Department of Electronics and
Electrical Communications Engineering
Faculty of Engineering - Cairo University

[Advanced driver assistant system (bump detection)]

Graduation thesis

By

Alaa Ibrahim
Neama ahmed
Nourhan karem
Hend atef
Hend fekry

Under supervision of:

Prof:Hassan mostafa
Prof: Amin nasar
Prof : Samah El-shafiey

Date of Submission
17/8/2020

Communications and electronics department
Faculty of engineering
Cairo university

List of Figures

No.	Figure	Page No.
1	Speed bump	3
2	System block diagram	6
3	Convolutional Neural Networks (CNN)	9
4	Single Shot Detector SSD.	12
5	RCNN.	12
6	Fast RCNN.	13
7	Faster RCNN.	14
8	*U-Net architecture*	16
9	*Comparison of different scaling methods*	18
10	*IOU score*	22
11	*Loss*	22
12	Distance Estimation with 2 cameras	30
13	depth map using two cameras	31
14	Horizon line	33
15	Distance estimation using Horizon	34
16	Mapping system block diagram	37
17	Mapping system components	38
18	starting map	43
19	Detecting bump	43

List of Tables

No.	Table	Page No.
1	List of Abbreviation	2
2	deaths and injured number in 2017&2018.	3
3	Scheduling table	5
4	performance of the output trained Unet segmentation model	22

Abstract

Most of the speed bumps in Egypt are not being constructed and maintained according to the public safety guidelines. These bumps may cause car crashes or accidents, severe discomfort to the driver and even causing loss of direction control which is leading to fatalities although speed bumps are constructed to force drivers reduce car speed to avoid accidents

Therefore, in this project, we propose a method that can help drivers in detecting the speed bump, estimating distance to speed bump and serving as a part of the advanced driver assistance systems (ADAS) for autonomous vehicle.
A small device "Raspberry Pi" is using to analyze the images and scenes with the help of the camera, which moves the images to the Raspberry Pi. Then, the process of analysis begins through long complex algorithms known as the deep neural network algorithms. This network analyzes the images to extract most important characteristics of the objects. When ensuring that the characteristics match the mathematical equations programmed in the language of the Python, the bump is detected finally and inform the driver about an upcoming unmarked and marked speed hump /bump in real time; two simple algorithms are used to estimate the approximate distance between the vehicle and the bump categorize it in ranges which helps the driver to control the vehicle speeds to be at safer limits in order to not cause any kind of discomfort to the passengers as well as damage to the vehicle ; and a simple Android application phone is used to sense position of bump when car is passing above then send the location of the bumps to server to collect database bumps which help drivers in next time passing bump by sending indicator of bump location on an offline Google maps.

Content

Chapter One: Introduction

1.1 Overview ……………………………………………………..… 2

1.2 Project Motivation ……………………………………….…. 2

1.3 Project Aims ………………………………………………… 2

1.4 Project idea and Importance …………………………………. 2

1.5 Literature Review ……………………………………….... 3

1.6 List of Abbreviation…………………………………………… 4

1.7 Scheduling table ……………………………….…………5

1.8 1.8 block diagram……………………………………………….5

Chapter Two: Object Detection & Segmentation with Tensor flow

2.1 Introduction …………………………………………………….8

2.**2** Tensor Flow …………………………………………………….8

2.2 Tensor Flow …………………………………………………….8

2**.**3 Why Tensor Flow? ……………………………………………….8

2.4 Neural Network ………………………………………………..….8

2.5 Object Detection with Tensor Flow ……………………………..9

2.5.1 Gathering a dataset……………………………………………9

2.5.2 Creating bounding boxes………………………………10

2.5.3 Install the object detection API……………………………11

2.5.4 Convert labels to The Tensor Flow Record format………11

2.6 Detection Models ………………………………………………11

2.6.1 Single Shot Detector

(SSD)……………………………………………………………….11

2.6.2 RCNN ……………………………………………...……12

2.6.3 Fast RCNN ……………………………………………13

2.7 Model Selection ……………………………………….……. 14

2.8 Train & Results…………….…………...……………………15

2.9 segmentation ………………………………………………………16

2.10 U-Net architecture………………………………………………..16

2.11 backbone network…………………………………………..18

2.12 transfer learning…………………………………………18

2.13 data augmentation……………………………………19

2.14 Hyperparameters choosing……………………………..20

2.15 Results…………………………………………………22

Chapter Three: Distance Estimation

3.1 Introduction………………………………………………**27**

3.2 Calibration………………………………………………….**27**

3.3 stereo visions…………………………………………………..28

3.3.1 Zed stereo camera……………………………………….. 28

3.3.2 Two identical cameras………………………………..28

3.4 single camera ……………………………………………………32

3.4.1 Deep learning model…………………………………32.

3.4.2 Simple Algorithm with known feature from object.. ……………………………………………………………………..32

3.4.3 Simple Algorithm using Horizon of image……………………..33

Chapter Four: MAPPING BUMPS LOCATION IN OFFLINE GOOGLE MAPS

4.1 Introduction …………………………………………………………...36

4.2 Literature……………………………………………………………….36.

4.3 System Architecture……………………………………………….37

4.4 System Architecture in details……………………………………38

4.6 Challenges ……………………………………………………………39

4.7 Methodology…………………………………………………………39

4.7.1 VCEH………………………………………………………………..39

4.7.2 Bumping Events Detection Algorithm……………………...41

4.7.3 Bumping index…………………………..………………...42

5.7.4 Data mining……………………………..........................42

4.8 Application functions……………………………………………42

Chapter Five: Future work

5.1 Merging segmentation, distance estimation and mapping blocks

……………………………………………………………………………..45

5.2 challenges ………………………………………………….……. 45

5.3 conclusion & future work ………………………………..…….. 46

5.3.1 Segmentation model………………………………………46

5.3.2 Distance Estimation…………………………………………46

5.3.3 Mapping…………………………………………………..46

Appendix

A

CHAPTER ONE

1

Introduction

1.1 Overview

1.2 Project Motivation

1.3 Project Aims

1.4 Project idea and Importance

1.5 Literature Review

1.6 List of Abbreviation

1.7 Scheduling table

1.8 block diagram

1.1 Overview

The aim of project bump detection is to help the drivers to avid the risks of bumps , protect vehicles from damages and reduce road accidents with a system that detect the bump at a certain distance and using server to set a database of all bumps in Egypt.

1.2 List of Abbreviation

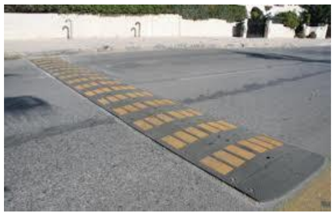
Abbreviation	Full Meaning
GPS	Global Positioning System
TF	Tensor flow
ssd	Single shot detector
CNN	Convolution neural network
API	Application Programming Interface
VCEH	Vertical component extraction heuristics
BI	Bumping index

Table 1: List of Abbreviation

1.3 Project Motivation

There are many speed bumps in the streets and roads of Egypt, the commonly observed and we rarely observe are rumble strips are shown in fig.1.
Quality and metrology organizations have set several conditions for creating speed bumps:
1.the area where the bumps are to be placed is chosen due to the large number of traffic accidents and in places where there are children and a lot of pedestrians.
2. Bumps are placed on roads whose speed is less than 60km/h.
3. They should be placed on lit roads.
4. They should be designed with length of 6.6m and a height ranging between 5.7 and 10cm.

Unmarked speed bump


Rumble strips

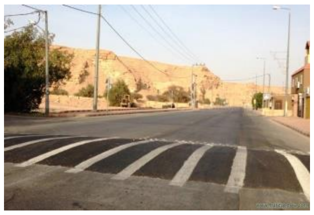

Marked speed bump

Fig1.speed bumps

Statistics indicate that 40% of road accidents in Egypt occur due to random bumps which doesn't match specifications, and that 50 people fall as victims of road accidents for every 100 kilometers covered by cars in Egypt, and the reason is due to dangerous curves and random bumps.

year	2017	2018
Deaths number	1929	1560
Injured number	7217	9563

Table 2. deaths and injured number in 2017&2018.
source: central agency for public mobilization and statistics.

Part from crashes and deaths, there are many problems which public face during their day-to-day commute due to travel through roads having non-standardized speed bumps. These are given as follows:

- Causing damage to vehicles.
- Causing discomfort and back injury to drivers and passengers.
- Causing vibration when vehicles navigate them and send shockwaves through the ground.
- Causing accidents.

1.4 Project Aims

The main objective of the project is to design and implement a smart system to detect bump location and alert driver.

Connect the camera with a Raspberry Pi.

Programing the Raspberry Pi using Python language, its powerful for processing. Process and analyze the camera records using Raspberry Pi in real time.

Detect bumps in front of the vehicle.

Estimate location between vehicle and bump.

Define best location of bump using GPS sensor of mobile phone

Create database bump location on server

Create a simple Android Application to send location of bumps

Design and build an alarm system to notify the user about the upcoming bumps.

1.5 Project idea and Importance

We propose a method that detects and informs the driver about the speed bump in real time using deep learning technique and vibration approach technique and gives the distance the vehicle is away from it.
With this driver or autonomous mode of the vehicle can control the vehicle speeds to be at safer limits in order to not cause any kind of discomfort to the passengers as well as damage.

1.6 Literature Review

According to literature there are three used techniques to detect bumps:

3-Dreconstruction based method:

Utilizes 3D laser /LiDar scanning that provides accurate 3D point clouds to measure elevation on the asphalt pavement, it offers outstanding performance with high accuracy of measurement compared to the other two approaches. However, it is not widely use due to its high cost and narrow range of detection.

Vibration-based method:

It uses gradient variation from accelerometer data, this method have been employed for uneven road surface detection due to its low cost and relatively simple detection algorithms. Nevertheless, the detection path for vibration-based method is only limited to the path of the car’s wheels. Furthermore, false detection may occur when the car pass over other objects present on the road.

Vision-based methods:

They are appropriate for their accuracy as well as relatively wide area detection for pothole and speed bumps at low cost. However it cannot be used under bad illumination or no light conditions.

1.7 Scheduling table:

activities	Months											
	1	2	3	4	5	6	7	8	9	10	11	12
Bump detection	🟪	🟪										
segmentation			🟪	🟪	🟪	🟪						
Distance estimation							🟪	🟪				
mapping									🟪	🟪		

Table 3: Scheduling table

1.8 block diagram:

the system gets the road information using monocular camera and smart phone sensors (accelerometer sensor and GPS receiver sensor).
based on image from camera vision we detect the speed bumps and estimate the distance between the vehicle and the bump then print out to the driver on the LCD that there is a bump with the estimated distance meters away. while the accelerometer senses the variance in acceleration so we can determine the location of the bump then upload this location via WI-FI to cloud server.
server will send back the exact location on google maps after processing the sent data.

preprocessing data block:

In the training phase the block preprocess on the collected dataset (data clean, data augmentations, data annotation ,..etc)to train the U-Net model.
In the operating phase the block get the input from camera sample it into 10 fps to send it frame by frame to the bump detection block.

bump detection module:

determines if the current frame contains a bump or not if bump detected the image and the generated mask send to the distance estimation module.

distance estimation module:

runs two different algorithms to estimate the distance from the bump then averages the outputs from the two algorithms to improve the accuracy.

obtain the vehicle velocity and sends the estimated distance and velocity to the mapping block sends the estimated distance to the decision block.

mapping bump location module:

gets the input from smart phone sensors analyzes it if the bump is detected sends its location to the server via wi-fi to mine road bumps and assigns it to google maps. deals with the input from distance estimation object.

the system functions as shown block diagram

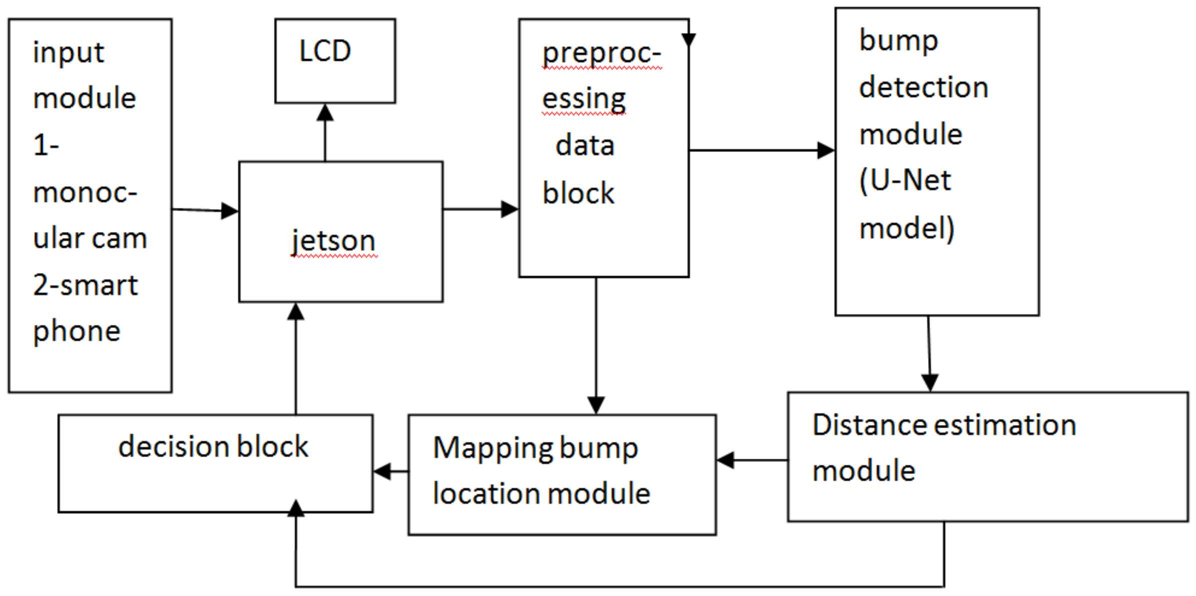

input module 1-monoc-ular cam 2-smart phone
LCD
jetson
preproc-essing data block
bump detection module (U-Net model)
decision block
Mapping bump location module
Distance estimation module

fig 2: system block diagram

CHAPTER TWO

2

Object Detection & Segmentation Using Tensor Flow

2.1 Introduction
2.2 Tensor Flow
2.3 Why Tensor Flow?
2.4 Neural Network
2.5 Object Detection with Tensor Flow
- **2.5.1 Gathering a data set**
- **2.5.2 Creating bounding boxes**
- **2.5.3 Install the object detection API**
- **2.5.4 Convert labels to The Tensor Flow Record format**

2.6 Detection Models
- **2.6.1 Single Shot Detector (SSD)**
- **2.6.2 RCNN**
- **2.6.3 Fast RCNN**
- **2.6.4 Faster RCNN**

2.7 Model selection
2.8 Train & Results
2.9 segmentation
2.10 U-Net architecture
2.11 backbone network
2.12 transfer learning
2.13 data augmentation
2.14 Hyperparameters choosing
2.15 Results

2.1 Introduction

Computer vision is an interdisciplinary field that deals with how computers can be made for gaining high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do. Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding.
One of the primary goals of computer vision is the understanding of visual scenes. Scene understanding involves numerous tasks including localizing the objects in 2D and 3D, determining the objects' and scene's attributes, characterizing relationships between objects and providing a semantic description of the scene.

2.2 Tensor Flow

Tensor Flow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

2.3 Why Tensor Flow

Python API
Portability: deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.
Flexibility: from Raspberry Pi, Android, Windows, IOS, Linux to server farms.
Auto-differentiation (no more taking derivatives by hand)
Large community (> 10,000 commits and > 3000 TF-related repos in 1 year).
Awesome projects powerful using Tensor Flow.

2.4 Neural Network

Neural Network commonly referred to as "Neural Networks" has been motivated right from its inception by the recognition that the human brain computes in an entirely different way from the conventional digital computer. The brain is a highly complex, nonlinear, and parallel computer (information -processing system). It has the capability to organize its structural constituents, known as neurons, so as to perform certain computations (e.g., pattern recognition, perception, and motor control) many times faster than the fastest digital computer in existence today. Consider, for example, human vision, which is an information -processing task. It is the function of the visual system to provide a representation of the environment around us and more important, to supply the information we need to interact with the environment. To be specific, the brain routinely accomplishes perceptual recognition tasks (e.g., recognizing) [.
Neural Networks help us cluster and classify. You can think of them as a clustering and classification layer on top of the data you store and manage. They help to group unlabeled data according to similarities among the example inputs, and they classify data

when they have a labeled dataset to train on. (Neural networks can also extract features that are fed to other algorithms for clustering and classification, so you can think of deep neural networks as components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression.)
The Convolutional Neural Networks (CNNs), an important and powerful kind of learning architecture widely diffused especially for Computer Vision applications. They currently represent state of the art algorithm for image classification tasks and constitute the main architecture used in Deep Learning.

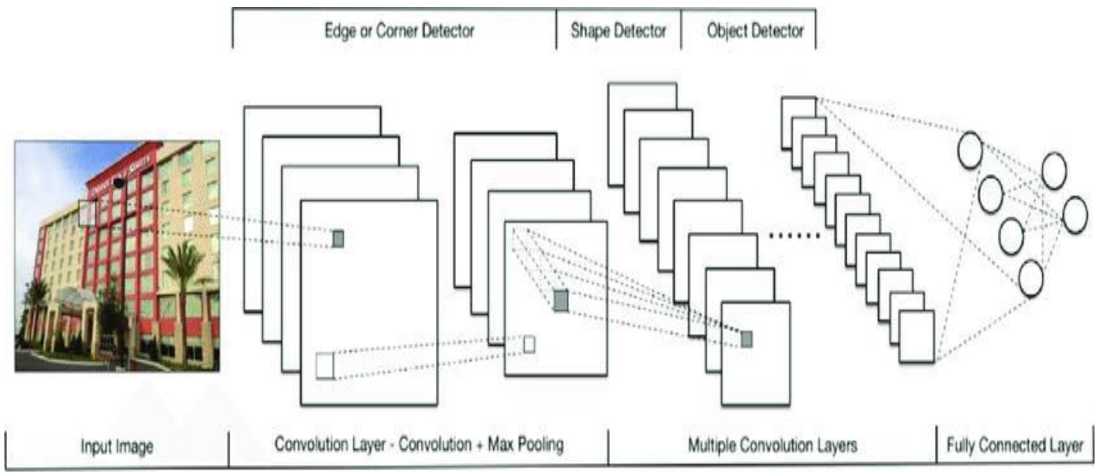


Figure **3**: Convolutional Neural Networks (CNN).

2.5 Object detection with Tensor Flow

we will walk through all the steps for building a custom object classification model using Tensor Flow's API**:**

2.5.1 Gathering a data set:

The data set is about 7000 images (5000 images containing bumps and
2000 images with no bumps.
The data can be divided as:

- There were 500 images of speed bump found available Mendeley Data.
- We have collected around **1800** mages of bump across various locations in and around the institution campus during various time of the day to capture dataset with variety of viewing and illumination conditions.
- Another team collected data of bumps we used around **2500** images
- There are **2000** images for un bumped.

2.5.2 Creating bounding boxes:

In order to train our object detection model, for each image we will need the image's width, height, and each class with their respective xmin, xmax, ymin, and ymax bounding box. Simply put, our bounding box is the frame that captures exactly where our class is in the image.
Creating these labels can be a huge ordeal, but thankfully there are programs that help create bounding boxes. LabelImg is an excellent open source free software that makes the labeling process much easier. It will save individual xml labels for each image, which we will convert into a csv table for training.

2.5.3 Install the object detection API:

Before getting started, we have to clone and install the object detection API into our Get Hub repository. Installing the object detection API is extremely simple; you just need to clone the Tensor Flow Models directory and add some things to your Python path.

2.5.4 Convert labels to The Tensor Flow Record format:

When training models with Tensor Flow using Tensor Flow Record, files help optimize your data feed. We can generate a Tensor Flow Record file using code adapted from this raccoon detector.

2.6 Detection Models

2.6.1 Single Shot Detector SSD:

The single shot name was developed because SSD is a single-stage detector. It doesn't follow the RCNN approach of having two separate stages for each of regions proposal and detections. The SSD approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections.

The architecture of the SSD model is composed of three main parts:

1. Base network to extract feature maps - a standard pretrained network used for high quality image classification and truncated before any classification layers. In their paper, C. Szegedy et al. used VGG16 network. Other networks like VGG19 and ResNet can be used and should produce good results.
2. Extra feature layers - a series of convolution filters are added after the base network. These layers decrease in size progressively to allow predictions of detections at multiple scales.
3. Non-maximum suppression - to eliminate overlapping boxes and keep only one box for each object detected.

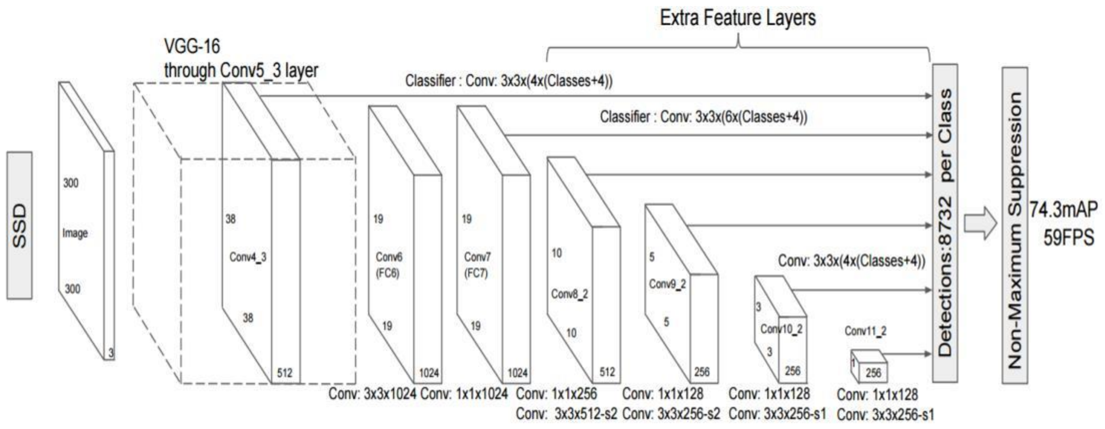


Figure **4**: Single Shot Detector SSD.

2.6.2 RCNN (Region Proposal + CNN)

The Region-based Convolutional Network method (RCNN) achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. R-CNN. Use selective search to come up with regional proposal First object detection method using CNN

.

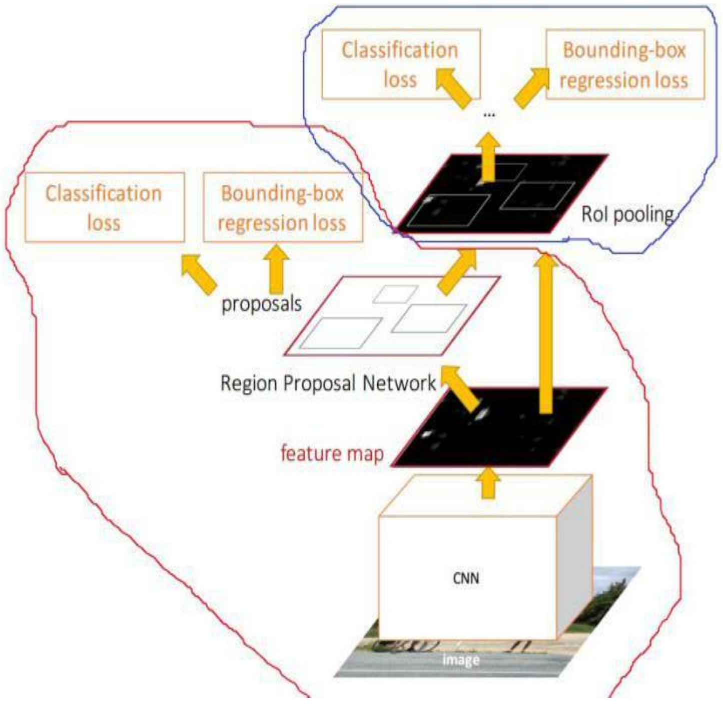


Figure 5: RCNN.

2.6.3 Fast RCNN

Faster RCNN is the follow on to Fast RCNN and RCNN. Faster RCNN starts with a CNN adds a Region Proposal Network (RPN) to create proposals (bounding boxes) from the features given by the CNN. Then ROI pooling and a classifier is used to classify and score each bounding box .Once a Fast R-CNN network is fine-tuned, detection amounts to little more than running a forward pass (assuming object proposals are pre-computed). The network takes as input an image and a list of R object proposals to score.

Share convolution layers for proposals from the same image Faster and More accurate than RCNN.

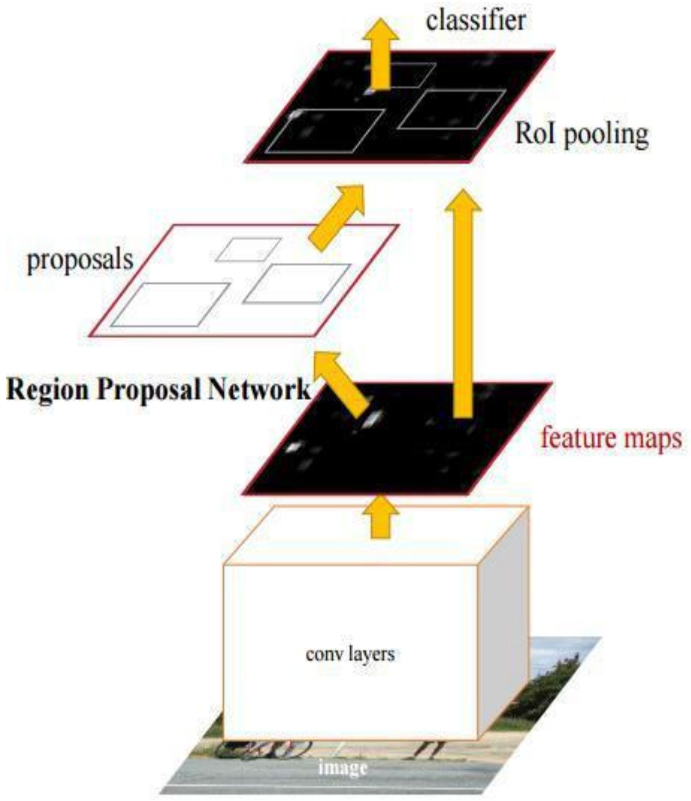


Figure **6**: Fast RCNN.

2.6.4 Faster RCNN

The architecture of Faster R-CNN can be described by two main networks:

1. Region Proposal Network (RPN) - selective search is replaced by a ConvNet that to propose regions of interest (RoI) from the last feature maps of the feature extractor to be considered for investigations. RPN has two outputs; the “objectness score” (object or no object) and the box location
2. Fast R-CNN - consists of the typical components of Fast R-CNN:
   a. Base network for Feature extractor: a typical pre-trained CNN model to extract features from the input image
   b. ROI pooling layer: to extract fixed-size regions of interest.
   c. Output layer: contains 2 fully-connected layers:
   1) a softmax classifier to output the class probability, and
   2) a bounding-box regression CNN to the bounding box predictions.

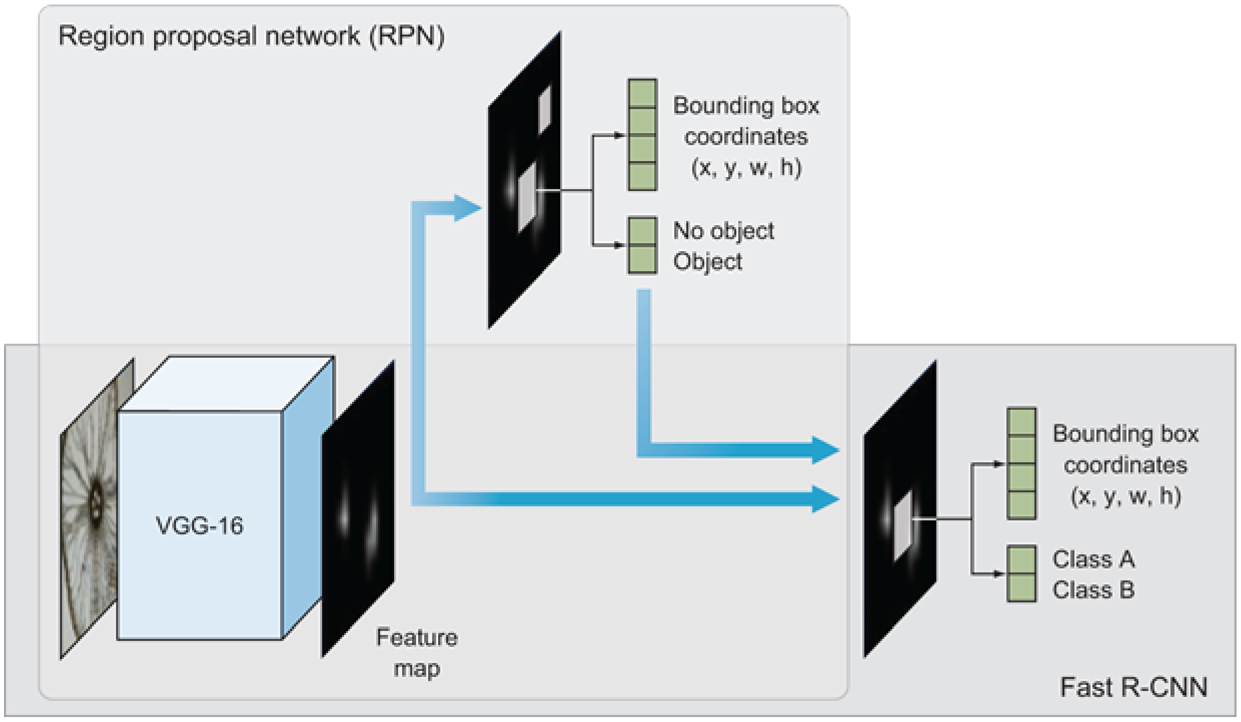

Region proposal network (RPN)
Bounding box coordinates (x, y, w, h)
No object
Object
VGG-16
Feature map
Bounding box coordinates (x, y, w, h)
Class A
Class B
Fast R-CNN

Figure 7: Faster RCNN.

2.7 Choose a model:

1) Instead of generating the regions of interest and classifying the regions separately, SSD does it simultaneously, in a “single shot”.
2) It is faster in inference with not much reduction with respect to mean average precision (mAP) tested on COCO dataset.
3) SSD detection model has better accuracy.

2.8 Train & Results:

Model was trained with CoCo dataset that detect 80 different objects

Froze layers and change last.

train_config:

```
   batch_size: 16
  rms_prop_optimizer:

   learning_rate:
    exponential_decay_learning_rate
            {
     initial_learning_rate: 0.0001
     decay_steps: 800720
     decay_factor: 0.95
            }
classification_loss { weighted cross enteropy }
```

Model has large over fitting, test loss is 3 and the train loss is 2.5

the reasons of that are:

1)our data isn't variant and few.
2)much layers in network.

fine tuning:

1)Dropout keeping parameters (0.9-0.6)
2)Reducing learning rate ((10)^(-4) _(10)^(-6)).
3)Frozen layers

But we fail to improve the test loss which the main reason is our data isn't variant and few .Hence we tried to accomplish the detection with another deep learning technique using segmentation model one of the advantage of segmentation model over ssd-mobilenet v2 model it classify the input image pixel-wise into two classes the foreground class (speed bump) and the rest of image into the background class which facilitates the bump feature extracting and we chose a model that doesn't need large number of data.
the segmentation model acquired a promising results and fitted the criteria of real time application so let's dive into it.

2.9 Segmentation

based on a method that detects the upcoming bump by using a deep learning algorithm called U-Net[1], which is a deep CNN architecture for semantic pixel-wise segmentation. The trained model will give segmented output from the monocular camera feed placed in front of the vehicle.
U-Net architecture

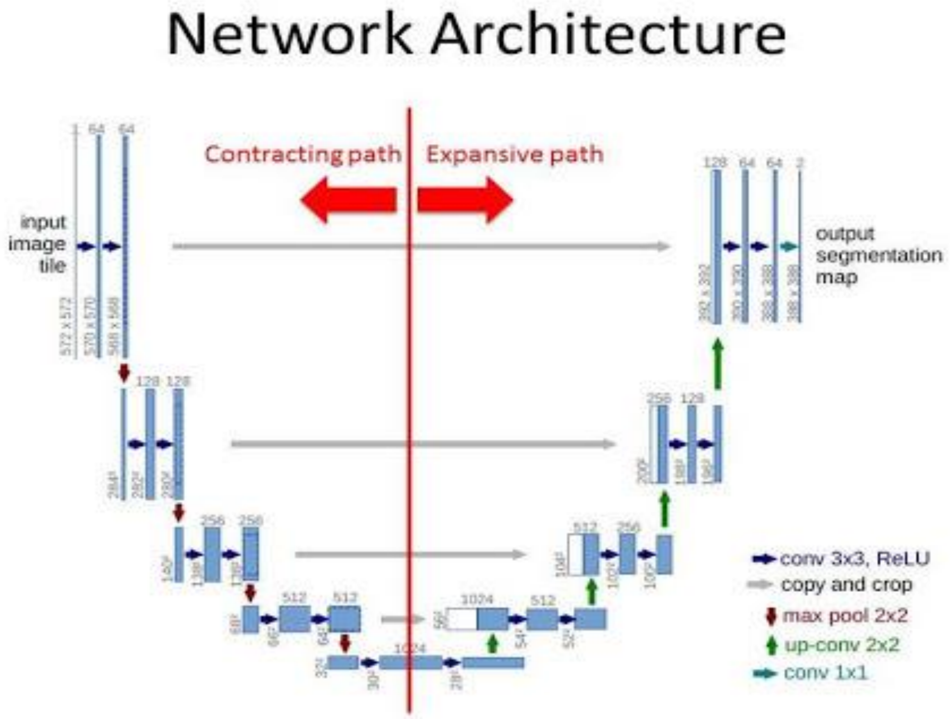


fig 8:U-Net architecture

The U-Net architecture is built upon the Fully Convolutional Network and modified in a way that it yields better segmentation. it consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. it works with very few training images and yields more precise segmentations.
The main idea in fully convolutional network is to supplement a usual contracting network by successive layers, where pooling operators are replaced by upsampling operators. Hence, these layers increase the resolution of the output. In order to localize, high resolution features from the contracting path are combined with the upsampled output. A successive convolution layer can then learn to assemble a more precise output based on this information. One important modification in U-Net architecture is that in the upsampling part it has also a large number of feature channels, which allow the network to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting path, and yields a u-shaped architecture. The network does not have any fully connected layers and only uses the valid part of each convolution, i.e., the segmentation map only contains the pixels, for which the full context is available in the input image. This strategy allows the seamless segmentation of arbitrarily large
images by an overlap-tile strategy. To predict the pixels in the border region of the image, the missing context is extrapolated by mirroring the input image. This tiling strategy is important to apply the network to large images, since otherwise the resolution would be limited by the GPU memory.

Model

U-Net architecture is separated in 3 parts:

- The contracting/downsampling path
- Bottleneck
- The expanding/up sampling path

Contracting/down sampling path

The contracting path is composed of 4 blocks. Each block is composed of

- 3x3 Convolution Layer + activation function (with batch normalization)
- 3x3 Convolution Layer + activation function (with batch normalization)
- 2x2 Max Pooling

Note that the number of feature maps doubles at each pooling, starting with 64 feature maps for the first block, 128 for the second, and so on. The purpose of this contracting path is to capture the context of the input image in order to be able to do segmentation. This coarse contextual information will then be transfered to the upsampling path by means of skip connections.
we replaced this part with Efficientnet-b3[2] as a backbone network which we will introduce it in short.

Bottleneck

This part of the network is between the contracting and expanding paths. The bottleneck is built from simply 2 convolutional layers (with batch normalization), with dropout.

Expanding/upsampling path

The expanding path is also composed of 4 blocks. Each of these blocks is composed of

- Deconvolution layer with stride 2
- Concatenation with the corresponding cropped feature map from the contracting path
- 3x3 Convolution layer + activation function (with batch normalization)
- 3x3 Convolution layer + activation function (with batch normalization)

The purpose of this expanding path is to enable precise localization combined with contextual information from the contracting path.

Advantages

- The U-Net combines the location information from the down sampling path with the contextual information in the up sampling path to finally obtain a general information combining localization and context, which is necessary to predict a good segmentation map.
- No dense layer, so images of different sizes can be used as input (since the only parameters to learn on convolution layers are the kernel, and the size of the kernel is independent from input image' size).
- The use of massive data augmentation is important in case of our limited dataset.

Backbone network

CNNs are commonly developed at a fixed resource cost, and then scaled up in order to achieve better accuracy when more resources are made available. For example, ResNet can be scaled up from ResNet-18 to ResNet-200 by increasing the number of layers. The conventional practice for model scaling is to arbitrarily increase the CNN depth or width, or to use larger input image resolution for training and evaluation. While these methods do improve accuracy, they usually require tedious manual tuning, and still often yield suboptimal performance, Hence in efficientnet they found a more principled method to scale up a CNN to obtain better accuracy and efficiency. they proposed a novel model scaling method that uses a simple yet highly effective *compound coefficient* to scale up CNNs in a more structured manner. Unlike conventional approaches that arbitrarily scale network dimensions, such as width, depth and resolution, this method uniformly scales each dimension with a fixed set of scaling coefficients. Powered by this novel scaling method and recent progress on Auto Machine Learning .The first step in the compound scaling method is to perform a grid search to find the relationship between different scaling dimensions of the baseline network under a fixed resource constraint (e.g., 2x more FLOPS).This determines the appropriate scaling coefficient for each of the dimensions then apply those coefficients to scale up the baseline network to the desired target model size or computational budget.

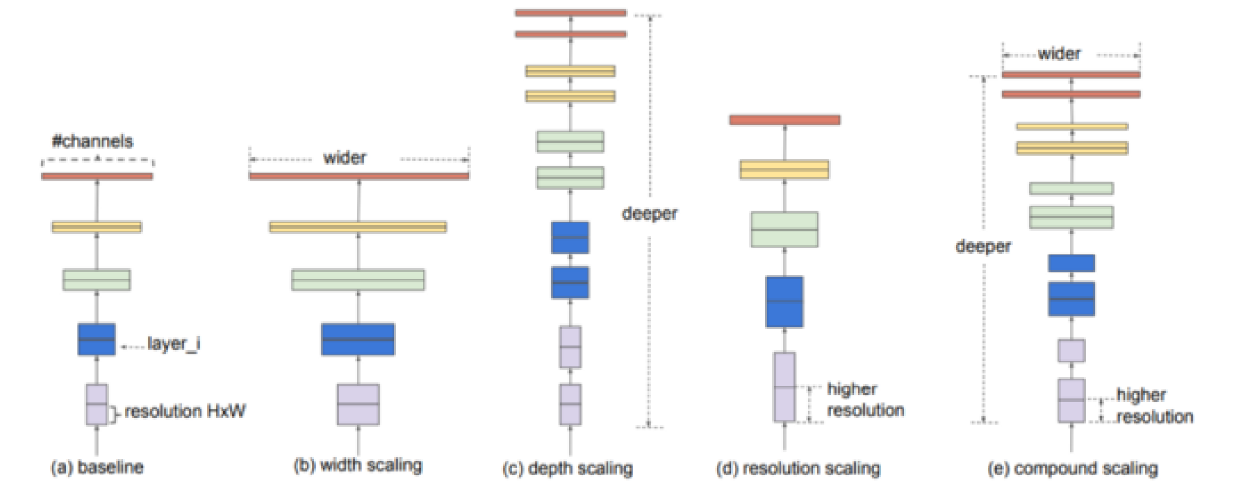


fig 9: Comparison of different scaling methods

Efficientnet-b3 choosing criteria:

- It is suitable for real time application due to smaller numbers of parameters (12 M) and the number of FLOPS required (1.8 B) yet it yields a good accuracy.

Transfer learning

Transfer learning[3] is the idea of overcoming the isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones. there is two different approaches to achieve this task

- Fine Tuning Pre-trained Models
- Pre-trained Models as Feature Extractors

Pre-trained Models as Feature Extractors

Deep learning systems and models are layered architectures that learn different features at different layers (hierarchical representations of layered features). These layers are then finally connected to a last layer (usually a fully connected layer, in the case of supervised learning) to get the final output. This layered architecture allows us to utilize a pre-trained network (such as Inception V3 or VGG) without its final layer as a fixed feature extractor for other tasks.
The key idea here is to just leverage the pre-trained model's weighted layers to extract features but not to update the weights of the model's layers during training with new data for the new task. This is one of the most widely utilized methods of performing transfer learning using deep neural networks.

Fine Tuning Pretrained Models

This is a more involved technique, where we do not just replace the final layer (for classification/regression), but we also selectively retrain some of the previous layers. Deep neural networks are highly configurable architectures with various hyperparameters. the initial layers have been seen to capture generic features, while the later ones focus more on the specific task at hand.
Using this insight, we may freeze (fix weights) certain layers while retraining, or fine-tune the rest of them to suit our needs. In this case, we utilize the knowledge in terms of the overall architecture of the network and use its states as the starting point for our retraining step. This, in turn, helps us achieve better performance with less training time.
In our project, we chose the U-Net Pre-trained Models on cam-vid dataset as Feature Extractor as cam-vid dataset (an outdoor collected dataset contains 11 classes such as road, building, cars, poles etc) is a quite similar to our dataset .also to avoid overfitting.

Data augmentation

we applied multiple methods of data augmentation in order to increase our training data and to increase the system robustness.
Horizontal flip augmentations:
Reversing the entire rows and columns of an image pixels in horizontally

1-Shift augmentation:

Shifting the entire pixels of an image from one position to another position is called as shift augmentation

We have two types of shift augmentation.

Horizontal shift augmentation and Vertical shift augmentation

2- padding:

In padding, the image is padded with a given value on all sides.

3- Random Crop

we create a random subset of an original image. This helps our model generalize better because the object of interest we want our models to learn are not always wholly visible in the image or the same scale in our training data.

Hyperparameters choosing

We trained the model using:

- Learning rate =0.0001 (has been chose a relative small value to avoid the overfitting)
- Adam optimizer
- F1-score metric
- IOU-score metric
- Combined focal and dice loss functions with same ratio
- 40 epoch

Adam-optimizer

The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

Adam-optimizer advantages:

- Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models.
- Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.
- Adam is relatively easy to configure where the default configuration parameters do well on most problems.

Loss functions:

As there is an imbalance between the foreground class (bump class) and the background we intended to use dice loss

- The dice coefficient is a measure of overlap of the predicted mask and the ground truth
- it only considers the segmentation class and not the background class. The pixels are classified as True Positive (TP), False Negative (FN) and False Positive (FP).

$$\mathrm{DSC} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FN} + \mathrm{FP}}$$

it outputs a score in the range [0,1] where 1 is a perfect overlap. Thus, (1-DSC) can be used as a loss function

- Pros: Easy solution to class imbalance without having to manually optimize any parameters

Cons: Potential for gradient explosion (this is easily manageable using batch normalization and ReLUs) , and generally slower to train than Binary cross entropy.
F1-score
F1-score is a metric which takes into account both ,precision and recall as we can’t always evaluate both and then take the higher one for our model. It is the harmonic mean of precision and recall. It tells us about the balance that exists between precision and recall.
Precision basically tells us that out of the results classified as positive by our model, how many were actually positive.
Recall tells us how many true positives (points labelled as positive) were recalled or found by our model.
F1-score gives equal weight to both, the precision and the recall.

$$\textbf{Precision} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Positives}}$$

$$\textbf{Recall} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Negatives}}$$

Results:

the performance of the model is very promising as shown below

loss	Mean IOU score	Mean F1-score
0.36047	0.77471	0.84867

Table 4 : performance of Unet trained model

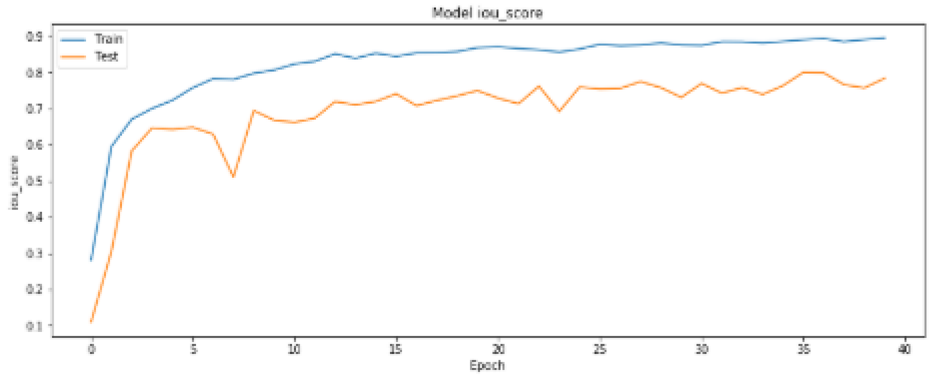


fig 10: IOU score

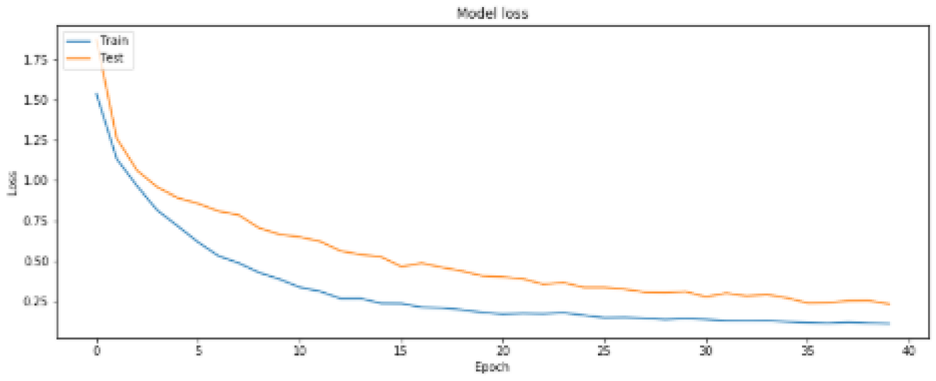


fig 11: Loss

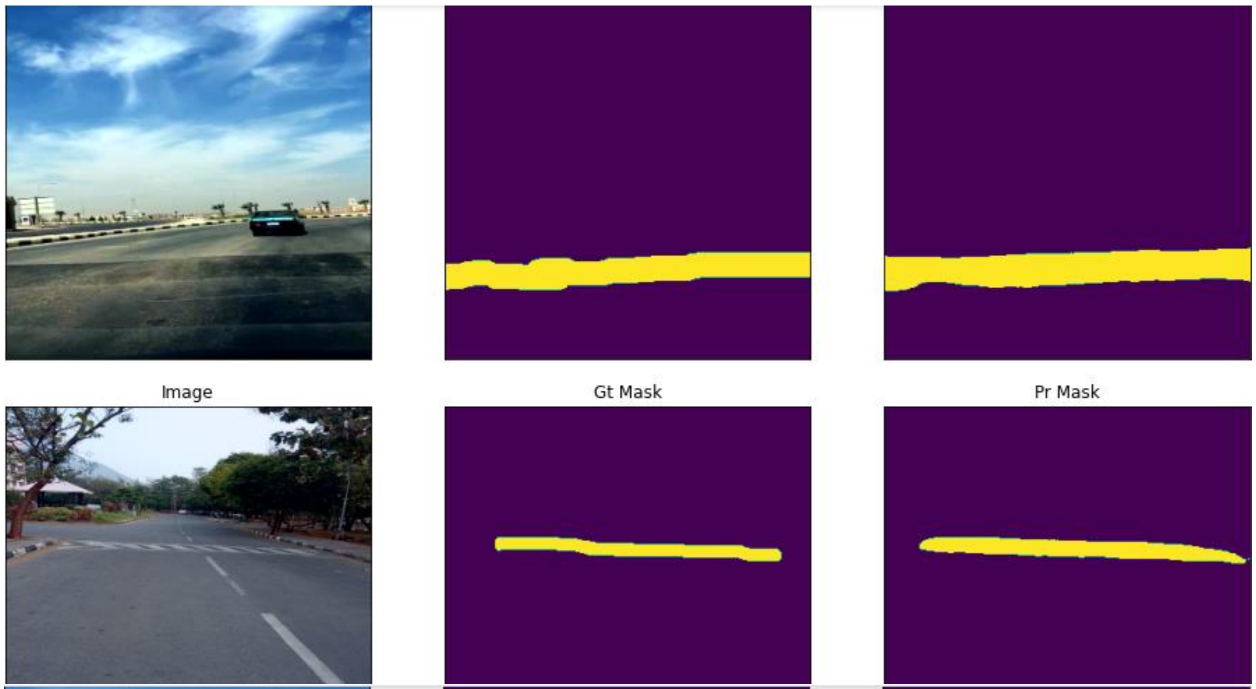


Image
Gt Mask
Pr Mask

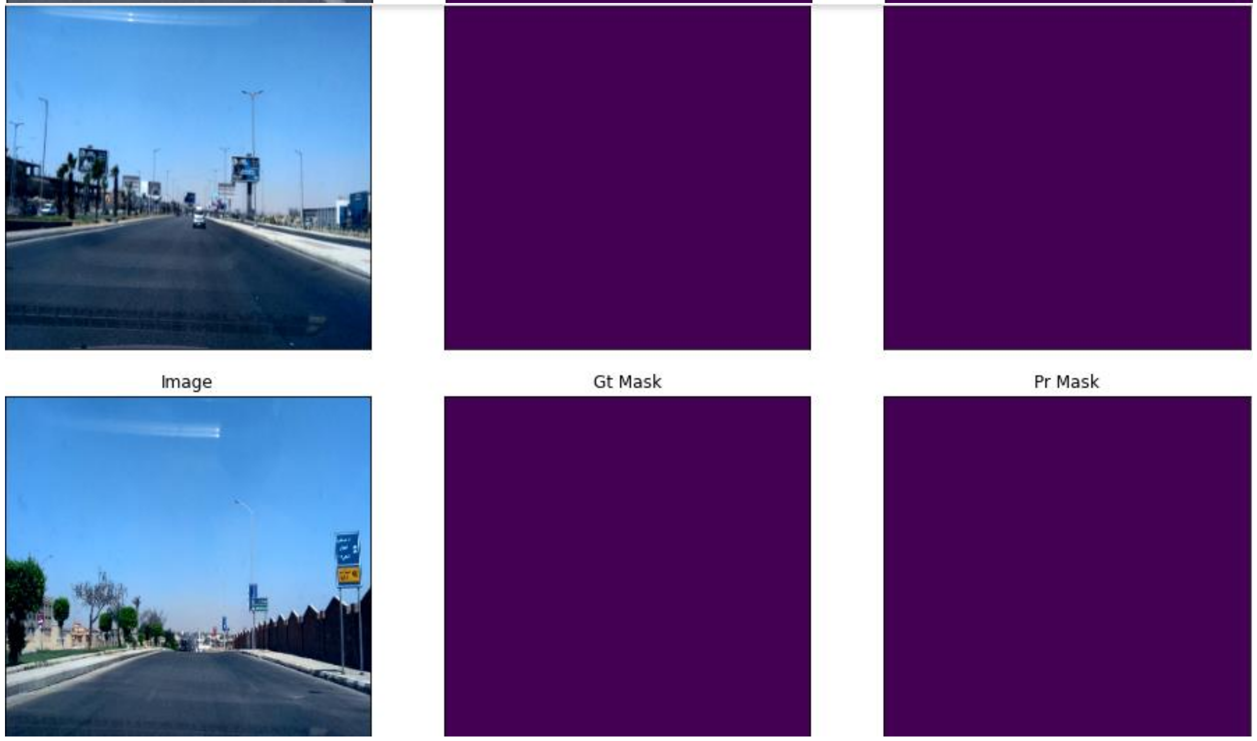


Image
Gt Mask
Pr Mask

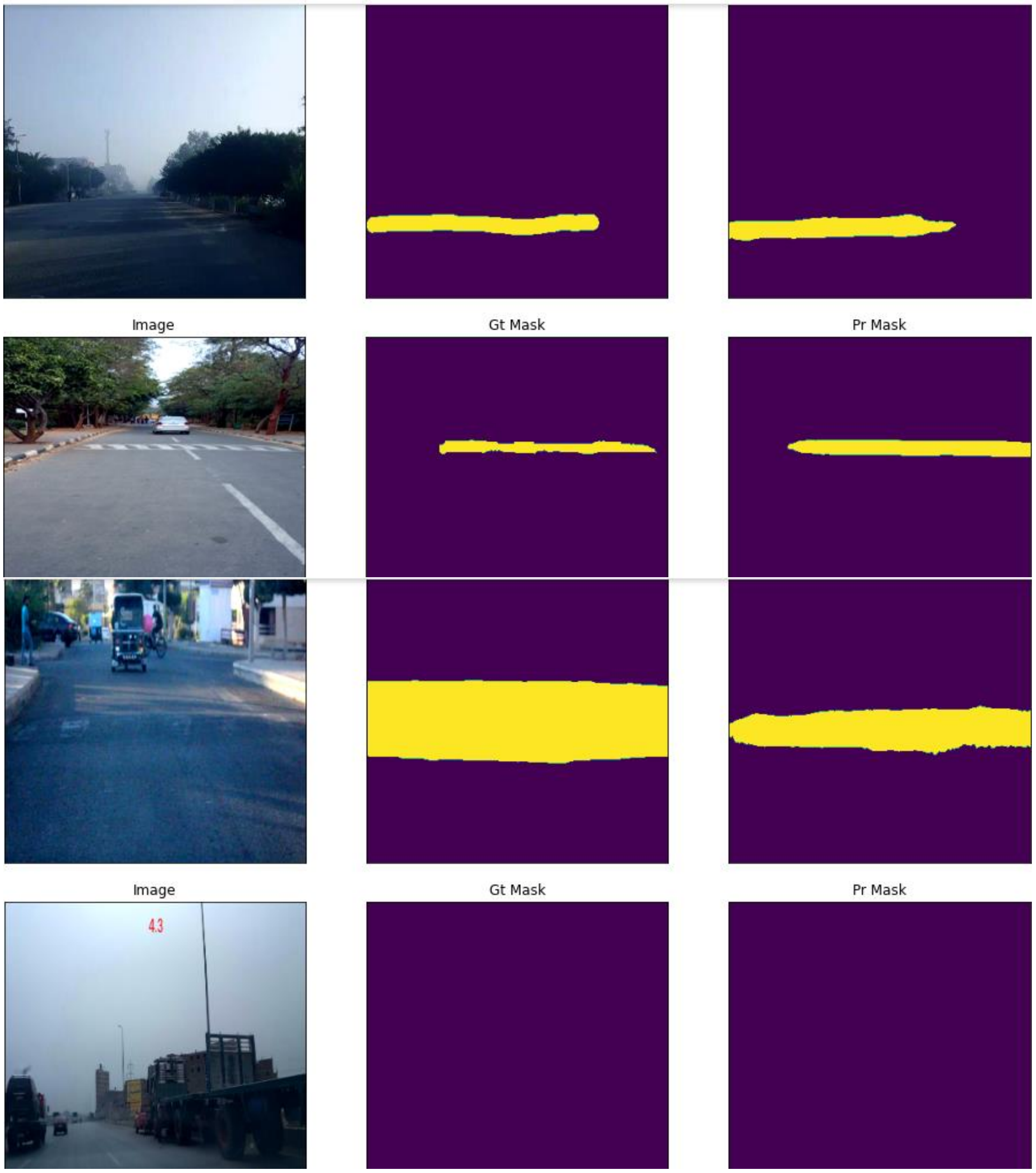


Image
Gt Mask
Pr Mask
Image
Gt Mask
Pr Mask
4.3

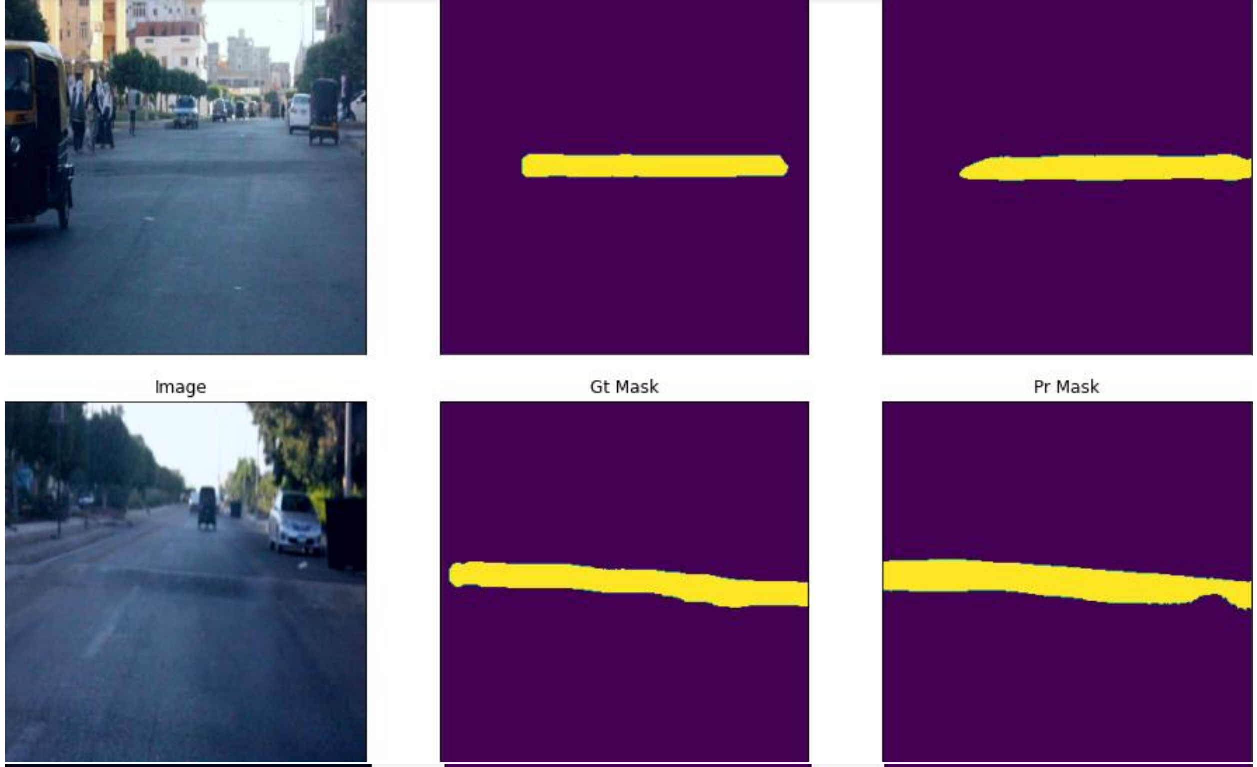


Image
Gt Mask
Pr Mask

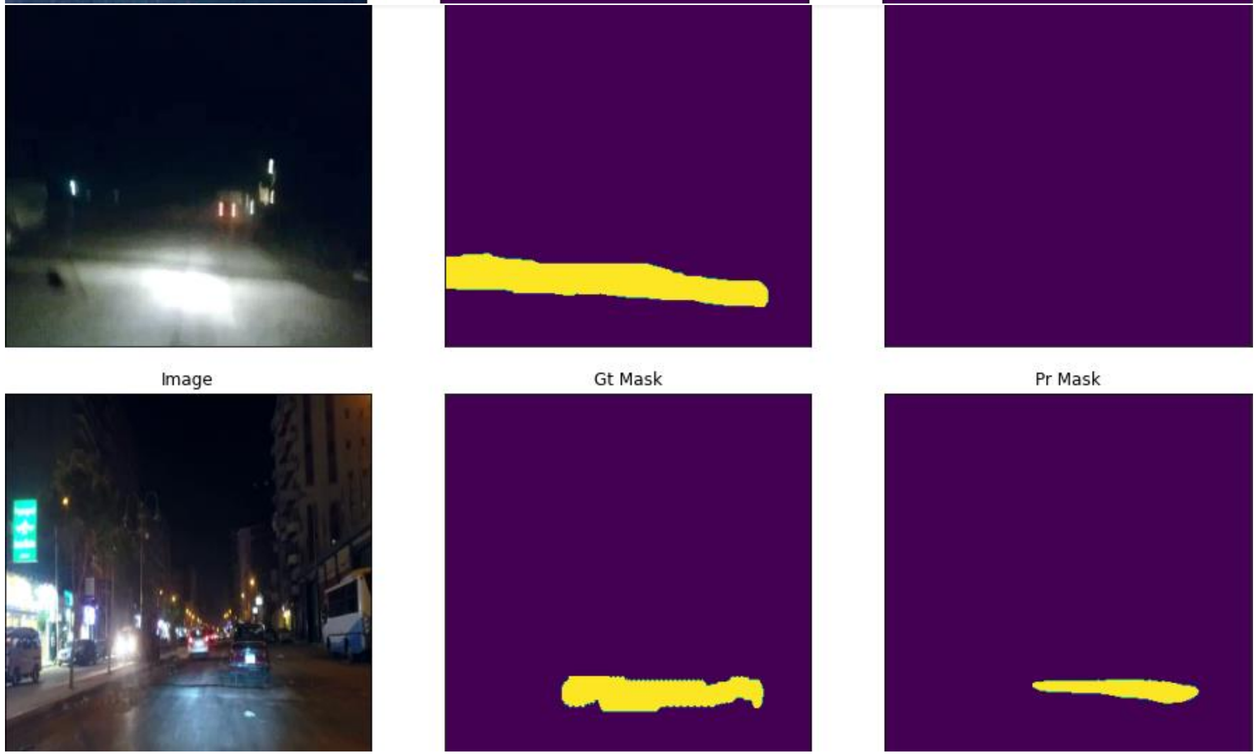


Image
Gt Mask
Pr Mask

CHAPTER THREE

3

Distance Estimation

3.1 Introduction

3.2 Calibration

3.3 stereo visions
- 3.3.1 Zed stereo camera
- 3.3.2 Two identical cameras

3.4 single camera
- 3.4.1 Deep learning model
- 3.4.2 Simple Algorithm with known feature from object
- 3.4.3 Simple Algorithm using Horizon of image

3.1 Introduction

The system detects bumps ahead and issues warnings in advance to drivers for avoiding or mitigating the harm caused by bump crashes. It identifies a target bump within trajectory of subject vehicle and determines range to the bump. And warning is issued when range is lower than a certain threshold.
There are many approaches to calculate distance with different accuracy and costs.
We will illustrate different approaches which we select between them to based on cost and accuracy.
Before illustrate approaches we will illustrate camera calibration as it is an important phase before calculating distance.

3.2 Calibration:

A camera projects 3D world points onto the 2D image plane
Calibration: Finding the quantities internal to the camera that affect this imaging process
•Image center
•Focal length
•Lens distortion parameters
Some pinhole cameras introduce significant distortion to images. Two major kinds of distortion are radial distortion which causes straight lines to appear curved. Radial distortion becomes larger the farther points are from the center of the image and tangential distortion and similarly occurs because the image-taking lenses is not aligned perfectly parallel to the imaging plane. So, some areas in the image may look nearer than expected.
After calibration and getting the camera ready we have to make sure to fix the positioning of it as a small variation will lead to defect the results

3.3 stereo visions:

3.3.1 Zed stereo camera:

One of the most recent developed stereo sensors is the ZED camera, it has two high-resolution cameras that capture images (left and right) at the same time and transmit them to an external computer device for processing. It is a passive device, that is, it needs additional equipment for the power supply. According to manufacturer’s information, it is designed and built to perceive the depth of objects in indoor and outdoor environments within a range of 1 to 20 meters and at 100 FPS. This device can be used on several platforms, including standard PC and also SBC (Single Board Computers) as the NVidia development kit. By attaching the ZED camera to the recently launched Jetson board of the NVidia constitutes a powerful tool for applications in the field of Robotic Vision.
The implementation of stereo vision in computers uses this basic principle to recreate a 3D scene representation based on the two images of it taken from different viewing points. This is known as stereo reconstruction.
In order to do stereo reconstruction, a series of steps are necessary, as calibration, rectification, and further depth determination using the ZED Calibration tool "ZED SDK".
But Zed stereo camera is expensive so we will see another methods to calculate distance.

3.3.2 Two identical cameras:

The advent of digital cameras has revolutionized the way users take pictures. Compared to their analog counterparts, such digital cameras (including mobile devices cameras) provide fast and easy image procurement, storage and retrieval. However, most of the popular digital cameras are capable of capturing a 2-dimensional (2D) projection of the scene while the components corresponding to depth are lost (not recorded). 3-dimensional (3D) imaging has emerged as advancement to the conventional 2D technology with the additional information of depth included. 3D cameras have started to appear in the market but are prohibitively expensive. For a normal user with a 2D digital camera, 3D images may be constructed by extracting the depth information from 2D images using a variety of techniques.
There are various approaches for generation of depth-maps. Some of them are supervised and others are unsupervised. A MATLAB algorithm can be used to construct depth mask using two static images. The algorithm displays the two images and the user matches corresponding points in both images. From the displacement of the selected image points the algorithm estimates a depth surface for the scene. It is a supervised approach; among these methods, depth-map generation from a stereo pair of images is the most popular one.

Essentially, a depth-map is a Grey-coded 2D image that gives the perception of depth by the intensity of colors. Darker regions in the Depth-Map are created for signifying that an object is far away and this darker color gradually decreases to brighter with decrease in depth and finally become white for closer objects
This can be done using an algorithm for the creation of depth-map starting from a stereo pair of images left (L–) and right (R–) images corresponding to the same scene by performing a pixel-to-pixel matching.

How it works:

The algorithm finds matching pixels by comparing the RGB components of the pixels in the L– and R–images. If the dissimilarity between the compared pixels is found to be less than a pre-specified tolerance (user defined) then those pixels are considered by the algorithm as a 'matching pair' of pixels. Binocular disparity is then calculated for the matched pixels which are further utilized to estimate depth information.

About this algorithm:

A depth map is a 2D image that gives the depth (with respect to the viewpoint) of an object as a function of the image coordinates. Usually, it is represented as a Grey level image with the intensity of each pixel registering its depth.

Tasks required for creation of Depth-map are:
•Capturing Images
•Image Preprocessing
•Depth Estimation
•Calculation of color value for all pixels.

A. Capturing Images

The algorithm uses two images of the same scene. The left and right images can be taken with two digital cameras. They capture images of the same scene, at the same time. These cameras are slightly displaced by some horizontal distance. This horizontal distance should be fixed.

B. Image Preprocessing

Read the headers of L– and R–images to find number of pixels along the height and width of the images. Read both images byte by byte and store them. Separate RGB (Red, Green & Blue) components of each pixel.

C. Depth Estimation

Depth estimation is the calculation of depth of different objects in a scene from a multiple views or images. It is required to find corresponding pixels in the different views, i.e., point of correspondence that identifies the same 3D points. By finding these points.

Depth information is calculated by following three main steps:
(1) Matching of Pixel.
(2) Choose the best in case of conflicts.
(3) Disparity Calculation.

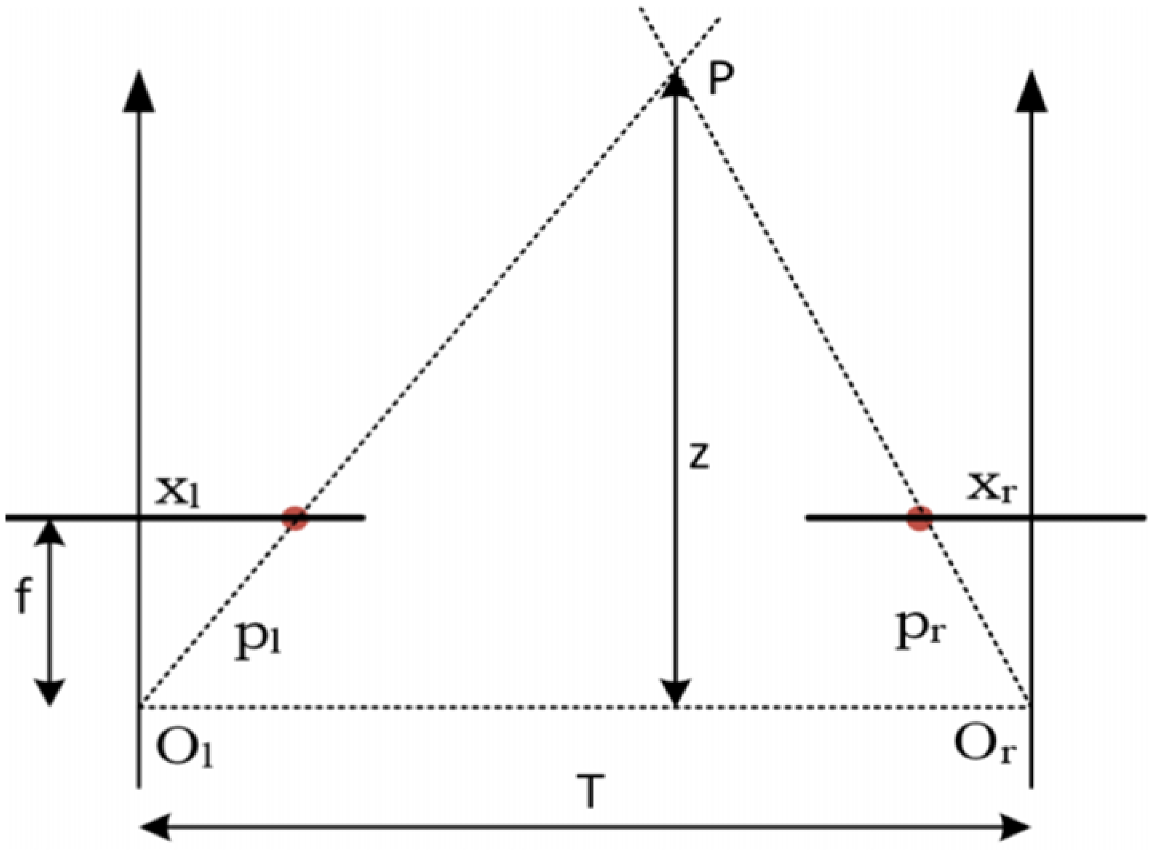


Figure **12** : Distance Estimation with 2 cameras

1) Matching of Pixel:

The matching criterion is based on Sum of Absolute Differences (SAD). SAD is a matching cost function, the metric of which is calculated for the three colors channels and the resulting three absolute difference values are simply added. If this value is less than or equal to ±2.5% of tolerance, then only consider the corresponding pixel of right image to be the matching pixel of the left image.

2) Resolving conflicts:

If a situation occurs at which it has been found that some other pixel in right image is similar to that pixel of left image, which is already matched with some other pixel, then the ambiguity is resolved by comparing the Tolerance of previous pixel with that of present pixel. If the tolerance value of present pixel is more than ignore it, otherwise discard the previous pixel and consider present pixel to be the right choice.

3) Disparity Calculation:

The method of Binocular disparity is used for creating the depth map. Binocular disparity is the difference between the two images or two eyes, as illustrated in Fig. 12. Here,

- pl : pixel value in the left image
- pr: pixel value in the right image corresponding to a similar pixel in left image
- f : focal length of the camera
- T: difference between the origins of the two cameras
- Z: depth value
- xl : distance of pixel in the left image
- xr: distance of pixel in the right image .

The disparity value of a point is often interpreted as the inverse distances to the observed objects. In other words, disparity is inversely proportional to Depth. Therefore, finding the disparity is essential for the construction of the depth map.

Disparity = $fT / z = xr - xl$

And $z = fT \, |xl - xr|$

D. Calculate Pixel value

For the calculation of color values first calculate the maximum depth from the depth of among all pixels. Give value equal to 0 i.e. black color for the pixel of having maximum depth.

The Color value for each pixel is calculated by:

Color = $255 - (\text{depth} * 255) / \text{maxdepth}$

Assign value equal to 255 i.e. white, to those pixels in left image which do not have any match in right image. By giving these pixel values, a depth-map can be created. In this depth map darker regions represent that object is far away and lighter regions represent that object is closer to the user. Fig. 2 depicts the details of the proposed algorithm.

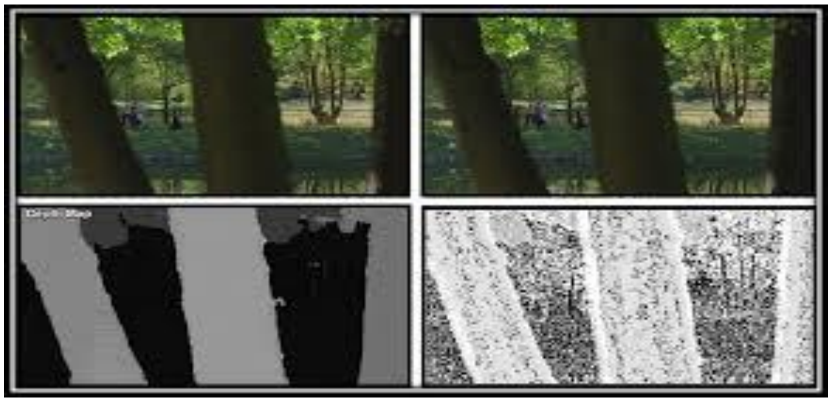


Figure **13**: depth map using two cameras

To apply this method properly:

•The two pictures (right and left) must be taken at the same time and with specified distance as mentioned before.
•This method produces a depth map for all the objects existing in the picture
•The accuracy is quite good but not enough for critical and real time applications

3.4 single camera:

3.4.1 Deep learning model:

Here we wanted to have a deep learning setup that provides the bump detection system with a method to estimate the distance from the monocular camera to the bump viewed with the camera is presented.
The presented distance estimation system is based on Multi Hidden-Layer Neural Network, which is used to learn and predict the distance between the bump and the camera sensor. The model was trained using a supervised learning technique where the inputs are the image and the boundary box of bump that is resulted from the object detection model and outputs are the distances to objects in the image**.**
The model was trained with KITTI dataset that consists of video sequences taken from a driving vehicle. Dataset consists of 56 scenes, and contains around 20,000 images overall.
When the model was tested with dataset similar to training dataset the accuracy is high but fail in case the objects are located on the sides of the camera or the curved road and in case objects are far away from the camera.
As we do not have images of bumps with known distance to train model and model accuracy is high so we will use model to give us range of distance (not accurate Distance).
But when we tested this model on our collected dataset and given the 2D-bounding box as input, the output distance was illogical.

3.4.2 Simple Algorithm with known feature from object:

The input of this algorithm is a 3d image with the wanted object marked and the output is the measured distance of it.
In order to determine the distance from our camera to a known object or marker, we are going to utilize triangle similarity.
The triangle similarity goes something like this: Let’s say we have a marker or object with a known width W. We then place this marker some distance D from our camera. We take a picture of our object using our camera and then measure the apparent width in pixels P. This allows us to derive the perceived focal length F of our camera:
$F = (P \times D) / W$
P: apparent width in pixels
F: focal length
D: distance from our camera.

To apply this algorithm:

- ➢ the width (or height) of the object which we calculate distance toward must be fixed so in our case we choose to fix the width of the bump which is not accurate so we will get ranges of distance where first range with distance less than 3 meter, range 2 with distance less than 6 meter, and range 3 with distance greater than 6 meter.

• After getting the focal length we can use this formula to get the distance

D= (W x F) / P

Results:

- Are quite accurate we tried it this algorithm to measure objects with range 10 meter away from the camera and the error is about 5 % Advantages of this technique.
- Simple and Available hardware
- Easy to apply.

Disadvantages:

- not accurate as bump width is not fixed.
- Output will be ranges

3.4.3 Simple Algorithm using Horizon of image:

In this algorithm the horizon, which is an important feature from image, is used to calculate distance.
First let's define Horizon line:
Vanishing points are imaginary points at infinite distance away from the station point. In practice, the point at which the visual ray from the eye to that infinitely
Distant point pierces the picture plane is referred to as the vanishing point.
Vanishing point is considered for group of parallel lines in real but in image it will intersect in a point which is Vanishing point.

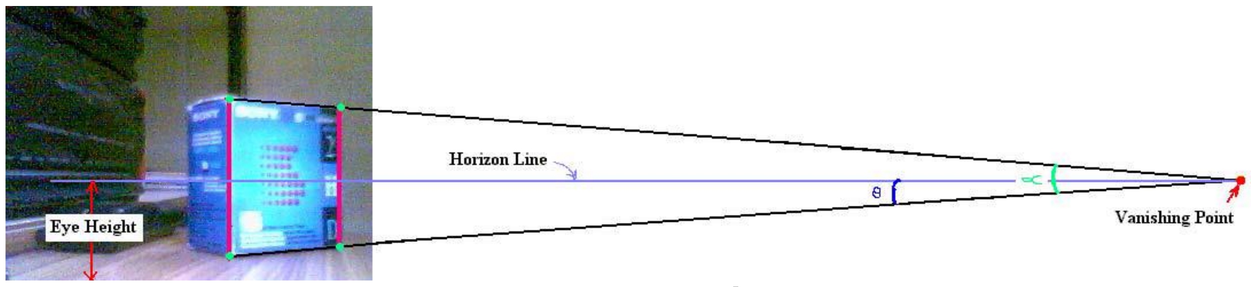


Figure 14: Horizon line

Connecting all these vanishing points will create horizon.
The horizon can be used to calculate distance which will be illustrated next.
So let’s start the algorithm:
The inputs of algorithm are the original image having the horizon and the output of segmentation model which determining the bottom line of bump. With camera kept at constant height, we can get the distance between camera and the target bump.

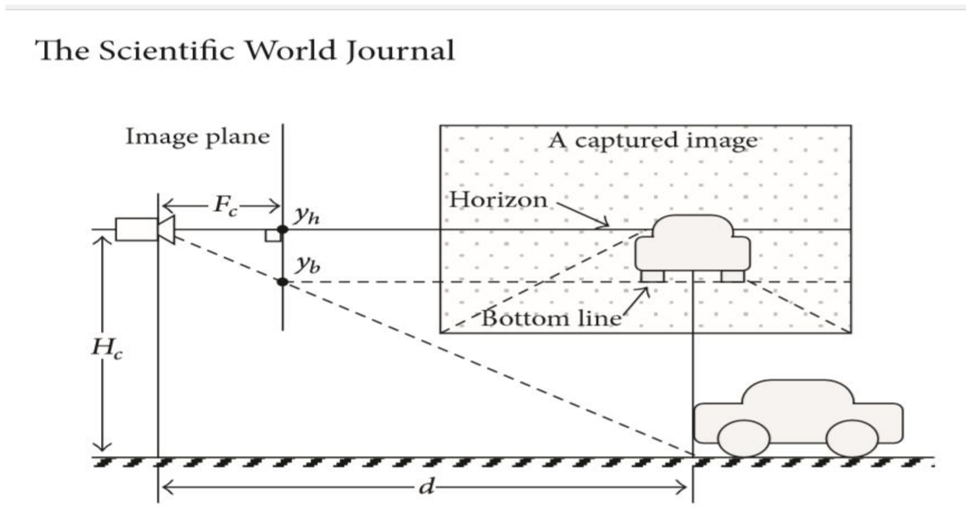


Figure 15: Distance estimation using Horizon

Let contacting line between bump bottom and road surface in image be bottom line and let horizontal line passing through vanishing point of road lanes be horizon. Horizon will pass through the center of image when optical axis of camera is parallel to road surface. It moves upward or downward depending on camera angle. Distance between bottom line of a bump and horizon is inversely proportional to range to the bump. When camera angle is negligibly small, range *d* to vehicle can be calculated as in the following:

$$d= fc*Hc /(yh-yb)$$

Where:
fc: focal length
Hc: camera height (must be constant)
yh: vertical coordinate of horizontal line
yb: vertical coordinate of bottom line

Experimental Results:

When we tried this system to calculate distance between random objects and the camera within small ranges of around 10 meters, it results in good accuracy of distance estimation [error around 0.3m to 0.5m.

Conclusion:

In this algorithm we propose a range estimation method which can be used in urban traffic environments. The proposed method estimates the bottom line from information of the bounding box of bumps in image which is obtained by object detection algorithm and calculates range from bump position in image with the virtual horizon. Small variation in position of horizon may result in large range error. Horizons determined by analyzing lane markings can be located even below bumps. Experimental results confirm that the proposed method provides robust results in urban traffic environment.

CHAPTER FOUR

4

MAPPING BUMPS LOCATION IN OFFLINE GOOGLE MAPS

4.1 Introduction

4.2 Literature

4.3 System Architecture

4.4 System Architecture in details

4.5 Challenges

4.6 Methodology

4.6.1 VCEH

4.6.2 Bumping Events Detection Algorithm

4.6.3 Bumping index

4.6.4 Data mining

4.7 Application functions

4.1 Introduction:

Various mobile sensing technologies have been proposed to detect road anomalies (such as potholes, uneven manhole covers, and speed bumps) or to assess road surface conditions. One of the most popular approaches is to detect the vibrations of vehicles as they run over anomalies. In most studies, accelerometers, usually called G-sensors, were used to measure the vibrations, and GPS receivers, simply called GPS for short, were used to obtain the speed and location of the vehicles

4.2 Literature:

There are methods include RADAR sensors or LIDAR sensors which are used for the adaptive cruise control system These sensors increase the cost of the car, and drivers who already own cars are not willing to pay for additional devices on board. Also, ultrasonic sensors can be used for detecting speed bumps but these sensors are not very reliable as their range is very limited and they may not provide sufficient time for the driver to brake.
Video sensors require image processing techniques to analyze the color and determine the distance and height of the bump. However, video sensors are limited because video is not effective at night.
Another method is the accelerometer sensors which continuously provide the x,
y, and z coordinates of any point on the road.
The accelerometer sensors can overcome the limitations of previous methods as they exist in most smartphones owned by drivers which reduces the cost. Also, these sensors can detect bumps more accurately than the aforementioned methods. This method collects big data that need efficient storage and processing before giving any good result. Existing accelerometer based methods are also limited to warning or notifying drivers about the bumps and are not accompanied with mechanical speed control solutions.

4.3 System Architecture:

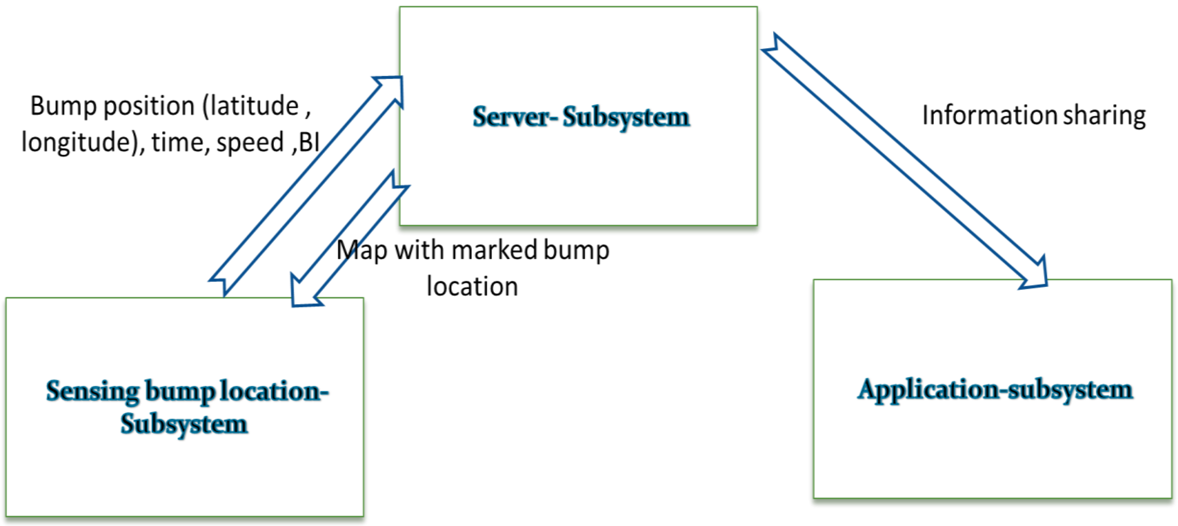

Bump position (latitude , longitude), time, speed ,BI
Server- Subsystem
Information sharing
Map with marked bump location
Sensing bump location-Subsystem
Application-subsystem

Figure 16: Mapping system block diagram

System mainly is consisting of three subsystems:

1. First subsystem is sensing, detecting bump and determining location by application mobile phone that sends latitude, longitude of bump location, time, car speed, and other features of bump that represent the severity of road bumps.

2. Second subsystem is server that provides service application programming interfaces (APIs) for collecting data from crowd and for sharing information with applications and data mining engines for extracting road anomaly information from crowd sourced data, and data warehouse capabilities.

3. Third subsystem is the application where the web-based information management system is used for road pavement managers to schedule and dispatch maintenance jobs and evaluates the performance of workers; the worker application is used to facilitate the road pavement repairs such as on-site confirmations and progress reports.

In this project we will focus on first 2 subsystems.

4.4 System Architecture in details:

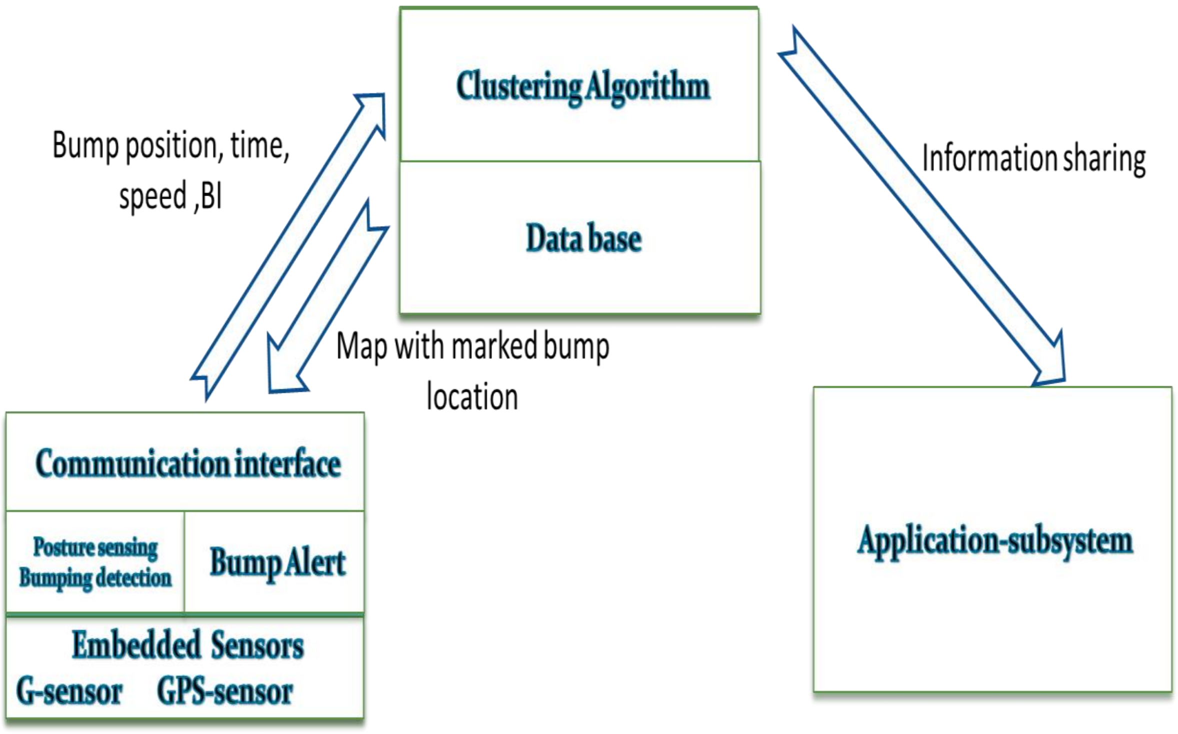


Figure 17: Mapping system components

•First subsystem is consist of 2 layers where first layer for bumping detection using Accelerometer sensor to detect bump by finding any change height of road surface, and GPS sensor to determine location every second. Sensing module should be adaptive to vehicle conditions, smart phone specifications and installation posture. And second layer is Communication interface is used Wi-Fi to send data to server or to receive location of nearby bumps.

•Second subsystem is Cloud server is used to collect data bumps in tables, apply machine algorithm clustering to determine best location of bumps where different cars are used in different speed, and to send maps with assigned bumps to user, server should continuous be updated.

4.6 Challenges:

In design application, there are some challenges that we should overcome to build intelligent system.
As we couple vehicle with mounted mobile phones
•Application should be simple and without bringing any issues to drivers, not installing on a fixed position or orientation ,to be more comfortable to use which will be solve by vertical component extraction heuristic.
•System must not affect by environment: for example Speed car, conditions of vehicles, type of vehicles, mobile frequency, and sensor sampling rate which will be solve by Data mining.
•Even different times with different speed of the same car or different users move on the same bump, we should have one location to bump which is best allocated one.
•Data usage: extract key features that represent the severity of road bumps reduce an excessive unnecessary data are reported which will be solve by Bumping index.
•Power consumption: application should be not use large size of memory on runtime.

4.7 Methodology:

- VCEH: is introduced to extract the VCs of g-vectors upon which the bumping detection algorithm relies.so driver do not need to install mobile in a fixed position.

- Bumping event: is a simple algorithm to detect bump by using statistics approach.

- BI: an assessment metric called bumping index will be introduced. In short, this index is the ratio of a short-term standard deviation to a long-term standard deviation of the VCs of g-vectors which express height.

- Data mining: The position obtained from the GPS and stored in the database is not totally accurate so data mining algorithm is used to reduce the error in the position. The algorithm is also necessary to ensure that the anomaly represented by the BI is a real bump**.**

4.7.1 VCEH:

An Accelerometer sensor is used to determine any disturbance of height surface of road where we use statistics approach: the proposed algorithms rely on statistics on sensor reading in periods of time Where not to install mobile phone in certain orientation a vertical extract of g-vectors.

Definitions:

The main task here is to estimate the gravity vector in the sensor frame, after that the (VC) vertical component can be given by scalar projection of g-vectors to estimated gravity vector, and then long-term standard deviation of VC will be calculated as a parameter to model detect event bumping.
The gravity is the major component of the g-vectors at most of times, in case of smart phone is mounted on a rack by applying (PCA) principle method analysis to determine direction of gravity, where to save power instead of applying PCA, we will use running average to determine gravity vector, to improve accuracy, where estimated gravity vector can be calculated as next equation:

$$\bar{g}_t = avg(\bar{g}_s)\ \{from\ \ t - T_1 < s < t\ \&\ (\bar{g}_s - \bar{g}_t) < \delta\ \}$$

Where δ is a stable threshold to filter out g-vectors that differ extremely from the previous running average **δ= ($\bar{g}_s - \bar{g}_t$).**

Where upon [**14**] field tests δ and CDF measurements and since the purpose of δ is to find out significant vibrations and to reduce unnecessary amount of computation, we also choose the value used in that paper (δ=2m/s2) to reduce 95% computations at the same time.

Define g⊥ to denote the VC of the g-vector at time t.

$$g\perp = \langle g_t, \bar{g}_t \rangle / \|\bar{g}_t\| .$$

Where $\langle g_t, \bar{g}_t \rangle$ is the inner product and $\|\bar{g}_t\|$ is the normalization.

Estimate gravity vector by online approach instead of batch approach
Batch approach: The gravity is obtained by taking average on g-vectors over all stable periods.
And online approach: The gravity is estimated by taking average over the stable periods in the last minute with 1-s period.
A smartphone collect about 30 min of data with online approach (window size 60 sec) where the RMSEs are 0.03 m/s2 and of the batch mode is 0.01 m/s2.

Window size selecting:
Window size was chosen 180 sec.
Based on experiment with two smartphone used with various running average window sizes from 10 to 90 s were tested, where the results with a window size 60 will achieve the RMSE around 0.02 m/s2.

Vertical Component Extraction Heuristic:
Upon [**14**] we used the same algorithm to extract vertical component.
Where:
Smart phone update parameters every 1 second
T_0 is time for system to initiate, which is set to 1 minute
T_1 is running average window size, which is set to 3 minute
g_i is g vector in i second
$\bar{g}_i$ is estimated gravity vector at end of i second
$g_i\perp$ is VC in i second
S set of indices of stable periods within running average window size.
σ_i is standard deviation of VCs in stable periods at the end of ith second.
The input of the algorithm is g_i for i=1, 2 …and the outputs of the algorithm is $\bar{g}_i$, g⊥ and σ_i for i=1, 2 …

ALGORITHM FOR VCEH:

```
1- $T_0 = 60, T_1 = 180$
2- Initialize parameters
3- // repeat part while application is running
4- Periodic get $g_i$ every 1 second
5- i++
6- Get $g_i ⍰ = \langle g_i, \bar{g}_{i-1} \rangle / \| \bar{g}_{i-1} \|$
7- If $| g_i ⍰ - \| \bar{g}_{i-1} \| | < \delta$ S=S U {i}
8- If i>$T_1$ S=S-{i-$T_1$ } end if
9- if |S|<=$T_0 /3$ go to step 2 Initialize parameters end if
10- Get $\bar{g}_i$ which is averaging window size $g_i$ for last 180 second.
11- $\sigma_i = standard\ deviation\ g_i \perp$
12- // until program is terminated
```

Where Initialize parameters, the program waits T_0 second to read g_i from i=0 to T_0 ,initialize variables
S= {1, 2,… $, T_0$ },$\bar{g}_i$ average of g_i from i=0 toT_0 , $g_i \perp$ from i=1 to T_0 which is projection of g_i to$\bar{g}_i$ the estimated gravity vector, σ_i from i=0 to T_0 which is standard deviation $g_i \perp$.
After initialize parameters, lines 3 to 12 will be executed every second until the end of program.

4.7.2 Bumping Events Detection Algorithm:

The bumping events can be detected by recognizing and analyzing abnormal vibrations, where we scan for a high pulse in the waveform and then analyze the following waveform to confirm or deny the observation, where a fixed threshold will be needed to detect high pulse of the waveform not to change with car dynamics.

In VCEH σ_(i)can be a parameter representing the oscillation system, so we will use it as a threshold to detect the high pulse that may imply bumping event.

1. Check if g_i⊥- g ̅_(i-1) > c1*σ_(i-1)
   Where c1 is a constant .

2. If condition happens , a potential bumping event is detected , then g_i⊥ is scanned from that time to ∆T millisecond backward to find maximum amplitude,
   Let σ_(event)= standard deviation (g_i⊥) from time before bumping event by ∆T to bumping event.
Where ∆T=500 millisecond.

3. If σ_(event)>c2*σ_(i-1)
Where c2 is a constant

Choosing c1 , c2 based on [] designed that method they did experiments on two vehicles (Ford Windstar 2003 and Mitsubishi Lancer 1997) and four different racks were used ,they used c1=2,c2=2.5, so we will use these values and if
we need to retune it we will change among experiments.

4.7.3 Bumping index:

Extract height of bumps that represent the severity of road bumps with mitigation car vibrations, where the vibrations are represented as a long term standard deviation. When the vehicle passes over a bump, the VCs change dramatically and the standard deviation changes for a short time. We call the ratio between the long term standard deviation and short term standard deviation the Bump Index (BI).

$$BI= \sigma\_event \ /\sigma$$

The BI value expresses the height of the bump.
The local database stores the BI value, the GPS coordinates at which the BI occurs and the timestamp. When the trip finishes and the smartphone connects to the internet, it uploads to the cloud the log file which contains the BIs during the entire trip.

4.7.4 Data mining:

The position obtained from the GPS and stored in the database is not totally accurate so a data mining algorithm is used to reduce the error in the position. The algorithm is also necessary to ensure that the anomaly represented by the BI is a real bump.
For this purpose, we use the K-means algorithm that is used for clustering. In our context, every car j submits a set of bumps positions Pj = (P1, P2, …., Pn) where each Pi is the GPS coordinates (xi , yi). If m cars pass through a road, the road will have m sets of bumps. The k-means clustering algorithm runs firstly on these points to find the mean value of the distance between points. The mean value should achieve the minimum mean square error .The outcome of the algorithm is b1, b2, …, bc where c is the number of bumps in one road. These mined data are stored in bump data table in the database**.**

4.8 Application functions:

an android application was written to map location of bumps by using accelerometer sensor to detect bump and GPS sensor to find location of bump and send location data to server which run clustering algorithm to define best location of bump to save into tables of bump to resend again to users when need .
 Other functions were added to help drivers more like showing current position every time during driving and surrounding public facilities
When application is started, the driver must to press to allow collecting data from accelerometer and GPS sensor.

A message will be send to enable read GPS data in first time.
After you press the activity will start and the offline Google map is appeared.

Accelerometer starts to collect data every 200 millisecond which will pass to VCEH algorithm every second by average (5 samples of 200 millisecond).

GOOGLE maps:
It marks your location by a blue marker and show important places restaurant, hospitals and all public facilities.

GPS sensor:
finds your current location to mark it in GOOGLE maps. If bump was detected the latitude and longitude of bump will be saved and sent to server to run clustering algorithm.

Server will send bumps location, when page is opened in arrays to mark on map.

A message will be sent says that “map is ready” and also marks your current position with blue marker as figure1
If bump is detected a new marker will be added and sent to database, also a message will be sent says that “a new bump was detected at latitude: …, longitude: …”

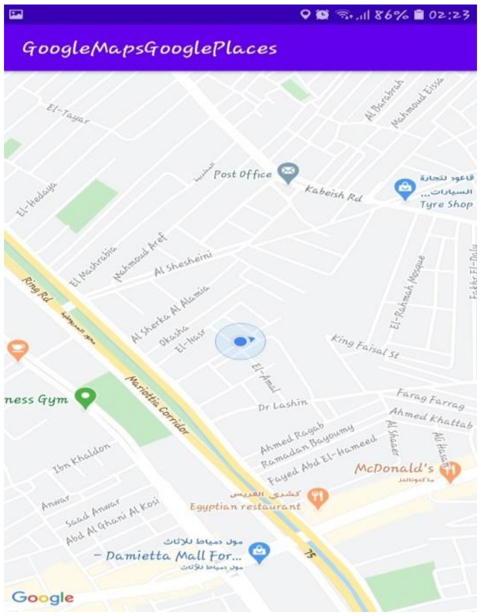
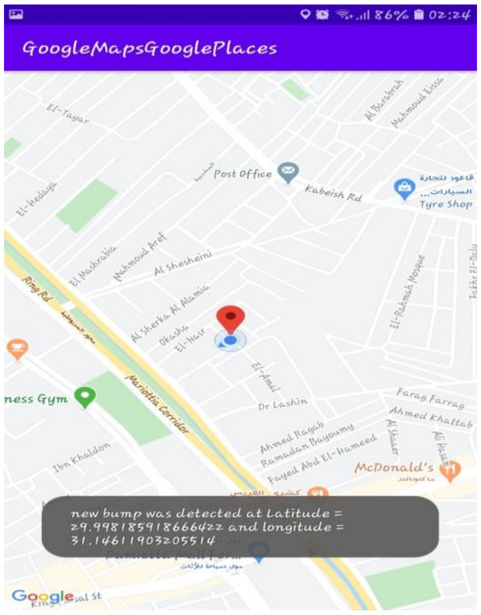


Figure 18: starting map



new bump was detected at Latitude = 29.998185918666422 and longitude = 31.14611903205514

Figure 19: Detecting bump

CHAPTER FIVE

5

Merging between 2 approaches & Future work

5.1 Merging between segmentation& Mapping

5.2 challenges

5.3 conclusion & future work

5.3.1 Segmentation model

5.3.2 Distance Estimation

5.3.3 Mapping

5.1 Merging segmentation, distance estimation and mapping blocks:

•Merging the two approach

Now we have 2-modules detect the bump
Segmentation module
Mapping module

We propose a merge method to combine the outputs from them

• Proposed approach

As illustrated in section related works, detecting bumps is determined by different method as machine learning, or sensor readings statistics (deviation, mean) with a fixed threshold, in this project we built a full system to detect bumps by two methods using deep learning from camera images for upcoming bumps and applying two different algorithms (as illustrated in section calculate distance) to estimate distance and another method to detect bumps after passing using sensor readings.
So we should merge the two different methods to generate a full picture of environment.

In this part we illustrate a method to merge 2 modules and its benefit.
To connect different parts IOT (internet of things) will be used,
So we should gather information from several sensors where camera will be connected to IOT device specifically a Raspberry pi which execute segmentation model with 2 algorithms to estimate distance to bump ,the Raspberry pi must connect with server to send flag if bump is detected and send distance estimated ,speed of car , and if then the sensing application mobile phone detect bump and send location of bump from GPS sensor to server so if server received information from 2 modules then location of bump will be saved to database to resend to other users with

5.2 challenges:

While building the system, there are many challenge was faced, such as:
The system is consist of 3 large parts with many details (deep learning ,control, design mobile Application).
Collecting data is taken a large time to have variety and large number of data as speed bumps data online is rare.
Annotating data is taken a large time.
Model parameters are retuned to get a result.
Not all the required component for the project is available in the Egypt market as zed stereo camera.
Collecting data from accelerometer and GPS sensor of mobile phone when car moving above bump is not available online to run clustering algorithm.

5.3 conclusion& future works:

Future works are Adding a new block to down speed of car automatic.
And improve 3 blocks output as:

5.3.1 Segmentation model:

As Model failed with shadow of car mirror so we should collect new data with
Shadow of car mirror.

Retune hyper parameters that may increase accuracy.

Use VGG19 to increase FPS**.**

Change decoder part to increase FPS.

5.3.2 Distance Estimation:

Using stereo- camera to increase range of distance.

Or collecting data of bumps with known distance and train model object detection 3D.

5.3.3 Mapping :

Using deep learning with features from sensor reading to detect road specification (bump- smooth road).

References

[1] Aslam, A. and Ansari, M., 2019. Depth-map generation using pixel matching in stereoscopic pair of images. arXiv preprint arXiv:1902.03471.

[2] Celaya-Padilla, J.M., Galván-Tejada, C.E., López-Monteagudo, F.E., Alonso-González, O., Moreno-Báez, A., Martínez-Torteya, A., Galván-Tejada, J.I., Arceo-Olague, J.G., Luna-García, H. and Gamboa-Rosales, H., 2018. Speed bump detection using accelerometric features: a genetic algorithm approach. Sensors, 18(2), p.443.

[3] Fleury, D. and Fleury, A., 2018. Implementation of Regional-CNN and SSD machine learning object detection architectures for the real time analysis of blood borne pathogens in dark field microscopy.

[4] https://keras.io/guides/transfer_learning/.

[5] Kariler, M., 2017. Road Surface Quality Detection Using Accelerometer Data (Doctoral dissertation, Master Thesis).

[6] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. Neural networks, 1(2):119–130, 1988.

[7] M Tan, QV Le - arXiv preprint arXiv:1905.11946, 2019 - arxiv.org.

[8] Murali, S. and Avinash, N., 2004, December. Estimation of Depth Information from a Single View in an Image. In ICVGIP (pp. 202-209).

[9] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detec ´ tion: An evaluation of the state of the art," *PAMI*, vol. 34, 2012.

[10] Park, K.Y. and Hwang, S.Y., 2014. Robust range estimation with a monocular camera for vision-based forward collision warning system. The Scientific World Journal, 2014.

[11] Ronneberger, P Fischer, T Brox - International Conference on Medical …, 2015 – Springer

[12] Rosebrock, A., 2015. Find distance from camera to object/marker using Python and OpenCV. *PylmageSearch*.

[13] Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., Jagersand, M. and Zhang, H., 2018. A comparative study of real-time semantic segmentation for autonomous driving. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 587-597).

[14] Yi, C.W., Chuang, Y.T. and Nian, C.S., 2015. Toward crowdsourcing-based road pavement monitoring by mobile sensing technologies. IEEE Transactions on Intelligent Transportation Systems, 16(4), pp.1905-1917.

[15] O Ronneberger, P Fischer, T Brox - International Conference on Medical …, 2015 - Springer

[16] M Tan, QV Le - arXiv preprint arXiv:1905.11946, 2019 - arxiv.org

[17] https://keras.io/guides/transfer_learning/

[18] DP Kingma, J Ba - arXiv preprint arXiv:1412.6980, 2014 - arxiv.org