**CHAPTER 1**

**INTRODUCTION**

* 1. **ABOUT THE PROJECT DOMAIN**
     1. **DEEP LEARNING**

A deep neural network (DNN) is an [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a [linear relationship](https://en.wikipedia.org/wiki/Linear_relationship) or a non-linear relationship. The network moves through the layers calculating the probability of each output. For example, a DNN that is trained to recognize dog breeds will go over the given image and calculate the probability that the dog in the image is a certain breed. The user can review the results and select which probabilities the network should display (above a certain threshold, etc.) and return the proposed label. Each mathematical manipulation as such is considered a layer, and complex DNN have many layers, hence the name "deep" networks.

DNNs can model complex non-linear relationships. DNN architectures generate compositional models where the object is expressed as a layered composition of [primitives](https://en.wikipedia.org/wiki/Primitive_data_type). The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network. Deep architectures include many variants of a few basic approaches. Each architecture has found success in specific domains. It is not always possible to compare the performance of multiple architectures, unless they have been evaluated on the same data sets. DNNs are typically feed forward networks in which data flows from the input layer to the output layer without looping back. At first, the DNN creates a map of virtual neurons and assigns random numerical values, or "weights", to connections between them. The weights and inputs are multiplied and return an output between 0 and 1. If the network did not accurately recognize a particular pattern, an algorithm would adjust the weights. That way the algorithm can make certain parameters more influential, until it determines the correct mathematical manipulation to fully process the data.

[Recurrent neural networks](https://en.wikipedia.org/wiki/Recurrent_neural_networks) (RNNs), in which data can flow in any direction, are used for applications such as [language modeling](https://en.wikipedia.org/wiki/Language_model). Long short-term memory is particularly effective for this use. [Convolutional deep neural networks (CNNs)](https://en.wikipedia.org/wiki/Convolutional_neural_network) are used in computer vision. CNNs also have been applied to [acoustic modeling](https://en.wikipedia.org/wiki/Acoustic_model) for automatic speech recognition (ASR).

Disparity refers to the difference in image location of an object seen by the left and right eyes that motion is the disparity. In a pair of images derived from stereo cameras, you can measure the apparent motion in pixels for every point and make an intensity image out of the measurement.

#### 1.1.2 CHALLENGES

As with ANNs, many issues can arise with naively trained DNNs. Two common issues are [overfitting](https://en.wikipedia.org/wiki/Overfitting) and computation time. DNNs are prone to overfitting because of the added layers of abstraction, which allow them to model rare dependencies in the training data.

* + 1. **OVERCOMING THE CHALLENGES**

DNNs must consider many training parameters, such as the size (number of layers and number of units per layer), the [learning rate](https://en.wikipedia.org/wiki/Learning_rate), and initial weights. [Sweeping through the parameter space](https://en.wikipedia.org/wiki/Hyperparameter_optimization#Grid_search) for optimal parameters may not be feasible due to the cost in time and computational resources. Speeds up computation. Large processing capabilities of many-core architectures have produced significant speedups in training, because of the suitability of such processing architectures for the matrix and vector computations.

**1.1.4**  **PREDICTIVE ANALYSIS**

Data analytics also a pivotal role in almost all the fields, because of the number of data keep generating at every second, it is essential to perform data analytics so that the data could be analyzed and can be maintained. It also helps in getting the information in a much more efficient manner or in the way we wanted to. Actually in our project we tend to perform Predictive data analytics so that with the Deep Learning being bounded, our goal to promote the pothole detection with the help of trained data by using CNN (Convolution Neural Network).

Predictive analytics is the branch of [data mining](http://searchsqlserver.techtarget.com/definition/data-mining) concerned with the prediction of future probabilities and trends. The central element of predictive analytics is the predictor, a variable that can be measured for an individual or other entity to predict future behavior. For example, an insurance company is likely to take into account potential driving safety predictors such as age, gender, and driving record when issuing car insurance policies.

**1.2 SCOPE OF THE PROJECT**

* **Making Predictions**

Predictive analytics takes past data and uses statistical methods to create projections about the future. Thus we take the past data images and then using the prediction algorithm. We train the data images accordingly, and with the trained image the currently live extracted images are compared, to find the pothole. Prediction was very rarely part of the daily operations of field workers.

The prediction algorithm analyses the pre-process data and predicts some rules for comparison of the pothole. The predictive algorithm used for accuracy percent and efficiency percent.

* **Managing the potholes**

In this, after the prediction is over the data images are stored in the cloud and whenever needed, the live extracted image is been compared with the previous trained data image and thus from that the conclusion will been obtained that the pothole is been identified or not.

**1.3 ORGANIZATION OF THE REPORT**

This report has been organized in such a way that it consists of seven chapters with each chapter explaining its purpose. Chapter 2 states the literature review that was made towards pothole detection using the sensors and with different learning algorithms. Chapter 3 describes about the problem definition: intakes issue in existing system along with its limitations. Chapter 4 gives the detailed description about our proposed pothole detection system with its input and output and system analysis requirements with its hardware and software requirements and respective UML diagrams. Chapter 5 gives the overall system architecture and the implementation of our project. Chapter 6 gives the results and discussions about the outcome of the project. Chapter 7 gives the brief conclusion of our project.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 PURPOSE OF THE STUDY**

In the last decades, the varsity numbers of research works were partitioned by Potential researchers for rapid determination from the work flow graph. A few of the Supreme works enclosed by them are examined below.

**2.1.1 DETAILED SURVEY**

At the beginning, to identify the potholes, sensors and different learning algorithms are been used. The sensors has some of the limitation and in other ways the external noise are also considered in the sensors and those noises also represent the pothole frequency. Such that the learning algorithms had the problem with the accuracy percent and efficiency percent. Thus this classifications are been briefly discussed in the below with the limitation and the dis advantages.

**2.2 POTHOLE DETECTION TECHNIQUES**

In the section below, different pothole detection techniques are discussed in detail. They have been classified into two types:

* Pothole detection using sensors.
* Pothole detection using different learning algorithms.

**2.2.1 POTHOLE DETECTION USING SENSORS:**

* **Pothole Detection using PIC 16F877A Micro controller, Ultrasonic Sensors HC-SR04, GPS Receiver and GSM SIM 900**

Proposed a system where sensors are mounted on public vehicles. These sensors record vertical and horizontal accelerations experienced by vehicles on their route. The installed GPS device takes note of its corresponding coordinates to locate the potholes and the collected data is processed to locate potholes along the path traversed earlier by the vehicle. The detection of pothole is done using the GPS sensors and 3-axis accelerometer [5]. The potholes are been further classified into their shapes and severity. The outputs are taken from the GPS sensor and 3-axis accelerometer and given as the input to the data cleaning algorithm.

* **Pothole detection using 3D scanner and Kinect sensor**

The potholes depth is detected using a 3D laser scanner in real-time but the overall cost of the 3D scanner is still high in vehicle significant and these works that are focused on the accuracy of the 3D measurement [15]. A Kinect sensor which is used to collect the pavement depth, images and calculate the approximate volume of a pothole. Although it is not cost-efficient as compared with industrial cameras and lasers, in this use of infrared technology based on a Kinect sensor for measurement is still an imaginary one and further research is necessary for the improvement in error rates

The optical device on a vehicle collects pot holes data, and the collected data is sent to a pothole detection algorithm [14]. From the pothole detection algorithm the information’s are been gathered and then sent to the road management center.

* **Evaluation of a potholes detection system on Ti C6678 digital signal processor**

To find the pothole, the approach was to find the discrepancies between the fitted plane and the actual plane, given a disparity map. Since road surfaces are generally smooth and continuous surfaces, a pothole area with uneven geometrical characteristics can be detected once the road surfaces are excluded [9].

The TI C6678 contains 8 Core Pac DSP cores. It supports its own proprietary Inter-Processor Communication (IPC) and Open MP for parallel programming. Software can benefit in terms of speed and efficiency through parallelism as more data can be processed concurrently. The detection stages of the pothole detection system, post optimization, have reached 5 frames per second. In order to make the complete system real-time applicable, the disparity calculation stage needs to be optimized and implemented in the multi-core DSP environment [13]. Optimizations can still be made in linear assembly or assembly to further reduce the computation time.

* **Smartphone Application to Estimate the potholes on Roads using Accelerometer and Gyroscope**

This device is a sensing application that makes the report of the road conditions. It requires the integration of hardware component for each of the vehicle, an embedded computer running Linux system is used for the data processing, a Wi-Fi module for transmitting the gathered data, a GPS for the localization, and a 3-axis accelerometer for monitoring the road surface [14]. In the data collection phase is the most important one; since it is the most responsible one for collecting the road information’s. Therefore, a pre-processing phase should be applied in order to reduce the noise and in order to improve the road quality recognition. Due to several factors such as jerks or vibrations, turning, veering, braking, and as well as subtle changes in sensor orientation, a considerable amount of noises are been added to those signals.

The decision tree classiﬁer is applied to train the data and to classify the road and to build our model. Our experimental results shows an accuracy of 80.6%. By using this approach, the visualization of a road quality check map of a selected region can be viewed. Hence, this approach can provide the immediate feedback to drivers and local authorities [15].

**TABLE 1 – 2.1** it is clear that, by analyzing the pothole using the sensor and the implementation done, has the following limitation.

**TABLE 1. SUMMARY OF SENSORS**

|  |  |  |
| --- | --- | --- |
| **TECHNIQUE** | **SENSORS** | **LIMITATION** |
| **Sensors based** | Using PIC 16F877A Micro controller, Ultrasonic Sensors HC-SR04. | * The sensors does not provide accurate data every time. * The sensors may get damaged due to the rough road condition. |
| 3D scanner and Kinect sensor. | * The 3D scanner is a high-cost material compared to the other scanners. |
| Ti C6678 digital signal processor. | * The system is complicated. |
| Accelerometer and Gyroscope sensor. | * Suitable only for smaller samples. * Higher maintenance and technology cost. * Difficulty in developing module. |

**2.2.2 POTHOLE DETECTION USING DIFFERENT LEARNING ALGORITHMS:**

* **Pothole Detection using artificial neural networks Algorithm**

The tracker takes advantage of an autonomous vehicle's onboard sensing, to triangulate the pothole positions in the 3D world and odometry information to estimate it, relative movement concerning the vehicle [8].

This location estimate is then corrected by a neural network that is trained specifically for that purpose. Machine learning approaches, especially deep learning approaches, need data to train on. In this they propose a system for detection, tracking, and 3D localization of potholes which utilizes deep learning. The detection is carried out by an end-to-end trained neural Network, which adapts to detect pothole as small as 3 × 10 pixels. Detecting objects in images is a Challenging task, especially if it is to be done reliably, in real-time, and work for both small and large objects [6].For potholes, in real-time detections with low false negative and false positive rates are needed. Though it has an efficiency of 83%.

* **Pothole Detection using decision tree Algorithm**

The decision tree algorithm is a type of technique which uses the supervised learning algorithm that is mostly used in classification problems. It works on both the categorical and continuous input and output variables. Different classifiers are trained and the data’s are tested to figure out the accurate detection of pothole from the images. The decision tree Algorithm has an efficiency of 76% [3], in this segment, the methods used for the pothole visualization and the detection techniques are discussed. It can be accomplished in two ways namely downs calling the image and color histogram.

This projects different colors and total count of pixels in each color. Histogram is said to be the perfect with the variety of data from the image. The presence of potholes in different shapes and colors makes figuring the unique feature a different job.

* **Pothole Detection using K-MEANS CLUSTERING & RANDOM FOREST Algorithm**

K means algorithm is the machine learning algorithm. In this they have considered the training data which is free from the low-frequency components. To remove low-frequency components they have used a high pass filter. The efficiency obtained here is 70%. The use of k means clustering algorithm is to classify the data into two different clusters. These two clusters as potholes and no potholes [7]. These clusters are made by taking into account the threshold value of the z-axis. They have used the Random Forest classifier to distinguish the data which are generate during the test phase [2]. Depending on that it will predict the road conditions status.

* **Pothole Detection using Euclidean distance mapping technique**

This proposed method for the pothole detection is based on which the potholes are represented with the intensity of the image, which uses the high values. The set of generated algorithm is based on the threshold values, by selecting the areas of the image which contains the pixels with high-intensity values. It selects the areas mostly which are above the car hood which represents the potholes that are visible by, sometimes the car gets too close to the wayside and the objects outside the roads are included [12]. The threshold value algorithm used to remove the ROI which represent the pixels, by computing the intensity of the image Y of the current frame F. With the intensity value lower than the threshold value. The threshold is denoted T and is set as

T = max (90, ¯ y + σ Y)

For each region, the features are extracted: by its size, by the regularity of the intensity surface, by the contrast with respect to a background model, and the region's contour with length and shape. The potholes that are successfully tracked are represented in the consecutive frames [13].

**TABLE 2. ALGORITHMS USED FOR POTHOLE DETECTION**

|  |  |  |
| --- | --- | --- |
| **TECHNIQUE** | **ALGORITHMS** | **LIMITATION** |
| **Algorithm based detection** | Decision tree Algorithm | * In this method it does not work well as the edges in the roads, it creates the needless contours which are indicated as potholes. * Thus it does not provide a high accuracy for pothole detection. |
| K-MEANS CLUSTERING & RANDOM FOREST Algorithm | * The training data may not be given in advance. * This is not a reliable method since the training data images are not given. * The efficiency is low when compared to the decision tree algorithm. |
| Artificial neural networks Algorithm | * The limitations of artificial neural networks currently is the amount of needed training data. Even though there exist data sets for pothole detection, we present a new data set. * False-positive detections can lead to the automated vehicle behaving unpredictably. |
| Euclidean distance mapping technique | * The cars passing through which are creating shadows which may possibly be mistakenly labeled as potholes. * The potholes are visible only in the frame, outside the frame the potholes are not detected. |

**TABLE 2 – 2.2** it is clear that, by analyzing the pothole using the different learning algorithm and the implementation done, has the following limitation.

**2.3 RESEARCH WORKS**

Pothole detection system provides endless scope for research ideas. The major problems, further to be considered in future are;

* Pothole detection is one of the emerging and trending method to find the potholes on roads. Prediction is always associated with inaccuracy. Therefore, efficiency of algorithm to predict the pothole density can be increased to predict accurate results.
* In case of sensor based detection, there is a huge disadvantage of placing the sensors due to their range issues. This can be overcome by increasing the range of sensors.
* The detection of the pothole provides more accuracy. Therefore, the videos can be taken as the input and divided into frames to find the potholes.
* Increasing efficiency of prediction technique that provides both current and future scenario prediction in a simplified design.

**CHAPTER 3**

**PROBLEM DEFINITION**

**INDIA, the second most populous Country in the World and a fast growing economy, is known to have a gigantic network of roads. Roads are the dominant means of transportation in India today. They carry almost 90 percent of country’s passenger traffic and 65 percent of its freight. However, most of the roads in India are narrow and congested with poor surface quality and road maintenance needs are not satisfactorily met. No matter where you are in India, driving is a breath-holding, multi-mirror involving, potentially life threatening affair. Thus I tend to provide a solution by using OpenCv and Deep Learning Algorithm, through this solution the cause to death can be reduced.**

**3.1 EXISTING SYSTEM**

* **Real Time Pothole Detection using Android Smartphones with Accelerometers:**

The mobile sensing system for road irregularity detection using Android OS based smart-phones. Selected data processing algorithms are discussed and their evaluation presented with true positive rate as high as 90% using real world data. The optimal parameters for the algorithms are determined and the recommendations for their application are described.

* **Pothole Detection and Warning System using Wireless Sensor Networks:**

This aims at proposing a novel pothole detection system which assists the driver in avoiding potholes on the roads, by giving prior warnings. Interest in Intelligent Vehicle Systems comes from the problems caused by traffic congestion worldwide and a synergy of new information technologies for simulation, real-time control and communications networks.

The architectural design further proposes a low response time, low maintenance and deployment cost solution to this problem.

The pothole sensor plug-in monitors the changes in the acceleration in order to detect potholes. For this, the user needs to have an Android smartphone. The device‘s built-in accelerometer is used to collect the x, y and z axis accelerations. The native GPS (Global Positioning System) chip is used to collect location co-ordinates. As seen in Figure 1, this is included in the Display module, with appropriate buttons to carry out the task. The implemented plug-in should be generic so that it can be used along with other context-aware applications.

The pothole detection algorithm is good, in terms of speed and accuracy. This algorithm accepts the training set and displays the pothole scenario in the particular area. Specific details related to the algorithm are present in following paragraphs, and one can get the gist of how values required by the algorithm are collected and used, by noting the Logic module.

**CHAPTER 4**

**PROPOSED SYSTEM**

**4.1 ABOUT THE PROPOSED WORK**

The project will provides the details about the potholes in the roads are sensed by the stereo cameras which is placed in the front end of the vehicle. So that it can sense over a wide angle, the sensed data is send to the software application so that it can analyze the potholes with the various size and density which are in the road the fetched data is processed with the current longitude and latitude through the GPS in the vehicle.

To detect the potholes we have been using disparity map algorithm through which the data are been pre - processed and then the inputs are taken with stereo cameras then the image are been spitted into two; namely the data set image and another is 3d image point cloud. Thus the 3d image point cloud refines the entire road into a single lane, such that it gives a good accuracy. According to the dataset the Google map is updated with the various color dots. The orange color dots refers the potholes in the road. So the map provide an alternative optimal path to the destination is made and traces the path in the map. If those potholes have repaired that can be removed for the Google map if not it will be mentioned as a long term thread.

Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from the raw input. Machine learning uses programmed algorithms that receive and analyze input data to predict output values within an acceptable range. As new data is fed to these algorithms, they learn and optimize their operations to improve performance, developing 'intelligence' over time.

When collecting information from various parts it will leads to larger data ware house, so in order to reduce the data size the Big data concept is used which is make to analyze the fetch the data from the data make easier. So the user can able to get the comfort places or routes through these types of added data in the Google map.

**4.2 SYSTEM ANALYSIS REQUIREMENTS**

In testing the internal requirements are required to say any how it will interact with the user. There are different types of interfaces. The hardware interfaces will describe about the hardware devices. The communication interfaces will describe about the network interfaces. So in our project we are using only the hardware and software interfaces.

**4.2.1 HARDWARE REQUIREMENTS**

|  |  |
| --- | --- |
| **Processor** | 7th Gen i7 Processor |
| **Hard disk** | 100 GB |
| **RAM** | 8 GB |

**4.2.2 SOFTWARE REQUIREMENTS**

|  |  |
| --- | --- |
| **Operating System** | Windows 8 / 10 – 64 bit  Lollipop 5.0 or more |
| **Tools** | * Android Studio * Firebase * Python * Java * Visual Studio Code |

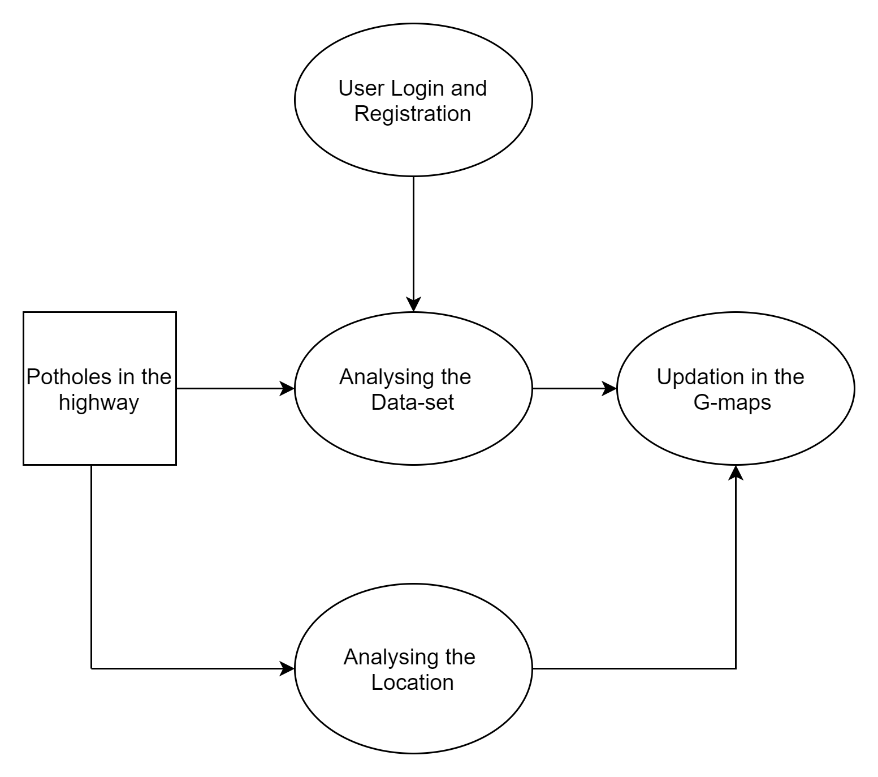
**4.3 OBJECT ORIENTED ANALYSIS**

Object-oriented analysis and design (OOAD) is a popular technical approach to analyzing, designing an application, system, or business by applying the [object-oriented paradigm](http://en.wikipedia.org/wiki/Object-oriented_programming) and visual modeling throughout the [development life cycles](http://en.wikipedia.org/wiki/Software_development_process) to foster better stakeholder communication and product quality.

According to the popular guide [Unified Process](http://en.wikipedia.org/wiki/Unified_Process), OOAD in modern software engineering is best conducted in an iterative and incremental way. Iteration by iteration, the outputs of OOAD activities, analysis models for OOA and design models for OOD respectively, will be refined and evolve continuously driven by key factors like risks and business value.

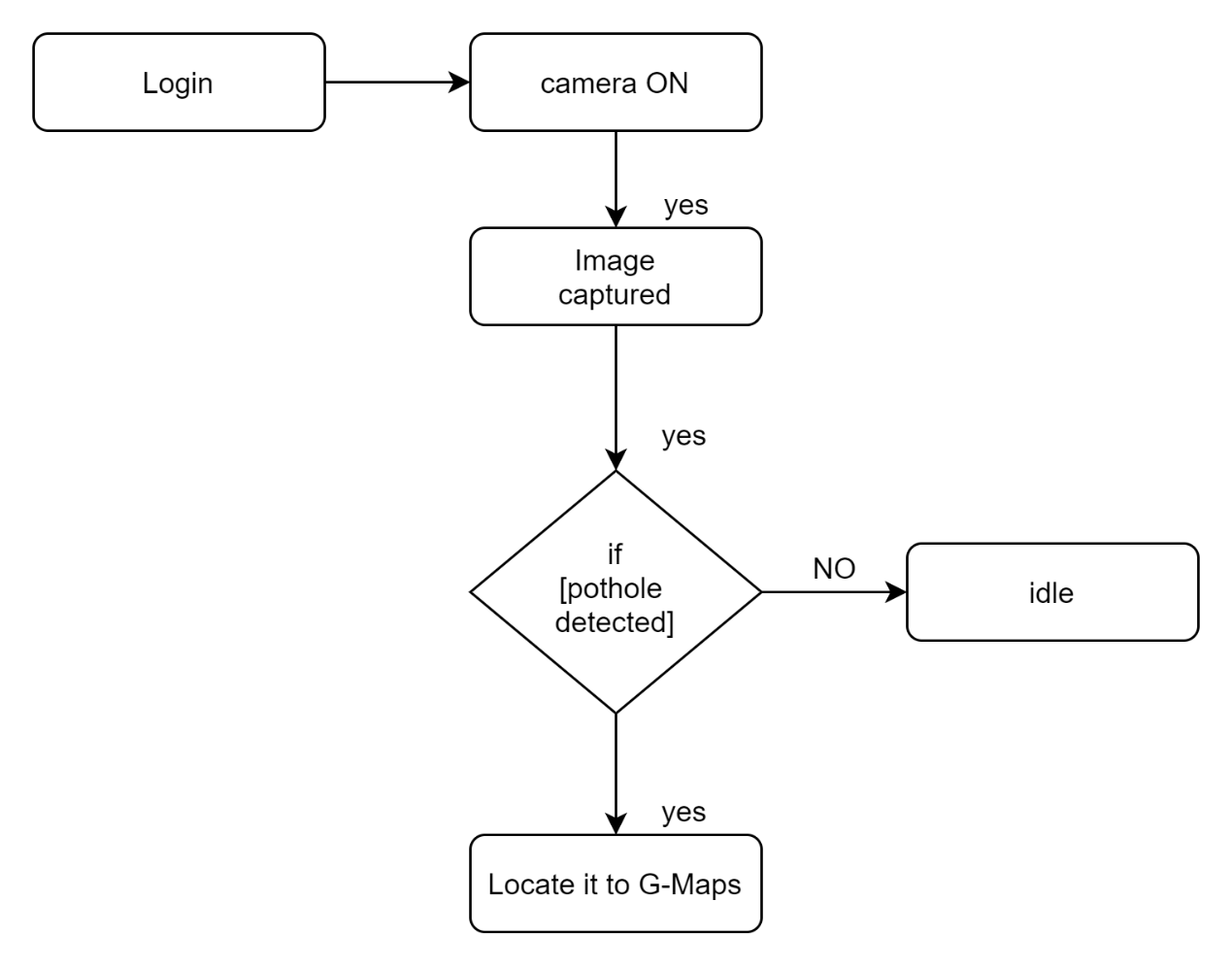
**4.3.1 UML DIAGRAMS**

The Unified Modeling Language (UML) is a general-purpose [modeling language](http://en.wikipedia.org/wiki/Modeling_language) in the field of [software engineering](http://en.wikipedia.org/wiki/Software_engineering), which is designed to provide a standard way to visualize the design of a system.

**4.3.1.1 USE CASE DIAGRAM**

**Fig 4.1 Use Case diagram for overall system**

**4.3.1.2 ACTIVITY DIAGRAM**

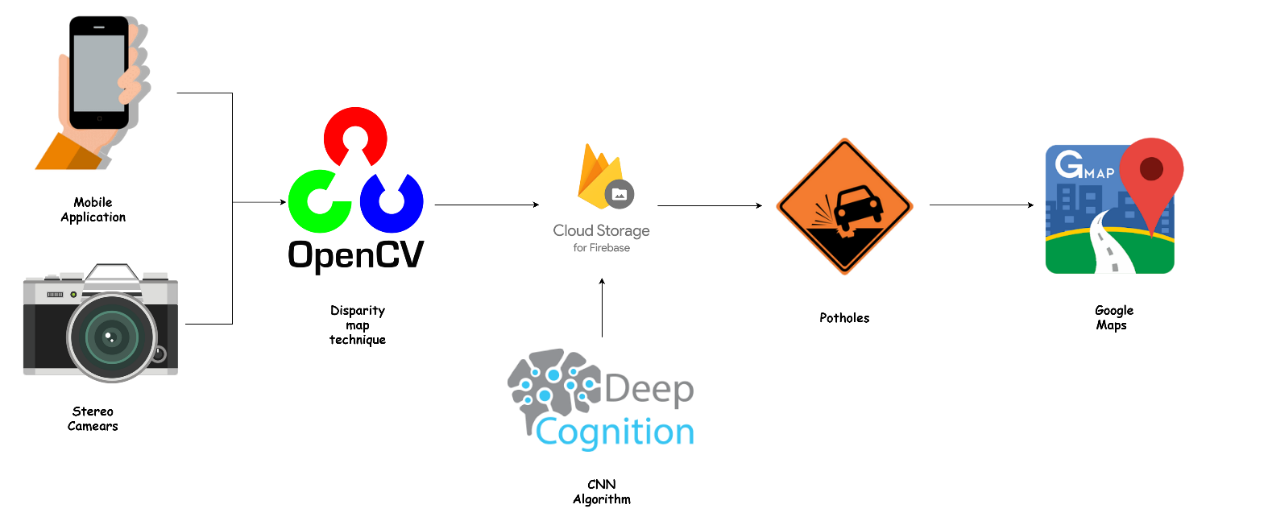
****

**Fig 4.2 Activity diagram for the proposed system**

**CHAPTER 5**

**SYSTEM DESIGN AND IMPLEMENTATION**

**5.1 SYSTEM ARCHITECTURE**

****

**Fig 5.1 System Architecture**

**5.2 MODULES DESCRIPTION**

**5.2.1 USER MODULE:**

On user module the user only has the access to login or register. From which then the user need to take the live capture of a pothole if it matches with the trained data then it gives the result as pothole is identified if not it display no potholes is present. It is also been integrated with Google maps to locate the potholes.

**5.2.2 LEARNING MODULE:**

Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence.

Deep Learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound, and text. For more about deep learning algorithms such as neural networks (RNN, CNN, and ANN), clustering (k-means, HCA,), etc..,

**5.2.3 CNN (Convolution Neural Network):**

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptron’s, a machine learning unit algorithm, for supervised learning, to analyze data. CNNs apply to image processing, natural language processing and other kinds of cognitive tasks.

A convolutional neural network (CNN) contains one or more convolutional layers, pooling or fully connected, and uses a variation of multilayer perceptron’s discussed above. Convolutional layers use a [convolution operation](https://en.wikipedia.org/wiki/Convolution) to the input passing the result to the next layer. This operation allows the network to be deeper with much fewer parameters Convolutional neural networks show outstanding results in image and speech applications.

**5.2.4 DISPARITY MAPPING MODULE:**

The Disparity Map, stereo correspondence algorithms take two (or sometimes more) rectified greyscale images as input, and produce a disparity map d(x, y) for each pixel in one of the input images, typically stored as a greyscale image.

Disparity map refers to the apparent pixel difference or motion between a pair of stereo images. That motion is the disparity. In a pair of images derived from stereo cameras, you can measure the apparent motion in pixels for every point and make an intensity image out of the measurements.

Disparity refers to the difference in image location of an object seen by the left and right eyes, resulting from the eyes' horizontal separation (parallax). In computer vision, binocular disparity refers to the difference in coordinates of similar features within two stereo images.

**5.3 SYSTEM IMPLEMENTATION**

The programming environment used in our project are java, android studio and Photoshop for designing the icons. Whenever the user has any grievance regarding environmental to rectify all the bugs that happened in previous version and that can be helpful to the common people also they can use this system to lodge their complaints.

On first step through Open CV the potholes are been detected and the images are been directly sent to the cloud where the data set trained images are stored. In second step the images are been compared with the trained images if any of the image matches it gives the result as the pothole is been identified.

**5.3.1 PYTHON LANGUAGE**

### Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. We use packages namely numpy, cv2, pygame etc..,

**5.3.2 JAVA FOR ANDROID STUDIOS**

It can dynamically construct and call objects, which improves the flexibility of malware logic execution. The corresponding function of the mechanism is java.lang.reselect.Method.invoke()

### **5.3.3 NATIVE DEVELOPMENT KIT (NDK)**

It uses Java Native Interface (JNI) to call native codes. However, the use of native codes cannot be limited by software application permissions, and native nodes can use vulnerabilities of the system to perform illegal operations, such as trying to get root permissions. When stating relevant functions, there are native words in codes to locate native functions.

**5.3.4 CONVULUTION NEURAL NETWORK ALGORITHM**

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

# Python program to create

# Image Classifier using CNN

# Importing the required libraries

import cv2

import os

import numpy as np

from random import shuffle

from tqdm import tqdm

'''Setting up the env'''

TRAIN\_DIR = 'E:/dataset / pothole vs non potholes / train'

TEST\_DIR = 'E:/dataset / pothole vs non potholes / test1'

IMG\_SIZE = 50

LR = 1e-3

'''Setting up the model which will help with tensorflow models'''

MODEL\_NAME = ' potholevsnonpotholes -{}-{}.model'.format(LR, '6conv-basic')

'''Labelling the dataset'''

def label\_img(img):

word\_label = img.split('.')[-3]

# DIY One hot encoder

if word\_label == 'pothole': return [1, 0]

elif word\_label == 'non pothole': return [0, 1]

'''Creating the training data'''

def create\_train\_data():

# Creating an empty list where we should store the training data

# after a little preprocessing of the data

training\_data = []

# tqdm is only used for interactive loading

# loading the training data

for img in tqdm(os.listdir(TRAIN\_DIR)):

# labeling the images

label = label\_img(img)

path = os.path.join(TRAIN\_DIR, img)

# loading the image from the path and then converting them into

# greyscale for easier covnet prob

img = cv2.imread(path, cv2.IMREAD\_GRAYSCALE)

# resizing the image for processing them in the covnet

img = cv2.resize(img, (IMG\_SIZE, IMG\_SIZE))

# final step-forming the training data list with numpy array of the images

training\_data.append([np.array(img), np.array(label)])

# shuffling of the training data to preserve the random state of our data

shuffle(training\_data)

# saving our trained data for further uses if required

np.save('train\_data.npy', training\_data)

return training\_data

'''Processing the given test data'''

# Almost same as processing the training data but

# we dont have to label it.

def process\_test\_data():

testing\_data = []

for img in tqdm(os.listdir(TEST\_DIR)):

path = os.path.join(TEST\_DIR, img)

img\_num = img.split('.')[0]

img = cv2.imread(path, cv2.IMREAD\_GRAYSCALE)

img = cv2.resize(img, (IMG\_SIZE, IMG\_SIZE))

testing\_data.append([np.array(img), img\_num])

shuffle(testing\_data)

np.save('test\_data.npy', testing\_data)

return testing\_data

'''Running the training and the testing in the dataset for our model'''

train\_data = create\_train\_data()

test\_data = process\_test\_data()

# train\_data = np.load('train\_data.npy')

# test\_data = np.load('test\_data.npy')

'''Creating the neural network using tensorflow'''

# Importing the required libraries

import tflearn

from tflearn.layers.conv import conv\_2d, max\_pool\_2d

from tflearn.layers.core import input\_data, dropout, fully\_connected

from tflearn.layers.estimator import regression

import tensorflow as tf

tf.reset\_default\_graph()

convnet = input\_data(shape =[None, IMG\_SIZE, IMG\_SIZE, 1], name ='input')

convnet = conv\_2d(convnet, 32, 5, activation ='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 64, 5, activation ='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 128, 5, activation ='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 64, 5, activation ='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 32, 5, activation ='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = fully\_connected(convnet, 1024, activation ='relu')

convnet = dropout(convnet, 0.8)

convnet = fully\_connected(convnet, 2, activation ='softmax')

convnet = regression(convnet, optimizer ='adam', learning\_rate = LR,

loss ='categorical\_crossentropy', name ='targets')

model = tflearn.DNN(convnet, tensorboard\_dir ='log')

# Splitting the testing data and training data

train = train\_data[:-500]

test = train\_data[-500:]

'''Setting up the features and lables'''

# X-Features & Y-Labels

X = np.array([i[0] for i in train]).reshape(-1, IMG\_SIZE, IMG\_SIZE, 1)

Y = [i[1] for i in train]

test\_x = np.array([i[0] for i in test]).reshape(-1, IMG\_SIZE, IMG\_SIZE, 1)

test\_y = [i[1] for i in test]

'''Fitting the data into our model'''

# epoch = 5 taken

model.fit({'input': X}, {'targets': Y}, n\_epoch = 5,

validation\_set =({'input': test\_x}, {'targets': test\_y}),

snapshot\_step = 500, show\_metric = True, run\_id = MODEL\_NAME)

model.save(MODEL\_NAME)

'''Testing the data'''

import matplotlib.pyplot as plt

# if you need to create the data:

# test\_data = process\_test\_data()

# if you already have some saved:

test\_data = np.load('test\_data.npy')

fig = plt.figure()

for num, data in enumerate(test\_data[:20]):

# pothole: [1, 0]

# not pothole: [0, 1]

img\_num = data[1]

img\_data = data[0]

y = fig.add\_subplot(4, 5, num + 1)

orig = img\_data

data = img\_data.reshape(IMG\_SIZE, IMG\_SIZE, 1)

# model\_out = model.predict([data])[0]

model\_out = model.predict([data])[0]

if np.argmax(model\_out) == 1: str\_label ='Dog'

else: str\_label ='Cat'

y.imshow(orig, cmap ='gray')

plt.title(str\_label)

y.axes.get\_xaxis().set\_visible(False)

y.axes.get\_yaxis().set\_visible(False)

plt.show()

**5.3.5 DISPARITY MAP ALGORITHM**

The input of this procedure is a dense disparity map having subpixel accuracy. Since the performance of disparity map modeling relies entirely on the disparity estimation accuracy, the dense disparity map was obtained from a stereo road image pair through our disparity estimation algorithm where the stereo matching search range propagates iteratively from the bottom of the image to the top, and the subpixel disparity map is refined by iteratively minimizing an energy function with respect to the interpolated correlation parabolas. ϴ Estimation and Disparity Map Rotation: Over the past decade, considerable effort has been made to improve the roll angle estimation.

**Algorithm 1:** ϴ estimation using GSS.

**Data:** disparity map

**Result:** ϴ

1 set ϴ 1 and ϴ 2 to –Π/2 and Π/2, respectively;

2 compute E0min (ϴ1) using Eq. 13;

3 compute E0min (ϴ2) using Eq. 13;

4 while ϴ2 􀀀ϴ1 > "ɛᶿ do

5 set ϴ 3 and ϴ 4 to k ϴ1 + (1 – K) ϴ2 and k ϴ2 + (1 – k) ϴ1,

Respectively;

6 compute E0min (ϴ 3) using Eq. 13;

7 compute E0min (ϴ 4) using Eq. 13;

8 if E0min (ϴ 3) > E0min (ϴ 4) then

9 ϴ 1 is replaced by ϴ 3;

10 else

11 ϴ 2 is replaced by ϴ 4;

12 end

13 end

**CHAPTER 6**

**RESULTS & DISCUSSION**

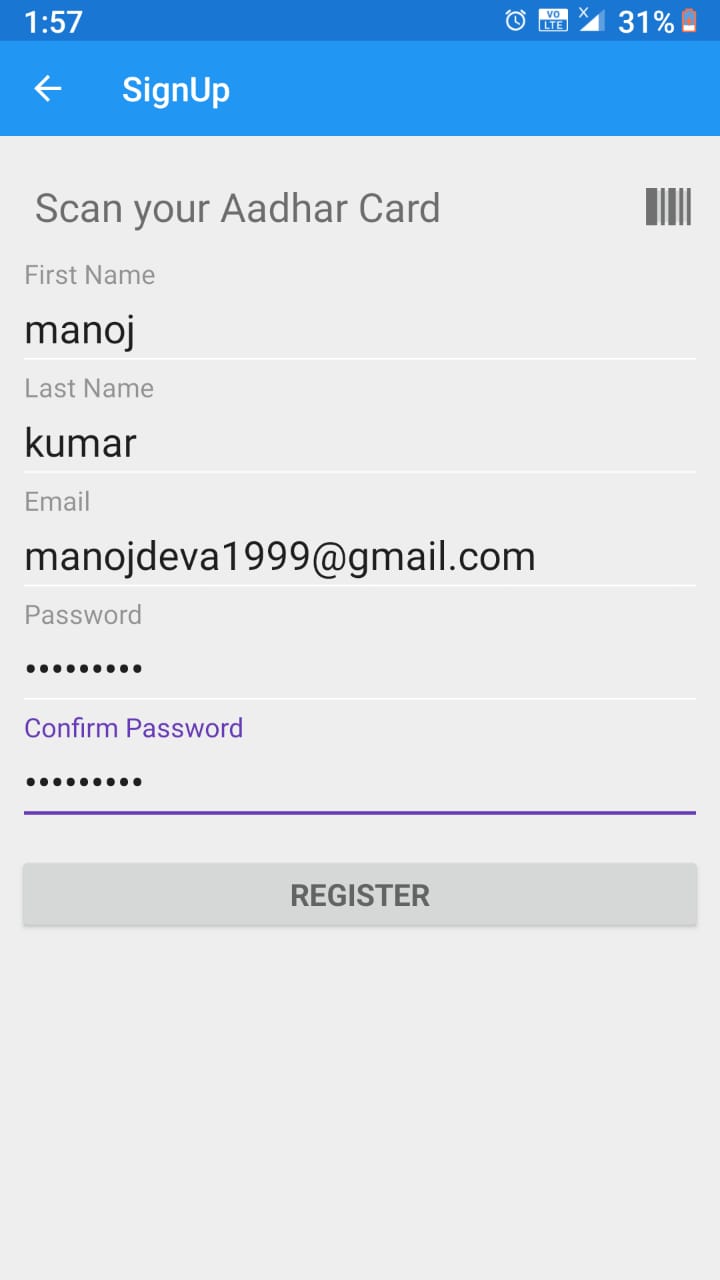
As we know that image data is trained and generated through the Convolution Neural Network algorithm for each and every field almost, with different angles so that the live captured image from any angle or direction can be compared to produce the output in an efficient manner. So regarding our project we have done trained the data on the roads.

We actually trained the data in a huge amount that in all possible ways the output can be obtained in an efficient manner. As the trained data set images are high the output excepted is also at high efficiency. We would be training the data set images using TENSORFLOW LITE such that lite allows us to integrate the code in the ANDROID STUDIOS. The trained data would be then moved to the cloud host and then it would be processed. Thus from the live footage, simentensously the image is been extracted and then it’s been compared with the data set stored in the cloud. The result proves that the whenever the image is been compared with the data set, it tends to provide a threshold value when the value is above than the average value then it is been defined as the pothole is detected. When the value is less than the average threshold value then it is defined as non-pothole.

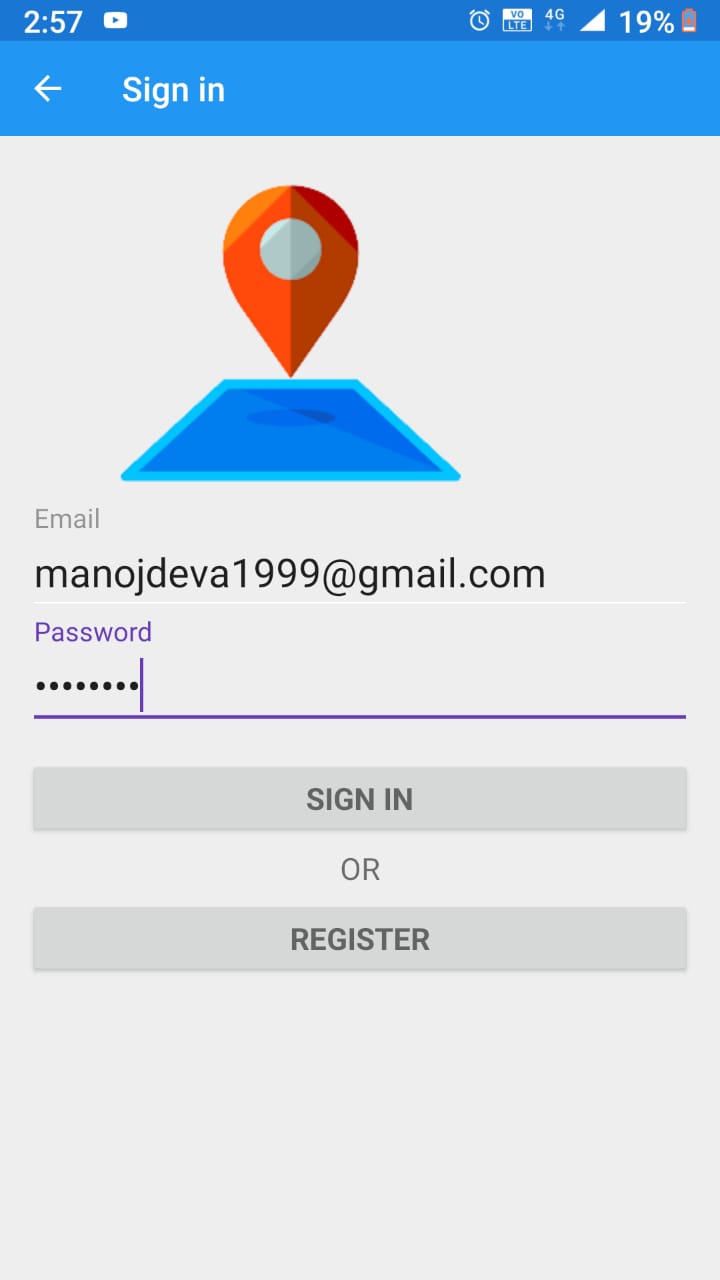
In the Tensor flow lite, the CNN algorithm has been pushed in, we have also compared all the algorithms and it proves that more efficiently CNN algorithm is predicting the result and with the help of the pre-process data we get the output/result. Tensor flow lite also provides the predictive calculus result.

****

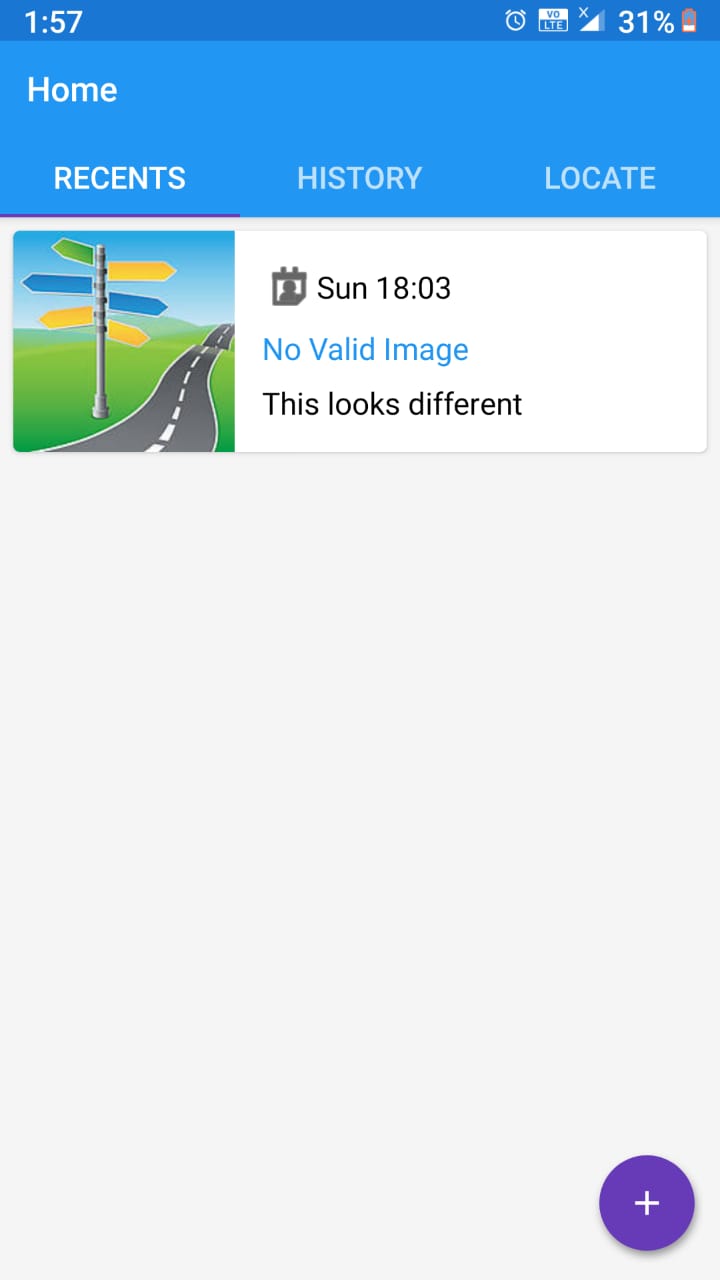
**Fig 6.1 HOME PAGE FOR THE ANDROID APPLICATION**

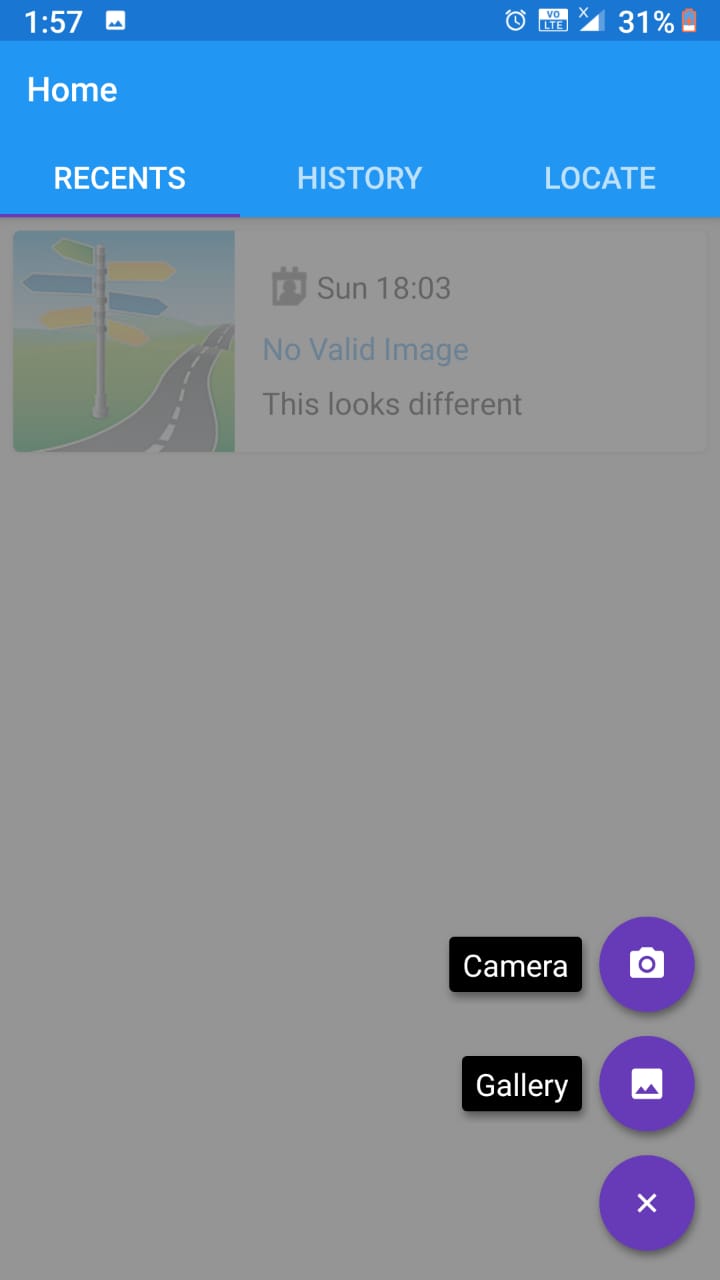


**Fig 6.2 REGISTRATION PAGE FOR NEW USER**

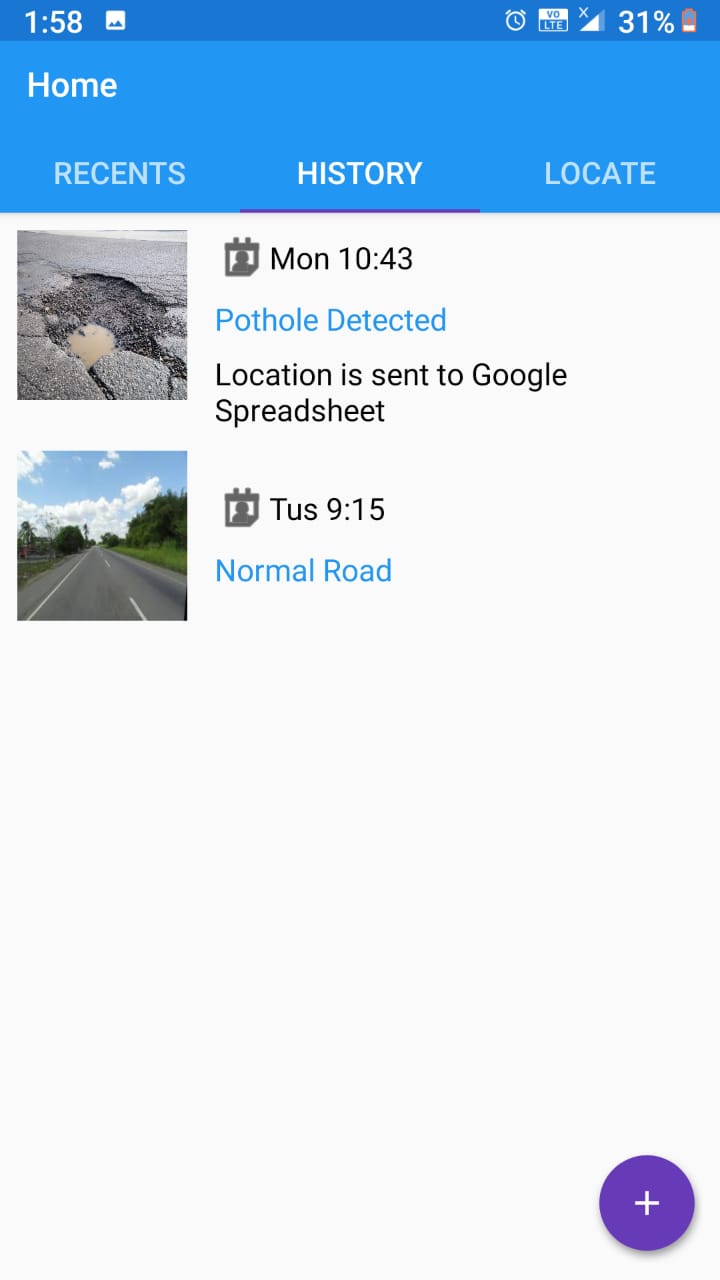


**Fig 6.3 LOGIN FORM FOR THE ANDROID APPLICATION**

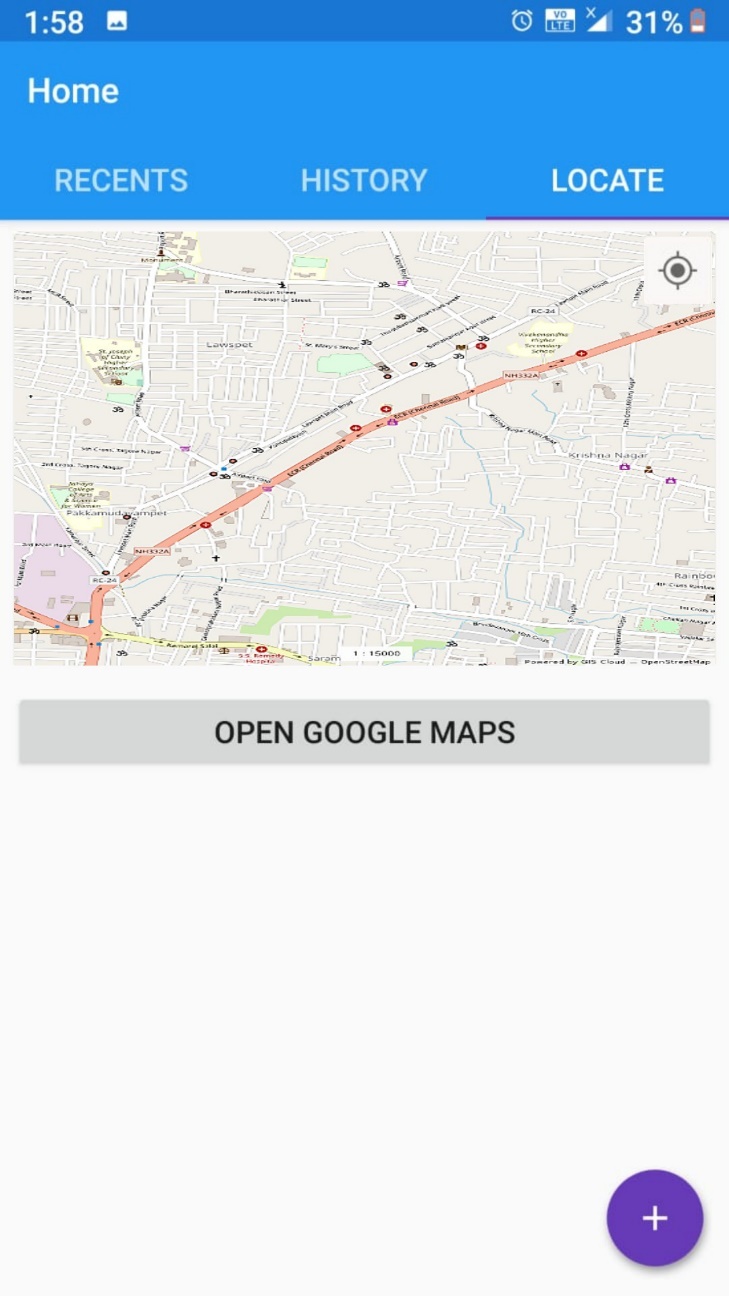


****

**Fig 6.4 to view the recent pothole detected**

****

**Fig 6.5 this tab shows the history about the potholes detected**

****

**Fig 6.6 shows in the map where the pothole is been detected**

**CHAPTER 7**

**CONCLUSION**

The Pothole Detection System is an attempt to provide its users with better knowledge about the routes of their transportation. We postulate that accurate pothole detection is possible. We believe that our experience will help to improve efficiency and reduce time and effort for further experiments using the Android platform. The conventional image processing techniques provide results that are then insignificant. The machine learning algorithm for prediction of road anomalies. We described the BDS (Bumps Detection System) and algorithms to monitor the road surface conditions. It uses cameras through open CV for collection of data and GPS for plotting the location of detected potholes in Google map. We have used CNN algorithm on training data.

The Smart phone based method is very useful to view the plotted areas on Google maps. It has the advantage of high scalability as smartphone users increases day by day. The proposed method gives the best possible results. The proposed system runs in real-time and achieves high accuracy in challenging conditions. We believe that our pothole detection pipeline can lead to smoother trajectories of automated vehicles in urban environments and allow for more accurate mapping systems.

**REFERENCES**

[1] "Real time pothole detection using Android smartphones with accelerometers," A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, L. Selavo, dcoss, pp.1-6, 2011 InternationalConference.

[2] Jeong-Joo Kim, Su-il Choi, “Implementation of Pothole Detection System Using 2D LiDAR”, International Conference on Electronics, Information, and Communication, 2017, 607-610.

[3] S. Nienaber, M.J. Booysen, R.S. Kroon, “Detecting potholes using simple image processing techniques and Real-world Footage”, 34th Annual Southern African Transport Conference SATC 2015 - Theme - Transport: Working together to deliver ‘SAKHASONKE', 6 to 9 July 2015, CSIR International Convention Centre, Pretoria, South Africa.

[4] Open CV is used for Image processing and Machine Learning: <http://docs.opencv.org>.

[5] Eriksson, Girod, ‘The pothole patrol: Using a mobile sensor network for road surface monitoring, ‘Sixth Annual International conference on Mobile Systems, Applications and Service IEEE,USA,june. 2008.

[6] Aniket Kulkarni, Nitish Mhalgi , Sagar Gurnani ‘Pothole Detection System using Machine Learning on Android,’ International Journal of Emerging Technology and Advanced Engineering, vol. 4, July 2014 .

[7] Roma K. Goregaonkar,Snehal Bhosale, ‘Assistance to Driver and Monitoring the Accidents on Road by using Three Axis Accelerometer and GPS System,’ Proc. 1st International Conference , vol.10, pp. 1-8, April 2014.

[8] Pothole Detection and Warning System using Wireless Sensor Networks” Sudarshan, Vijay, Goyal, Kulkarni, Arya, Embedded Real-Time Systems Laboratory, Indian Institute of Technology Bombay (2009).

[9] IMPLEMENTATION AND EVALUATION OF A POTHOLE DETECTION SYSTEM ON TI C6678 DIGITAL SIGNAL PROCESSOR Chee Kin Chan, Yuan Gao, Zhen Zhang, and Naim Dahnoun University of Bristol, Merchant Venturers' Building, Woodland Road, Clifton, BS8 1UB.

[10] RoadSense: Smartphone Application to Estimate Road Conditions using Accelerometer and Gyroscope Azza Allouch, Anis Koubaa, Tarek Abbes, and Adel Ammar ˆ.

[11] Image Processing Techniques for Automated Road Defect Detection: A Survey IH. Bello- Salau, 2A. M. Aibinu, 3E. N. Onwuka, 4J. J. Dukiya and sA. J. Onumanyi.

[12] Automatic Detection of Potholes and Humps on Road Parag Kadale1, Shivam Barde2, Anand Pawar3.

[13] Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone Hiroya Maeda∗, Yoshihide Sekimoto, Toshikazu Seto, Takehiro Kashiyama, Hiroshi Omata University of Tokyo, 4-6-1 Komaba, Tokyo, Japan.

[14] Detection and Counting of Pothole using Image Processing Techniques Vigneshwar.K, Instrumentation Engineering, Dept. of EIE. Pondicherry Engineering college Puducherry, India vigneshjack3@gmail.com Hema Kumar.B Assistant Professor, Dept. of EIE. Pondicherry Engineering college Puducherry, India [hemakumarb@pec.edu](mailto:hemakumarb@pec.edu).

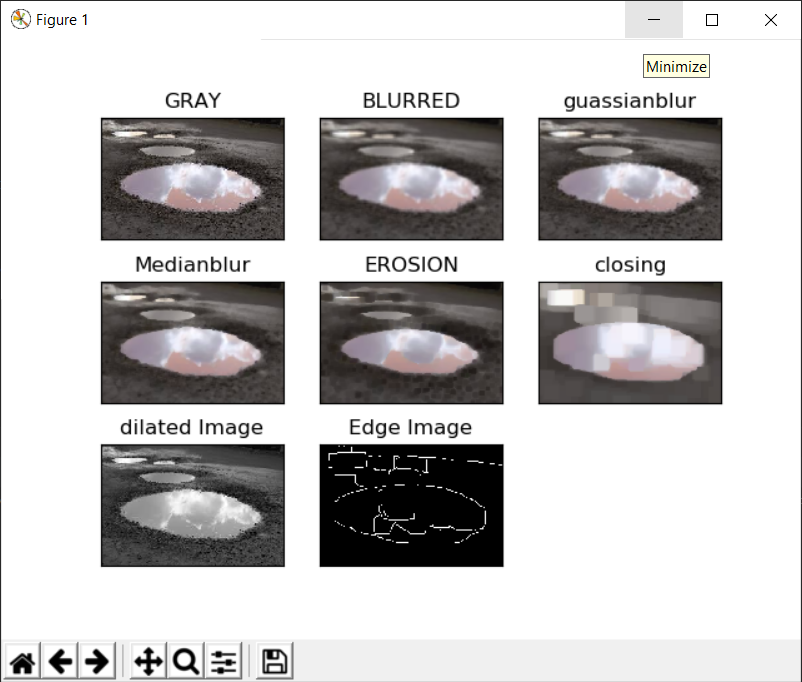
[15] Pothole Detection and Tracking in Car Video Sequence Ionut Schiopu∗, Jukka P. Saarinen†, Lauri Kettunen‡, and Ioan Tabus∗ ∗Department of Signal Processing, Tampere University of Technology, Tampere, Finland †Nokia Technologies, Tampere, Finland ‡Department of Electrical Engineering, Tampere University of Technology, Tampere, Finland.

**APPENDIX I**

**SCREEN SHOTS**

****

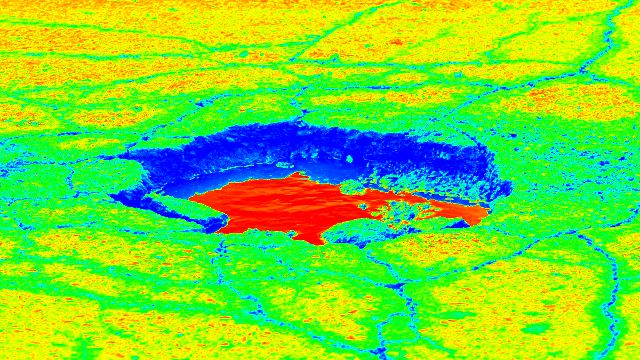
**Fig 1 A real image of the pothole**



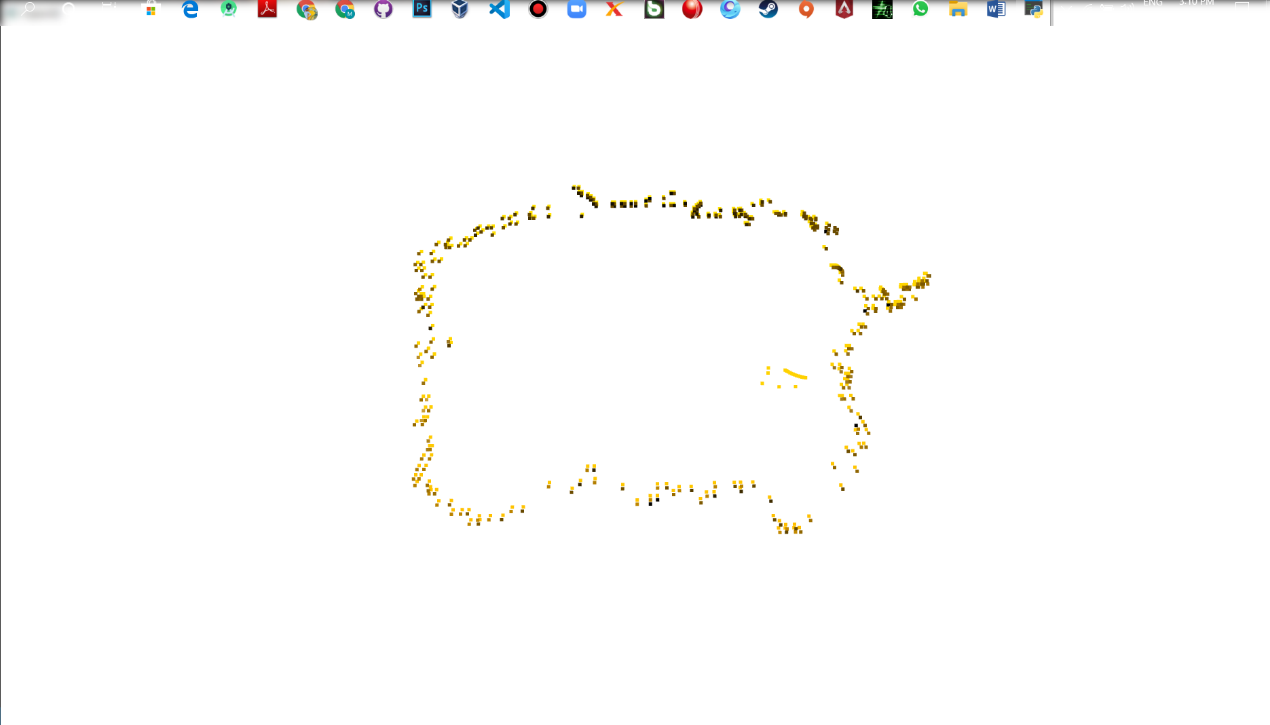
**Fig 2 Predictive analysis for the Potholes**

****

**Fig 3 pothole image to convert it into heat map**

****

**Fig 4 heat map generated image of the pothole**



**Fig 5 3d point cloud image of the pothole**

**APPENDIX II**

**SAMLE CODE**

import cv2

import numpy as np

import pygame

import cv2 as cv

import time

#import smtplib

from matplotlib import pyplot as plt

im = cv2.imread('index4.jpg')

# CODE TO CONVERT TO GRAYSCALE

gray1 = cv2.cvtColor(im,cv2.COLOR\_BGR2GRAY)

# save the image

cv2.imwrite('g.jpg', gray1)

#CONTOUR DETECTION CODE

imgray = cv2.cvtColor(im,cv2.COLOR\_BGR2GRAY)

ret,thresh = cv2.threshold(imgray,127,255,0)

contours1, \_ = cv2.findContours(thresh,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_NONE)

contours2, \_ = cv2.findContours(thresh,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_SIMPLE)

#img1 = im.copy()

img2 = im.copy()

#out = cv2.drawContours(img1, contours1, -1, (255,0,0), 2)

out = cv2.drawContours(img2, contours2, -1, (250,250,250),1)

#out = np.hstack([img1, img2])

cv2.imshow('img1',img2)

cv2.waitKey(0)

plt.subplot(331),plt.imshow(im),plt.title('GRAY')

plt.xticks([]), plt.yticks([])

img = cv2.imread('index4.jpg',0)

ret,thresh = cv2.threshold(img,127,255,0)

contours,hierarchy = cv2.findContours(thresh, 1, 2)

cnt = contours[0]

M = cv2.moments(cnt)

#print M

perimeter = cv2.arcLength(cnt,True)

#print perimeter

area = cv2.contourArea(cnt)

#print area

epsilon = 0.1\*cv2.arcLength(cnt,True)

approx = cv2.approxPolyDP(cnt,epsilon,True)

#print epsilon

#print approx

for c in contours:

rect = cv2.boundingRect(c)

if rect[2] < 100 or rect[3] < 100: continue

#print cv2.contourArea(c)

x,y,w,h = rect

cv2.rectangle(img2,(x,y),(x+w,y+h),(0,255,0),8)

cv2.putText(img2,'Moth Detected',(x+w+40,y+h),0,2.0,(0,255,0))

cv2.imshow("Show",img)

#cv2.waitKey()

#cv2.destroyAllWindows()

k = cv2.isContourConvex(cnt)

#to check convexity

print(k)

#blur

blur = cv2.blur(im,(5,5))

#guassian blur

gblur = cv2.GaussianBlur(im,(5,5),0)

#median

median = cv2.medianBlur(im,5)

#erosion

kernel = np.ones((5,5),np.uint8)

erosion = cv2.erode(median,kernel,iterations = 1)

dilation = cv2.dilate(erosion,kernel,iterations = 5)

#erosion followed dilation

closing = cv2.morphologyEx(dilation, cv2.MORPH\_CLOSE, kernel)

#canny edge detection

edges = cv2.Canny(dilation,9,220)

#plotting using matplotlib

plt.subplot(332),plt.imshow(blur),plt.title('BLURRED')

plt.xticks([]), plt.yticks([])

plt.subplot(333),plt.imshow(gblur),plt.title('guassianblur')

plt.xticks([]), plt.yticks([])

plt.subplot(334),plt.imshow(median),plt.title('Medianblur')

plt.xticks([]), plt.yticks([])

plt.subplot(337),plt.imshow(img,cmap = 'gray')

plt.title('dilated Image'), plt.xticks([]), plt.yticks([])

plt.subplot(338),plt.imshow(edges,cmap = 'gray')

plt.title('Edge Image'), plt.xticks([]), plt.yticks([])

plt.subplot(335),plt.imshow(erosion),plt.title('EROSION')

plt.xticks([]), plt.yticks([])

plt.subplot(336),plt.imshow(closing),plt.title('closing')

plt.xticks([]), plt.yticks([])

plt.show()

#alerting the driver

pygame.init()

pygame.mixer.music.load("buzz.mp3")

pygame.mixer.music.play()

time.sleep(5)

#content ="detection of pothole in locality basapura road hosur road junction "

#mail = smtplib.SMTP('smtp.gmail.com',587)

#mail.ehlo()

#mail.starttls()

#mail.login('harika3196@gmail.com','hariammu3196@gmail.com')

#mail.sendmail('fromemail','receiver',content)

#mail.close()

import tkinter as tk

from tkinter import \*

from tkinter import messagebox as mb

from tkinter.ttk import \*

import numpy as np

import open3d as o3d

from cv2 import cv2

#heat map function

def heatMap():

img = cv2.imread('1.jpg', 1)

gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

heatmap\_img = cv2.applyColorMap(gray\_img, cv2.COLORMAP\_JET)

cv2.imshow('image',heatmap\_img,)

cv2.waitKey(0)

cv2.destroyAllWindows()

#point cloud data function

def pclView():

print("Load a ply point cloud, print it, and render it")

pcd = o3d.io.read\_point\_cloud("cloud.ply")

print(pcd)

print(np.asarray(pcd.points))

pcd.paint\_uniform\_color([1, 0.706, 0])

o3d.visualization.draw\_geometries([pcd])

def callback():

if mb.askyesno('Verify', 'Really quit?'):

mb.showwarning('Yes', 'Not yet implemented')

else:

mb.showinfo('No', 'Quit has been cancelled')

def tkinterUI():

root = tk.Tk()

root.geometry("240x240")

Label(root, text='Pothole Detection', font=(

'Verdana', 15)).pack(side=TOP, pady=10)

# close button config

closeIcon = PhotoImage(file=r"C:\Users\manojkumar\Desktop\Main Project\main projeect 1\implementation\PotholeDisparity\PotholeDisparity\close.png")

closeButIcon = closeIcon.subsample(15,15)

# cloud button config

cloudIcon = PhotoImage(file=r"C:\Users\manojkumar\Desktop\Main Project\main projeect 1\implementation\PotholeDisparity\PotholeDisparity\cloud.png")

cloudButIcon = cloudIcon.subsample(15,15)

#Quit Button

button = tk.Button(root, text="Quit", command=quit,

image=closeButIcon, compound=LEFT, font=('Verdana'))

button.pack(padx=5, pady=10, side=tk.BOTTOM)

# heatmap button config

heatMapIcon = PhotoImage(file=r"C:\Users\manojkumar\Desktop\Main Project\main projeect 1\implementation\PotholeDisparity\PotholeDisparity\heatmap.png")

heatMapButIcon = heatMapIcon.subsample(15,15)

#Point Cloud Button

button = tk.Button(root, text=" Point Cloud", command=pclView, image=cloudButIcon, compound=LEFT, font=('Verdana'))

button.pack(padx=5, pady=10, side=tk.BOTTOM)

#Heat Map Button

heatMapVar = tk.Button(root, text="Heat Map", command=heatMap,

image=heatMapButIcon, compound=LEFT, font=('Verdana'))

heatMapVar.pack(padx=5, pady=10, side=tk.BOTTOM)

root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

tkinterUI()

**APPENDIX III**

**PUBLICATION DETAIL**

|  |  |
| --- | --- |
| **TITLE** | POTHOLE DETECTION USING DISPARITY MAP ALGORITHM & CONVULUTION NEURAL NETWORK |
| **AUTHORS** | Dr. C. Punitha devi  Dr. G. Shanmugasundaram  Mr. S. Manojkumar  Mr. E. M. Mohammed Rafiyudheen |
| **STATUS** |  |
| **PLACE** |  |
| **DATE** |  |

|  |  |  |
| --- | --- | --- |
| **MEDIA** | **IEEE CATALOG NUMBER** | **ISBN** |
| EXPLORE |  |  |
| DVD |  |  |