damage-costs-u-s-drivers-3-billion-annually/>.(2017)
- [2] Bad roads killed over 10k people in 2015; 3,416 deaths due to potholes. <https://timesofindia.indiatimes.com/india/Bad-roads-killed-over-10k-people-in-2015-3416-deaths-due-topotholes/articleshow/53482615.cms>. (2015)
- [3] List of countries by traffic-related death rate. Wikipedia, [https://en.wikipedia.org/wiki/List\\_of\\_countries\\_by\\_traffic-related\\_death\\_rate](https://en.wikipedia.org/wiki/List_of_countries_by_traffic-related_death_rate). (2018)
- [4] AUSTROADS, 1994. Road Safety Audit. Sydney (2006)
- [5] L. Vogel, C. J. Bester : A Relationship between accident types and causes. (1999)
- [6] Rajeshwari Madli, Santosh Hebbar, Praveenraj Pattar, and Varaprasad Golla : AutomaticDetection and Notification of Potholes and Humps on Roads to Aid Drivers. IEEE SensorJournal. VOL. 15, pp. 4313--4318. (2015)
- [7] I. Moazzam, K. Kamal, S. Mathavan, S. Usman, M. Rahman: Metrology and visualizationof potholes using the Microsoft Kinect sensor. Proc. 16th Int. IEEE Conf. Intell. Transp.Syst, pp. 1284—1291. (2013)
- [8] J. Lin, Y. Liu : Potholes detection based on SVM in the pavement distress image. Proc. 9th Int. Symp. Distrib. Comput. Appl. Bus. Eng. Sci., pp. 544—547. (2010)
- [9] Song, Hyunwoo & Baek, Kihoon & Byun, Yungcheol. (2018). Pothole Detection using Machine Learning. 151-155. 10.14257/astl.2018.150.35.
- [10] A. Tedeschi and F. Benedetto, “A real-time automatic pavement crack and pothole recognition system for mobile Android-based devices,” *Adv.Eng. Inform.*, vol. 32, pp. 11–25, Apr. 2017.
- [11] B.-H. Lin and S.-F. Tseng, “A predictive analysis of citizen hotlines 1999 and traffic accidents: A case study of taoyuan city,” in *Proc. Int. Conf. Big Data Smart Comput.*, Feb. 2017, pp. 374–376.
- [12] D. Santani *et al.*, “Communisense: Crowdsourcing road hazards in nairobi” in *Proc. Int. Conf. Hum.-Comput. Interact. Mobile Devices Services*, Aug. 2015, pp. 445–456.

- [13] D. O'Carroll, "For the love of pizza, Domino's is now fixing potholes in roads," Wellington, New Zealand, Tech. Rep., Jun. 2018. [Online]. Available: <https://stuff.co.nz>
- [14] A. Dhiman, H.-J. Chien, and R. Klette, "Road surface distress detection in disparity space," in *Proc. Int. Conf. Image Vis. Comput. New Zealand*, Dec. 2017, pp. 1–6.
- [15] A. Dhiman, H.-J. Chien, and R. Klette, "A multi-frame stereo vision based road profiling technique for distress analysis," in *Proc. ISPAN*, Oct. 2018, pp. 7–14.
- [16] A. Dhiman, S. Sharma, and R. Klette, *Identification of Road Potholes*. Stratford, U.K.: MIND, 2019.
- [17] A. Mednis, G. Stardins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using Android smartphones with accelerometers," in *Proc. Int. Conf. Distrib. Comput. Sensor Syst. Workshops*, Jun. 2011, pp. 1–6.
- [18] M. Ghadge, D. Pandey, and D. Kalbande, "Machine learning approach for predicting bumps on road," in *Proc. Int. Conf. Appl. Theor. Comput. Commun. Technol.*, Oct. 2015, pp. 481–485.
- [19] F. Seraj, B. J. van der Zwaag, A. Dilo, T. Luarasi, and P. Havinga, "RoADS: A roadpavement monitoring system for anomaly detection using smart phones," in *Proc. Int. Workshop Modeling Social Media*, Jan. 2014, pp. 128–146.
- [20] J. Ren and D. Liu, "PADS: A reliable pothole detection system using machine learning," in *Proc. Int. Conf. Smart Comput. Commun.*, Jan. 2016, pp. 327–338.
- [21] K. Georgieva, C. Koch, and M. König, "Wavelet transform on multi-GPU for real-time pavement distress detection," in *Proc. Comput. Civil Eng.*, May 2015, pp. 99–106.
- [22] K. Doycheva, C. Koch, and M. König, "Implementing textural features on GPUs for improved real-time pavement distress detection," *Real-Time Image Process.*, vol. 33, pp. 1–12, Sep. 2016.
- [23] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *Syst. Man*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973.
- [24] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. San Mateo, CA, USA: Morgan Kaufmann, 2016.

- [25] Mostafa Sharifi, XiaoQi Chen, Christopher Pretty, Don Clucas, Erwan Carbon-Lunel presented “Modelling and simulation of a non-holonomic omnidirectional mobile robot for offline programming and system performance analysis” in *Simulation Modelling Practice and Theory* 87 (2018) 155-169 by ELSEVIER
- [26] Shamshiri R R, Hameed I A, Pitonakova L, Weltzien C, Balasundram S K, Yule I J, Et. al. Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison. *Int J Agric & Biol Eng*, 2018; 11(4): 15–31.
- [27] L. Powell and K. G. Satheeshkumar, “Automated road distress detection,” in *Proc. Int. Conf. Emerg. Technol. Trends*, 2016, pp. 1–6.
- [28] A. Rasheed, K. Kamal, T. Zafar, S. Mathavan, and M. Rahman, “Stabilization of 3D pavement images for pothole metrology using the Kalman filter,” in *Proc. Int. Conf. Intell. Transp. Syst.*, 2015, pp. 2671–2676.
- [29] H. Hirschmüller, “Accurate and efficient stereo processing by semi global matching and mutual information,” in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, pp. 807–814.
- [30] Q. Li, M. Yao, X. Yao, and B. Xu, “A real-time 3D scanning system for pavement distortion inspection,” *Meas. Sci. Technol.*, vol. 21, no. 8, pp. 015702-1–015702-8, 2010.
- [31] X. Yu and E. Salari, “Pavement pothole detection and severity measurement using laser imaging,” in *Proc. Int. Conf. Electro/Inf. Technol.*, 2011, pp. 1–5.
- [32] S. Gidaris and N. Komodakis. Object detection via a multi-region & semantic segmentation-aware cnn model. In *ICCV*, 2015.
- [33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Deep Residual Learning for Image Recognition”.
- [34] Z. Zhang, X. Ai, C. K. Chan, and N. Dahnoun, “An efficient algorithm for pothole detection using stereo vision,” in *Proc. Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 564–568.
- [35] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

- [36] H. Youquan, W. Jian, Q. Hanxing, Z. Wei, and X. Jianfang, “A research of pavement potholes detection based on three-dimensional projection transformation,” in *Proc. Int. Conf. Image Signal Process.*, 2011, pp. 1805–1808.
- [37] C. Zhang and A. Elaksher, “An unmanned aerial vehicle-based imaging system for 3D measurement of unpaved road surface distresses,” *Comput.-Aided Civil Infrastruct. Eng.*, vol. 27, no. 2, pp. 118–129, 2012.
- [38] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. arXiv:1302.4389, 2013.
- [39] T. Naidoo, D. Joubert, T. Chiwewe, A. Tyatyantsi, B. Rancati, and A. Mbizeni, “Visual surveying platform for the automated detection of road surface distresses,” in *Proc. Int. Conf. Sensors MEMS Electro-Optic Syst.*, 2014, Art. no. 92570D.
- [40] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 2014.
- [41] T. Garbowski and T. Gajewski, “Semi-automatic inspection tool of pavement condition from three-dimensional profile scans,” *Intell. Transp. Syst.*, vol. 172, pp. 310–318, Jan. 2017.
- [42] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing coadaptation of feature detectors. arXiv:1207.0580, 2012.
- [43] T. Shen, G. Schamp, and M. Haddad, “Stereo vision based road surface preview,” in *Proc. Int. Conf. Intell. Transp. Syst.*, 2014, pp. 1843–1849.
- [44] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. TPAMI, 33, 2011.
- [45] Y.-H. Tseng, S.-C. Kanga, J.-R. Changb, and C.-H. Leea, “Strategies for autonomous robots to inspect pavement distresses,” *Autom. Construct.*, vol. 20, no. 8, pp. 1156–1172, 2011.
- [46] Z. Ying, G. Li, X. Zang, R. Wang, and W. Wang, “A novel shadow-free feature extractor for real-time road detection,” in *Proc. Int. Conf. Pervas. Ubiquitous Comput.*, 2016, pp. 611–615.
- [47] Song, Hyunwoo & Baek, Kihoon & Byun, Yungcheol. (2018). Pothole Detection using Machine Learning. 151-155. 10.14257/astl.2018.150.35.

- [48] T. Veit, J.-P. Tarel, P. Nicolle, and P. Charbonnier, "Evaluation of road marking feature extraction," in *Proc. Int. Conf. ITSC*, Beijing, China, 2008, pp. 174–181.
- [49] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng, "ROS: an open-source Robot Operating System", *ICRA Workshop on Open-Source Software*, vol. 3, pp. 5, 2009.
- [50] Wang T, Wu Y, Liang J, Han C, Chen J, Zhao Q. "Analysis and experimental kinematics of a skid-steering wheeled robot based on a laser scanner sensor" in *Sensors* (Basel). 2015;15(5):9681-9702. Published 2015 Apr 24. doi:10.3390/s150509681
- [51] V. Pereira, S. Tamura, S. Hayamizu, and H. Fukai, "A deep learning based approach for road pothole detection in timor leste," in *Proc. Int. Conf. Service Oper. Logistics, Informat.*, 2018, pp. 279–284.
- [52] K. E. An, S. W. Lee, S.-K. Ryu, and D. Seo, "Detecting a pothole using deep convolutional neural network models for an adaptive shock observing in a vehicle driving," in *Proc. Int. Conf. Consumer Electron.*, 2018, pp. 1–2.
- [53] Y. Bhatia, R. Rai, V. Gupta, N. Aggarwal, and A. Akula, "Convolutional neural networks based potholes detection using thermal imaging," *King Saud Univ., Comput. Inf. Sci.*, to be published. doi: 10.1016/j.jksuci.2019.02.004.
- [54] A. Mikhailiuk and N. Dahnoun, "Real-time pothole detection on TMS320C6678 DSP," in *Proc. Int. Conf. Imag. Syst. Techn.*, 2016, pp. 123–128.
- [55] Arduino, <https://www.arduino.cc/>
- [56] G. Lin, A. Milan, C. Shen, and I. D. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," *CoRR*, vol. abs/1611.06612, 2016.
- [57] Robot Platform,  
[http://www.robotplatform.com/knowledge/Classification\\_of\\_Robots/wheel\\_control\\_theory.html](http://www.robotplatform.com/knowledge/Classification_of_Robots/wheel_control_theory.html)
- [58] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, 2015, 3431–3440.
- [59] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. CVPR*, 2016, pp. 2818–2826.

- [60] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyaama, and H. Omata, “Road damage detection using deep neural networks with images captured through a smartphone,” 2018, *arXiv:1801.09454*. [Online]. Available: <https://arxiv.org/abs/1801.09454>
- [61] J. Huang *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors,” in *Proc. CVPR*, 2017, pp. 7310–7311.
- [62] A. G. Howard *et al.*, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, *arXiv:1704.04861*. [Online]. Available: <https://arxiv.org/abs/1704.04861>