Using Stereo Vision and Transfer Learning to Identify Potholes on the Road

Amita Dhiman

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

Supervisors
Prof. Reinhard Klette
Dr. Waqar Khan

School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology
New Zealand
November 2019

To the two loves in my life, my darling husband Dharamveer, cheerful daughter Aahana and of course to the reader who is reading this.

Declaration

I hereby declare that I am the sole author of this thesis. Any part of this thesis is not submitted for the requirement for any other degree. To the best of belief and knowledge, this work does not contain any written or published material by another person. I understand that this thesis can be made electronically available for public.

Amita Dhiman

Acknowledgments

I am delighted to acknowledge the support of my family through out my journey of research. This research would not have been completed without the blessings of my dear parents (Charanjeet and Ramesh Dhiman), continuous encouragement from my husband and of course my daughter who is a little bundle of curiosity.

I would like to express my deepest gratitude for all the suggestions, immense knowledge and guidance that were so graciously offered by my "Guru" Professor Reinhard Klette.

I thank Dr. Waqar Khan for his friendly suggestions and support for guiding me through my research adventure. I also thank Dr. Hsiang-Jen Chien for his unhalting supply of suggestions, friendly support and fun yet informative consultations over coffee.

To the rest of the team of Prof. Reinhard's research group, big thanks for accepting me in your research group and for having a wonderful time during Waiheke, trip in September 2017. Thanks to all the co-authors, with whom I have published my papers during my Ph.D.

Last but not the least, I bow and bestow my love at the feet of "Lord Shiva" and "Maa Durga" for providing me all of the above and turning me from nothing to the one who is writing this today.

There is only this life, this movement. This is the only thing we can be sure of. Everything else is theory. - Shiva trilogy.

Amita Dhiman
Auckland
November 18, 2019

Publications

A. Dhiman, H.-J. Chien, and R. Klette, "Road surface distress detection in disparity space", IVCNZ New Zealand, 2017.

A. Dhiman, H.-J. Chien, and R. Klette, "A multi-frame stereo vision-based road profiling technique for distress analysis", ISPAN, China, 2018.

I. Ernst, H. Zhang, S. Zuev, M. Knoche, A. Dhiman, H.-J. Chien, and R. Klette, "Large-scale 3D Roadside Modelling with Road Geometry Analysis: Digital Roads New Zealand", Pervasive Systems, Algorithms and Networks, 2018.

A. Dhiman. S. Sharma, and R. Klette, "Identification of road potholes", MIND, India, 2019.

A. Dhiman, W. Khan, and R. Klette, "Pothole detection using deep learning", t-tech19, ITS, New Zealand, 2019.

A. Dhiman and Reinhard Klette, "Pothole detection using computer vision and learning", To appear in IEEE Transactions on Intelligent Transport Systems; accepted in July, 2019.

Abstract

Identification of dangerous road distress on road surfaces is essential to many applications such as improved driving comfort, safety, the country's economy and better traffic efficiency. For these reasons, research around the world has comprehensively explored strategies for identification of road distress. In this study, two different state-of-the-art approaches have been implemented and compared for pothole detection. The first approach, focuses on identification of road distress using stereo vision. Single and multiple 3D frame reconstruction techniques are performed for 3D plane fitting to model the road surface using a digital elevation model. Then a road manifold is constructed and further investigated for the detection of major road distress such as potholes.

As potholes might be dry, or snow or water filled, the second approach focuses on identifying potholes under different weather conditions. Transfer learning based techniques using Mask R-CNN and YOLOv2 convolutional neural networks focus on improved pothole identification. The pre-trained convolution layers of Mask R-CNN and YOLOv2 are trained to identify common natural features in an image such as edges or corners. This knowledge is adapted and transferred to the task of pothole identification using transfer learning. Potholes are identified with promising accuracy using a transfer learning based model. This study also serves the purpose of providing pointers to different datasets recorded on different days and light conditions.

Keywords: stereo vision, digital elevation model, transfer learning, convolutional neural networks.

List of Tables

List of Figures

Contents

Chapter 1

Introduction

In the era of smart cities, internet-of-things, the cloud, and autonomous vehicles, the road, the basic transportation medium, and road maintenance might seem to be behind in discussion. Fixing road distress in time can save millions of dollars and increases transportation efficiency. Road distress is a depression on a road surface, mainly in the form of a pothole. A pothole is usually a bowl-shaped depression in the road surface and is, generally, an end result of a major crack. With the invention of new technologies, the detection and repair of road distress can be done more efficiently and quickly. These developments, persuaded this study to achieve the goal of road distress identification.

1.1 Motivation

Potholes are a perennial problem which starts with imperceptible microscopic cracks in road surfaces (see Fig. 1.1 for an example). Potholes may arise because of traffic overloading or weather affecting the road surface [1]. Due to rain and other weather factors, these cracks clog water inside which freezes and expands during cold weather. This void hole shape structure becomes a pothole, either water or snow filled, or dry.

Potholes cost lives: Currently, of approximately $33,000$ traffic fatalities each year in the USA, one-third involve poor road conditions; in the UK, about 50 cyclists are seriously injured every year because of Britain's poor roads [2]. In October 2017, Auckland Transport, New Zealand, received 276 requests for compensation for damages or injuries related to roads [3]. In India alone, potholes have killed over $11,000$ people in the last four years [4]. In Rome, potholes have caused an untold number of accidents and shredded the tyres of 15 vehicles [5] in 2018.

Figure 1.1: A pothole on a road.

Potholes cost money: Potholes put a big dent in driver's budgets and a country's economy. In New Zealand Christchurch spent $525,000$, Wellington $12,782$, Invercargill $60,000$, and Dunedin around $27,000$ in order to fix potholes [6]. In 2018, one of the largest pizza chains (Domino) commented that "*potholes cause irreversible damage to pizzas during delivery to homes*" and dispensed a $\$5,000$ grant to fix 53 potholes around 20 locations in the U.S [7]. The UK government announced a budget of 420 million euros [8] and Rome 17 million euros to fix potholes in 2018 [5].

Hence, potholes cost both lives and money. As discussed above potholes present grave danger to all transport vehicles, including cyclists.

Automotive industry inventions: Automotive industries use a variety of systems for road surface scans. For instance, the 2018 Jaguar Land Rover also invested in pothole detection technology. The system measures vibrations caused in the vehicles and adjusts its suspension for more driving comfort. However, for this technique of measuring vibrations, the vehicle still drives over the pothole [10]. The 2017 Ford Fusion used 12 high-resolution cameras to adjust computerised controlled dampers in a car for best ride comfort [11]. The 2013 Mercedes Benz S class uses a stereo-vision system to facilitate a road surface scan. Forward-looking cameras of the stereo-vision system are mounted near the central rear-view mirror and facilitates more driving comfort when there is a bump or speed breaker on the road [12]. Road surface inspection is also done commercially using specialized ve-

hicles [13, 14].

Unfortunately, all these technologies are equipped with expensive sensors and are without technology disclosure, therefore limiting its accessibility to the general public. This thesis aims to fill some gaps in the extensive field of road maintenance and road distress detection.

In Section 1.1, motivation for the research is introduced. Section 1.2 details our rationale behind conducting this research. In Section 1.3 we[1] define the contributions of conducting this research. Section 1.4 outlines the organisation of the thesis.

1.2 Research justification

Potholes are broadly classified into two types - shallow potholes or deep potholes. A shallow pothole is usually just in the top layer of the road, whereas a deep pothole involves failures beneath the road surface and can become very large in a short span of time. However, the classification of potholes is very subjective and mainly depends on funding and priority assigned to the road civil authorities [15]. A comprehensive technique is needed to identify potholes under different scenarios. A pothole, usually of irregular shape, can also be dry, water filled or covered with some material on the road (see Fig. 1.2).

The identification of potholes becomes very challenging under different weather conditions, such as different intensities of sunlight or in rainy weather. Road distress analysis can be done manually, i.e. using humans as sensors [16–18]. A study conducted in Taoyuan, Taiwan, uses a data-analytic approach applying correlation and regression analysis [16]. The Authors quantify how roads with higher proportions of road potholes result in a larger number of accidents. Several mobile crowdsourcing based applications have been developed to report data about road hazards (e.g. [17]).

With other vehicular sensing solutions such as the accelerometer, a pothole can be identified while the vehicle drives over it by measuring vibrations [19, 20]. An accelerometer is used because of its low cost and relatively simple detection procedure. However, there are more chances of false positive occurrences when a vehicle drives over speed bumps or suddenly swerves.

[1] The use of "we" throughout this thesis is wilful and is used to involve the reader with the thesis as recommended by Knuth et al. [9].

Figure 1.2: Dried leafy pothole on the left, and water filled pothole on the right, showing different intensities of light

For hazard avoidance, it is often desired to identify distress at a distance. This is achieved using *2-dimensional* (2D) vision-based methods [21, 22], potholes are identified from imagery data. Process design in this case is highly dependent on the application for which a 2D dataset is being processed. Road surface distress is generally of irregular shape.

A more accurate technique is to detect distress in *3-dimensional* (3D) space [23, 24]. Advanced automated road-inspection technologies often use specialized vehicle-mounted 2D laser scanners [25], 3D laser scanners [26], or ground penetrating radar [27] for producing accurate data. However, these special-purpose devices are expensive, limited to use in specialized automated vehicles, and not designed for integration into any vehicle.

We use and compare two techniques based on stereo-vision and transfer learning. Stereo-vision analyses road environments ahead of the vehicle and provides 3D measurements to extract geometric features of road distress. Stereo-vision cameras are cost effective, and their performance capabilities equate to 3D laser scanners, thus making them a wise choice for real-world applications. However, most of these techniques use high-resolution datasets for processing which are captured through specific cameras. The second technique, using neural networks, accepts images or videos as input and the developed model is useful to identify potholes using any types of images or videos captured through any camera.

1.3 Significance and Contributions of this research

For automatic identification of road surface distress, this thesis proposes and compares two different techniques- based on stereo-vision and transfer learning. At the heart of which is the assumption that automatic pothole detection can save both money and lives. Below are the key points of my research significance and contributions:

1. This research is of significance for better traffic efficiency, safety, and improved comfort while driving.
2. These techniques allow identification of potholes from a distance and in an accurate manner, as supported by experiments.
3. The techniques proposed in this thesis are applicable to monocular vision, inertial-monocular vision, or monocular LIDAR systems.
4. This thesis also fills the gap of benchmarking pothole detectors by providing required datasets to conduct pothole identification experiments.
5. This thesis also contributes to a data science field for pothole detection by collecting pothole data, labelling and training models.

By doing a set of experiments to evaluate the performance of techniques, we contribute to the present best practices for automatic identification of potholes.

1.4 Thesis structure

Chapter 1 serves as an introduction to the thesis as a whole. It explains our motivation, research justifications and our contributions in related fields. The overall format is shown in Fig. 1.3.

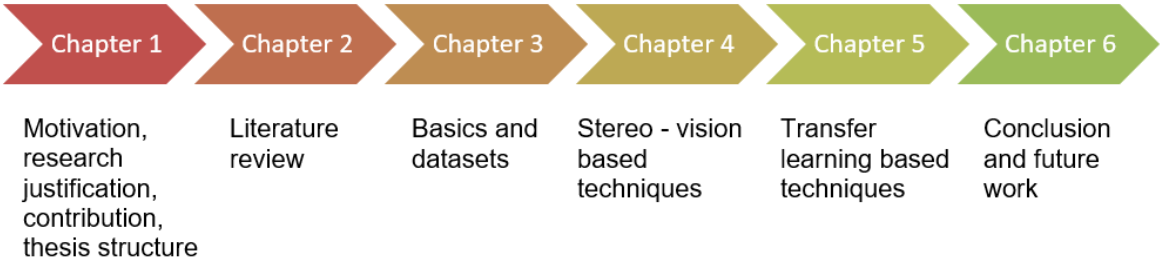

Figure 1.3: Thesis structure

Chapter 2 begins with a review of related background studies in the field of road distress techniques and developments.

Chapter 3 serves to provide brief explanations of basic theories adopted in this thesis. It also introduces the datasets used for techniques in Chapters 4 and 5.

Chapter 4 details two proposed techniques based on stereo-vision for identification of potholes. Chapter 5 presents two techniques of transfer learning using Mask R-CNN and YOLO networks. Chapter 5 also presents our results obtained by making a comparison between stereo vision and transfer learning using Mask R-CNN.

Finally, Chapter 6 summarises the findings of this thesis. It presents the conclusions drawn from the results of Chapters 4 and 5. This chapter also addresses further areas of interest.

Chapter 2

Literature review

Research around the world has comprehensively explored strategies for the identification of road distress. Current methods use a variety of sensors such as inertial measurements, 3D scanners, and optical sensors. However, in the already defined digital world, the reporting and identification of potholes still depends mainly on public reporting. In this chapter, a general review is given based on various road distress detection techniques.

2.1 Road distress detection techniques

Over time, an extensive literature has developed in the field of pothole detection. Based on our background study, we have divided road distress techniques as shown in Fig. 2.1. From the end user's point of view, these technologies can be categorised based on the sensors being used in the vehicle. In this study, we categorise the methods mainly based on the used techniques.

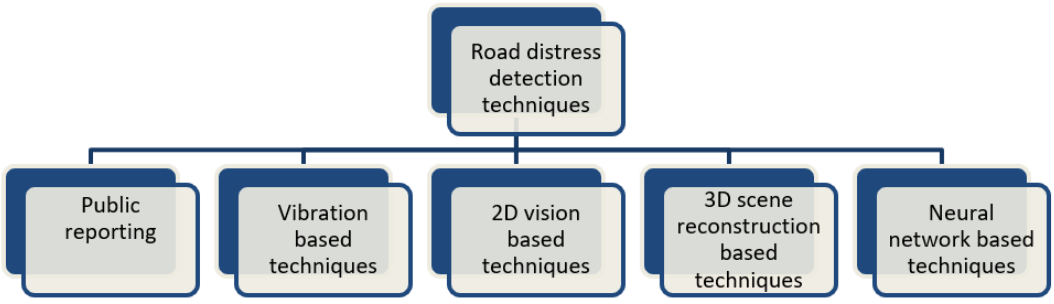

Figure 2.1: Classification of road distress detection techniques

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

Table 2.1: Public reporting; listed names on the left identify the websites of those applications (e.g. `www.fixmystreet.com`). Citizen Hotline 1999 is the name of an innovative open data platform used in Taiwan; the related research publication was in 2017 [16].

2.1.1 Public reporting

This type of system enhances civic engagements by government and facilitates participation by people in the country. This system uses humans as sensors. The public can report potholes by capturing its pictures through their phones or other devices and by uploading or merely sending information about locations to a website or application. However, this approach is manual and time consuming. Some of the examples are listed in Table 2.1.

2.1.2 Vibration based techniques

Vibration based techniques include approaches of collecting abnormal vibrations caused in the vehicles while driving over road anomalies [20]. Vibrations of the vehicle are collected using accelerometer. The main drawback of the vibration based methods is that the vehicle has to drive over the distress in order to measure the vibrations caused by the distress on the road. J. Ren et al. [28] used `K`-Means clustering to detect potholes based on the data collected through an accelerometer and the *global positioning system* (GPS). The proposed system lacks with regards to the isolation of detected potholes from other road anomalies.

M. Ghadge et al. [29] used an accelerometer and GPS to analyse the condition of roads to detect locations of potholes and bumps using a machine-learning approach of `K` means clustering on training data and a random forest classifier for testing data. `K` means clustering divides the data in two clusters of pothole or non-pothole.

Then, a random forest classifier is used to validate the proposed approach of the clustering algorithm. However, clustering does not perform well when clusters of different size and severity are involved and here, size and severity of a pothole are the major factors involved in the system.

M. Badurowicz et al. [30] proposed a road-relative unevenness index scale term to measure road quality using accelerometer data and informing other users of the road using crowd-sourcing.

Eriksson et al. [31] proposed a pothole patrol system to detect potholes which gathers data from vibration and GPS sensors and processes this data to access road surface conditions.

Other researchers such as K. Chen et al. [32], N. Kalra et al. [33], M. Fekry et al. [34] and Y. Ren et al. [35] also used an accelerometer to detect road distress.

2.1.3 2D vision based techniques

Vision-based methods use 2D images or video data, captured using a digital camera, and process this data using image or video processing techniques [36, 37]. The choice of the applied image processing techniques is highly dependent on the application for which 2D images are being processed. These methods [21, 22] rely heavily on manual processing; it appears impractical to define 2D features of potholes due to their irregular shape.

C. Koch et al. [22] proposed a method aiming at a separation of defect and non-defect regions in an image using histogram-shape-based thresholds. The authors consider the shape of a pothole as being approximately elliptical based on a perspective view. The authors emphasize on using machine learning in future work, and claim that the proposed work already results in 86% `accuracy` along with 86% `recall` and 82% `precision`, with the common definitions of

$$\texttt{Precision} = \frac{\texttt{TP}}{\texttt{TP}+\texttt{FP}} \tag{2.1}$$

$$\texttt{Recall} = \frac{\texttt{TP}}{\texttt{TP}+\texttt{FN}} \tag{2.2}$$

$$\texttt{Accuracy} = \frac{\texttt{TP}+\texttt{TN}}{\texttt{TP}+\texttt{TN}+\texttt{FP}+\texttt{FN}} \tag{2.3}$$

$$\texttt{F1} = 2 \cdot \frac{\texttt{Precision} \cdot \texttt{Recall}}{\texttt{Precision}+\texttt{Recall}} \tag{2.4}$$

where TP is the number of true positives, FP of false positives, TN of true negatives, and FN of false negatives.

A. Tedeschi et al. [38] recently suggested a system for *automatic pavement distress recognition* which is able to perform in real time by identifying road distress including fatigue cracks, longitudinal and traversal cracks, and potholes. The authors used a combination of 2D vision-based technologies, and, for the classification, three different types of road distresses. Three classifiers have been used based on local binary pattern features; they achieved more than 70% for `precision, recall`, and the `F1-measure`. Authors discussed difficulties of defining the severity of considered kinds of road distresses. For texture classification, the authors used Haralick's features [39] based on gray-level co-occurrence matrices (GLCMs) and then classified image regions using a tool from [40].

S.-K. Ryu et al. [41] proposed a method to detect potholes both for asphalt or concrete road surfaces using 2D images collected by a mounted optical device on a survey vehicle. The system mainly works in three steps of image segmentation, candidate region extraction and decision. The system fails to detect potholes in darker images (image regions) due to shadows (e.g. of trees or cars) present in real-world road recordings.

L. Powell et al. [42] presented a method for the detection of potholes by segmenting images into defected or non-defected regions. After extracting the texture information from defected regions, this texture information is compared with texture information obtained from non-defected regions. The proposed system considers shadow effects on the road and aims to remove those effects of shadows using a shadow-removal algorithm. The system is unable to perform in rainy weather. The authors concluded that the system should be further extended to perform also on video data as the system was only tested on 2D images collected using an iPhone camera with 5 megapixel image resolution.

V. Bashkar et al. [43] propose a methodology of detecting mean depth of potholes by using SURF [44] features on uncalibrated stereo pairs of images (without employing disparity images). A particular methodology has been developed for this purpose, but appears to suffers from uncalibrated stereo rectification; it is far from providing good results.

Z. Ying et al. [45] proposed a system which can detect road surface based on a feature detector which is shadow-occurence optimised. This system uses a con-

nected component analysis algorithm and other morphological algorithms. The authors had demonstrated their findings using images of datasets provided by KITTI [46] and ROMA [47].

M. V. Thekkethala et al. [48] used two (stereoscopic) cameras and applied stereo matching to estimate depth of a pothole on asphalt pavement surface. After performing binarization and morphological operations, a skeleton of a pothole is estimated. The system is tested on 24 images. The system can detect skeletons of potholes of great depression. Authors did not estimate the road manifold.

A. Akagic et al. [49] used 2D vision based method for pothole detection achieving an accuracy of 82%. The method works by performing image segmentation on extracted pothole areas from an RGB colour space image. Then, a search is performed in the segmented image which contains the ROI. This method is effective as a pre-processing stage for a supervised algorithm.

2.1.4 3D scene reconstruction based technique

3D scene reconstruction is the method of capturing the shape, depth, and appearance of objects in the real world; it relies on 3D surface reconstruction which typically demands more computations than 2D vision. Rendering of surface elevations helps to understand accuracy during the design of 3D vision systems. 3D scene reconstruction can be based on using various types of sensors, such as Kinect, stereo-vision cameras, or a 3D laser. Kinect sensors are mainly used in fields of (indoor) robotics or gaming.

3D lasers define an advanced road-survey technology; compared to camera-based systems it still comes with higher costs; [50, 51] report survey cycles of (usually) once in four years. A 3D laser uses a laser source to illuminate the surface and a scan camera for capturing the created light patterns. The authors of [52] applied the common laser-line projection; the recorded laser line deforms when it strikes an obstacle (and supports thus the 3D reconstruction), but does not work well, e.g., on wet roads or potholes filled with water.

Stereo vision cameras are considered to be cost-effective as compared to other sensors. Stereo vision aims at effective and accurate disparities, calculated from left-and-right image pairs, to be used for estimating depth or distance; see, for example, [55]. commonly, the canonical left-right calibrated stereo camera setup is used while aiming at a reconstruction dense 3D surfaces. A *disparity map* represents per-pixel

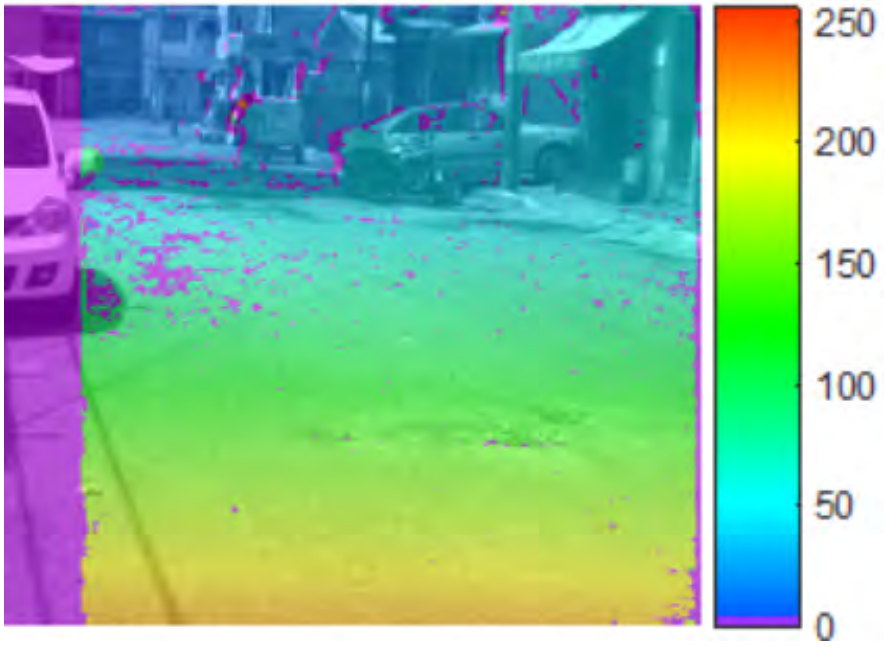

Figure 2.2: Recorded road scene with transparent colour-encoded disparity map. The used colour key is shown on the right; disparity 250 encodes a distance very close to the host vehicle and 0 encodes "very far away" and not matched.

correspondences for a rectified stereo pair. Figure 2.2 illustrates a recorded 3D scene with a calculated (colour-encoded) disparity map. We discuss more about stereo vision in Chapter 3.

The authors of [53] use a stereo vision based system embedded on an *unmanned aerial vehicle* (UAV) to inspect road conditions using disparity maps. The authors concluded that combining disparity maps with other stereo vision algorithms will help to reconstruct 3D maps.

Table 2.2 summarises a few 3D reconstruction-based methods for detecting road distress.

T. Garbowski et al. [56] presented a semi-automatic *pavement failure detection system* (PFDS) which is a part of the FEMat [57] road package. It allows a user to inspect the condition of road pavement based on calculated clouds of 3D points. The presented system considers a small *region of interest* (ROI) in reference to a larger region of a road surface and is able to detect certain types of cracks including "alligator

Table 2.2: Examples of 3D reconstruction-based methods and used sensors

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

cracks", but not potholes.

A. Rasheed et al. [58] presented a technique to stabilize 3D images of pavement to reduce the effect of noise induced into the data during image acquisition process. The authors used Kalman filter with affine transformations to stabilize 3D pothole images which are acquired through kinect sensors.

T. Shen et al. [59] propose the use of Takata's stereo-vision system for performing a road surface preview along the host vehicle. Video data recorded with the used compact stereo-vision sensor (with a baseline of 16.5 cm) is analysed in an embedded system, already tested in various vehicles, also in combination with various driver assistance systems such as *forward collision warning* (FCW), *automatic emergency breaking* (AEB), or *lane departure warning* (LDW). Authors state that the proposed system achieves satisfactory accuracy; they also state that it does not perform well when there is glare on the road surface.

Calculated disparities within detected road-surface image segments support the estimation of a *manifold*, approximating the road-surface. Commonly the road surface is assumed to be planar (i.e., the manifold is thus a plane). But this planarity

assumption is often not corresponding to actual uneven road surfaces. To simplify, the road manifold is often modelled in driving direction by a *profile*, i.e. a curve whose parallel translation left-to-right creates the road manifold. A line creates a plane, and a quadratic polynomial profile creates a quadratic road manifold.

Quadratic road manifolds are discussed by X. Ai et al. [23]. In [23], a stereo vision-based algorithm is proposed for the detection of obstacles that are protruding from the road surface. The algorithm applies a quadratic surface fitting technique on a disparity-derived 3D point cloud to model the road manifold. To build a DEM, 3D points are mapped into discrete 2D grids.

After fitting road surface points into a quadratic surface, an obstacle mask is attained. A connected component analysis proceeds to segment the derived obstacle disparity histogram. Obstacles are detected using minimised bounding boxes with a constraint that no obstacle shall sit underneath the quadratic road surface.

For visualisation, the authors back-projected a high-level descriptor to the image domain. By using domain representations, a connected component algorithm, and minimised bounding boxes for object detection, complex objects have been identified at high accuracy. The authors concluded that their algorithm can be applied to stereo vision data towards optimised disparity search.

For a consideration of twisting and bending surfaces of roads, see A. Mikhailiuk et al. [60]; their algorithm has been implemented on a Texas Instrument C6678 multi-core SoC digital signal processor.

Z. Zhang et al. [61] proposed an efficient algorithm to estimate the size, depth, position and severity of potholes by modelling the road surface as a quadratic manifold by using a *random sample census* (RANSAC) approach. Pothole detection and segmentation are achieved by using a *connected-component labelling* (CCL) algorithm.

Staniek [62] emphasized on solving the problem of matching points in stereo images by using a Hopfield neural network which is a form of a recurrent neural network. The author used stereo-vision cameras, GPS, and inertial sensors; a Hopfield neural network is studied for detecting depth discontinuities assuming that they occur due to sudden significant changes in intensities.

The author also considered the subproblem of discontinuities at border fragments in stereo image pairs. The proposed Hopfield neural network is based on energy minimization in the neural network. The author achieved 66% `accuracy` when evaluating matching pixels for 50 image pairs using a CoVar method [63] for

evaluation.

Other sensors such as ground penetrating radar [27], 2D laser scanners [25], 3D laser [26], or stereo vision cameras [55] offer an accurate option based on 3D reconstruction [24]. Ground penetrating radar is mainly used for special purposes. Stereo vision cameras or a 3D laser are the logical choice for identifying a pothole from a distance, where a 3D laser is (still) an expensive sensor, also usually not applicable when a pothole is filled with water.

2.1.5 Learning based techniques

In these techniques, a network is usually trained for the task of object detection where the choice of the network is dependent highly on the required output. Since the advancement in technologies such as fast processors and memories, lots of progress has been achieved in this field. The choice of training a network has shifted from machine learning in general to deep learning in particular. Deep learning requires lots of data to be processed, however, using data augmentation processes and transfer learning it is possible to train the deep neural network using less data.

H. Song et al. [64] use a *convolutional neural network* (CNN) approach to detect potholes. The authors used a smart phone as a sensor to acquire movement information and the Inception V3 [65] classifier; they adapted the final fully connected layer in the CNN. H. Maeda et al. [66] used a state-of-the-art CNN, trained by using a vast dataset of road images collected in Japan to detect road-surface damage. The authors used SSD Inception V2 [67] and SSD MobileNet [68] to identify different sorts of road damages. Detected road damages are identified by generating

Table 2.3: Examples of learning based methods

Authors	Type	Object
H.Song et al.	CNN	Potholes
H. Maeda et al.	CNN	Cracks
A. Zhang et al.	CNN	Cracks
N. D. Hoang et al.	NN and SVM	Potholes
Y. Cha et al.	CNN	Cracks
J. Bray et al.	NN	Cracks
L. Zhang	CNN	Cracks

bounding boxes.

As examples for the second type, A. Zhang et al. [69] suggested a CrackNet to predict class scores for all the pixels in existing road damage. Detected road damages are identified within enclosed bounding boxes.

V. Badrinaraynan [70] proposes *SegNet*, a multiclass deep-encoded-decoder-based CNN, that is more memory-efficient than the FCN and performs semantic pixelwise segmentation. SegNet eliminates the need of upsampling, as this decoder uses pooling indices, computed in the max-pooling step of the corresponding encoder, for non-linear upsampling.

N.-D. Hoang [71] presented pothole detection on asphalt pavement surfaces; the model is trained using two machine learning algorithms - artificial neural network and least squares support vector machine. Various image processing algorithms are first used for feature extraction processes. Experiments show that least-squares-support-vector-machines perform better than artificial neural networks, with 89% and 86%, respectively.

Y. Cha et al. [72] proposed a classifier build using a CNN for classifying crack damage detection using concrete images. The classifier built by the authors is less influenced under shadow casting and illumination conditions. The authors mentioned that the classifier learned feature extraction process automatically without any dedicated feature extraction process and involved computation as used by conventional approaches.

J. Bray et al. [73] used a NN based binary classifier to classify whether an image belongs to a crack or normal road image. The network accepts the feature of the images before classification process.

L. Zhang et al. [74] also proposed a low-cost solution using deep CNN for identification of road cracks. K. An et al. [75] used various pre-trained CNNs to classify whether an image has pothole or not. The authors concluded that models achieved high accuracy when given gray-scale images.

Extensive research has been carried out already for mainly image segmentation using CNNs (see Table 2.4) such as PSPNet [76], RefineNet [77], or Large-Kernel-Matters [78]. CNNs for image segmentation may also be of relevance, such as the *fully convolutional neural network* (FCN) by Long et al. [79], in which a final fully-connected layer is replaced by another convolutional layer for a large receptive field to capture the global context of a scene. However, this results into coarse segmenta-

Table 2.4: Examples of CNNs for image segmentation, and used data sets

CNN	Year	Used datasets
FCN	2014	PASCAL VOC
$SEGNET$	2015	CamVid
$DilatedConvolutions$	2015	VOC2012, COCO
$DeepLab$	2014-2017	PASCAL 2012, CityScapes
$RefineNet$	2016	PASCAL 2012
$PSPNET$	2016	PASCAL 2012, CityScapes
$LargeKernelMatters$	2017	PASCAL 2012, CityScapes
$MaskR-CNN$	2017	COCO

tion maps by upsampling layers of the FCN.

One more class of CNNs, which uses dilated or atrous convolutions, is proposed in DeepLab by Chen [80]. However, this type of convolutions is computationally very expensive because of its application for high-resolution feature maps.

The authors [81] have proposed a model to detect potholes based on YOLOv2 architecture. However, their reported architecture differs from our proposed model in *LM2*. Also, the tested frames basically show not much more than potholes, while the real road scene is much complex.

The CNN model proposed by the authors [82] to detect potholes has been trained on a CPU and experiments shows that CNN based model perform better than Conventional SVM based approach. However, the system is not able detect potholes under varying illumination conditions.

The authors [83] have proposed a CNN based model mainly to classify a region on a road as pothole or non-pothole. The author has collected the dataset using smartphone camera mounted on the front windshield of the vehicle and the authors have used preprocessed cropped frames with ROI to train the proposed model. The authors [84] have developed CNN based model using thermal images to classify whether an image has pothole or not. The thermal images are recorded using a thermal camera.

The authors of [54] use a UAV to capture road images and classify them into normal or damaged pave images using four different techniques of SVM, *artificial neural network* (ANN), random forest, *K-nearest neighbour* (KNN). The authors concluded that advanced learning based methods such as CNN has a greater potential

to assess the road condition.

There is currently strong progress towards object detection and recognition based on deep learning; a common issue is the lack of training data. Therefore, reflecting this common problem in our paper, we detect potholes transfer learning based approach.

2.2 Multi-sensors based approach

Proposed methods may also follow a multi-sensor approach [85]. Y. -H. Tseng et al. [86] developed an automated survey robot which performed in simulated test-field environments to detect five types of distress, namely alligator cracks, small patches, potholes, rectangular, and circular manhole covers.

F. Seraj et al. [87] used a *support vector machine* (SVM) as a machine learning approach to classify road anomalies. The proposed system uses accelerometer, gyroscope and Samsung's galaxy as sensors for data collection; data labelling is performed by a human. A high-pass filter is used to remove the low frequency components caused due to turns and accelerations.

Naidoo et al. [88] suggested two prototypes; a first prototype uses stereo-vision cameras, and a second prototype uses one *gigabit ethernet* (GigE) high-speed camera. The authors have designed a two-stage neural-network (NN) based classifier that takes frequency-transformed *hue, saturation, value* (HSV) images as input to achieve pothole classification, and trained the networks using a recorded video of selected roads in Gauteng, South Africa.

The first NN classifies a section of a road in an image, while the second detects potholes in that section. The trained NN, however, was not able to perform inferences in real time. The authors tested on two video sequences with 26 and 9 potholes that were manually counted, and compared with the results of the NN classifiers. The reported accuracy is 73% for the first video, and 100% for the second video.

Hsu et al. presented in [89] a multi-sensor approach to detect potholes and for measuring a road quality index by integrating laser, camera, GPS, and an *inertial measurement unit* (IMU) into an experimental golf-cart system. The analysis is based on imagery as well as laser-scanned 3D data. Lin and Liu [90] used a collection of pavement images for an SVM to distinguish potholes from other defects such as cracks. The authors have proposed an SVM-based method which is used for classification of distresses on road. The author has extracted histogram based on texture measure from image region and the experiment has been carried out on 80 images of size 64×64.

To achieve satisfactory results from the non-linear SVM, image normalization has been performed. The authors conclude that the system needs further research as potholes filled with water cannot be identified correctly. A. Kulkarni et al. [91] proposed a pothole detection algorithm by using a high frequency filter and an NN

approach.

C. Zhang et al. [92] presented a UAV (unmanned aerial vehicle) based imaging system that is capable of providing 3D information of unpaved surface distresses (ruts etc.) for further road condition assessment. For the data acquisition part, the system itself consists of a camera mounted on a mini UAV helicopter capable of flying 200 metre above the ground and travelling at a maximum speed of 10 metre per second. The authors had mainly developed a technique for bundle adjustment in image orientation and then processed 3D reconstruction and measurement of surface distresses.

Chapter 3

Basics

This chapter covers the basic notations, theories and underlying concepts later adopted in this thesis. Sections 3.1 to 3.3 cover the fundamentals of computer vision approaches used in Chapter 4, and Sections 3.6 to 3.8 are dedicated to the basics of deep learning techniques used in Chapter 5. The chapter also provides an introduction to the different datasets used in this research.

3.1 Stereo-vision based techniques

Stereo vision is inspired by the human vision system. The two perspective views of human eyes provide information about the slight visual offsets of objects in the world. These two views are sent to the brain for the processing of matching similarities while also understanding the differences in the two views. The obtained outcome is a 3D stereo image. Human beings do it several times per second in order to obtain the actual distance to visible objects, which is very important in many cases such as for driving.

The word stereo originated from the Greek word *stereos* which means *solid*. Stereo vision helps us to see a solid object in the 3D world along with its height, width and depth information.

The slight visual offset of an object surface point is known as a *disparity*, which leads to the phenomenon of depth perception. Depth estimation of the stereo perspective views originating from a 3D scene is determined by using pixel correspondences of similar features in different views.

We use a left-hand image coordinate system as shown in Fig. 3.1, where I denotes an image and (x, y) are the image pixel's coordinates. According to the left-hand coordinate system, the image will have its origin at the upper left corner and the

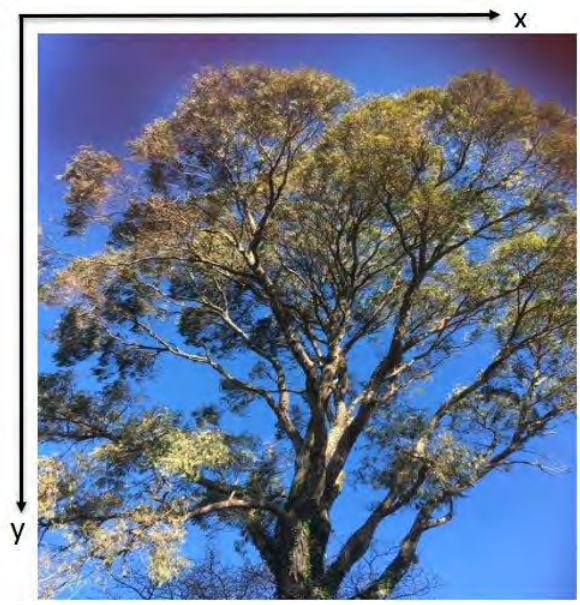

Figure 3.1: Left hand image coordinate system (picture taken at author's garden, Auckland, New Zealand)

positive x points right and y points down. By $p = (x, y)$ we denote the pixel location in the 2D image grid domain denoted as Ω. The cardinality $|\Omega|$ denotes the number of pixels in the domain Ω. We use upper case bold letters for matrices.

3.1.1 Stereo vision

Stereo vision uses two or more cameras to acquire the 3D information of a scene. In this study, we detail road distress detection concepts based on two cameras. To avoid additional difficulties, the two stereo-vision cameras are ideally identical twins placed on a horizontal bar (see Fig. 3.2). Our experiments throughout Chapter 4 use calibrated and rectified images, corresponding to such an ideal camera set-up. As we use rectified images, our dataset includes left and right camera images having the same principal point (x_c, y_c) and identical focal length f.

The main motive is to obtain two different views of the current scene being observed. For a pixel at (x, y) in the left camera image, we search for a corresponding pixel at $(x - d, y)$ in the right camera image, where d is the disparity value later de-

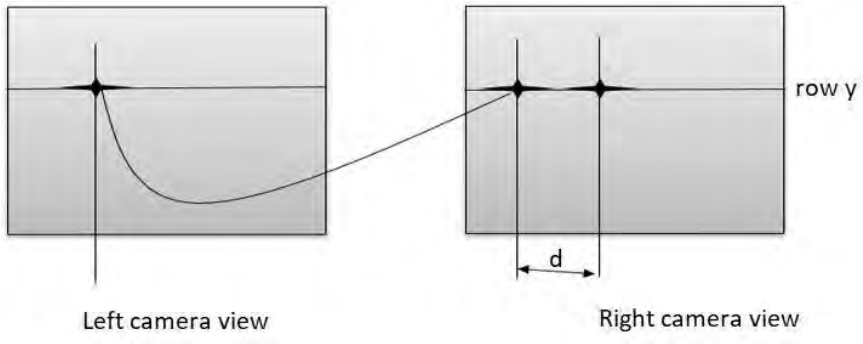

Figure 3.2: Illustration of a stereo image pair. The curved line connects a stereo pair (i.e. two corresponding pixels) at row y. Here, d denotes the resulting disparity

scribed in Section 3.1.2. The search for a corresponding pixel is performed along a horizontal line, known as the epipolar line. Using this epipolar line, the search space is reduced from 2D to 1D. This epipolar line is defined by epipolar geometry which emerged from the following basic definitions:

- An epipole is a projection of one projection centre into the image plane of the other camera.

Figure 3.3: Gray-level images (left and right) captured by the left camera as available in the `CCSAD` dataset

- The base line connects the projection centres of both cameras.
- An epipolar plane is defined by both projection centres and a selected pixel location in one image plane.
- The epipolar plane intersects the second image plane at the epipolar line, which is the possible search space for the corresponding pixel.

In the ideal camera setting (i.e. rectified epipolar geometry) briefly mentioned above, the epipolar lines coincide with image rows, thus simplifying the search for corresponding pixels. This search is, for example, difficult due to large regions showing only minor differences in intensity textures (see Fig. 3.3).

3.1.2 Disparity map

Disparity calculation is a computationally expensive process as it measures differences in appearance along epipolar lines in the left image compared to the right image. The epipolar geometry is rectified if image planes are co-planar and the principal axes of both cameras point into the same direction [55]. The left and right images

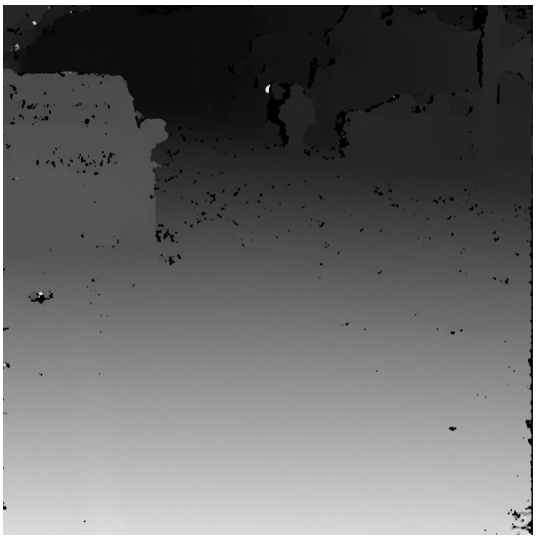

Figure 3.4: Disparity map computed for stereo frames illustrated in Fig. 3.3 using a gray-level key for visualising different disparities (bright is close and dark is far away)

used in a correspondence process are addressed as reference and matching image respectively. During this correspondence process, all objects are "shifted" in the same direction and every pixel is assigned a positive value. This "correspondence shift" is between $p = (x, y)$ and $p\prime = (x - d, y)$, where $d \in \mathbb{R}_+$ is upper-bounded by d_{max}. A *disparity map* is visualised in Fig. 3.4.

The maximum disparity value d_{max} means that the object is closest to the camera, whilst $d = 0$ means that the object is (theoretically) at infinity.

3.1.3 Depth map

An obtained disparity map D is used to derive real-world localisation information, known as a depth map. The transformation of a pixel at (x, y) with disparity d into a 3D point (X, Y, Z) is basically a distance assignment to the pixel at (x, y) in the image plane, where X, Y, and Z denote 3D coordinates, with Z being measured along the optical axis of the used camera, also called *depth*. Here, we use a left-hand image coordinate system where positive rotation is clockwise about the axis of rotation.

The depth Z at pixel (x, y) is estimated using calibrated camera parameters as follows:

$$Z = f \cdot \frac{b}{d} \tag{3.1}$$

where f is the focal length, b is the length of the baseline of the two horizontally

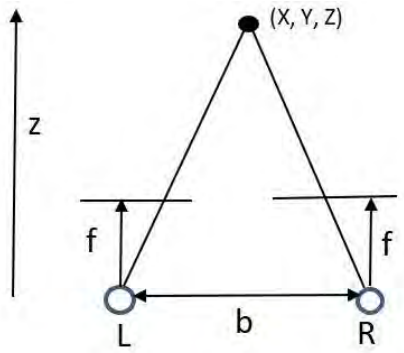

Figure 3.5: 3D coordinate estimation using binocular stereo-vision cameras. Coordinates X, Y, and Z are calculated using calibrated parameters f_x, b, and coordinates (x, y) and $(x - d, y)$ of corresponding points in left (L) and right (R) images.

placed stereo vision cameras (measured in world units) and d is the disparity at pixel (x, y). Note that a depth map is inversely proportional to the disparity map.

Coordinates X and Y can be calculated similarly to Eq. (3.1) using coordinates x, $x - d$ and y of two corresponding pixels [55] (see Fig. 3.5). The start pixel has coordinates (x, y) in the left image, and the matched pixel has coordinates $(x - d, y)$ in the right image.

3.2 Visual odometry

The term *visual odometry* (VO) was first introduced by Nister in [93]. Odometry denotes the use of motion sensors for estimating the change in position over time. For example, vehicle odometry may be based on a wheel odometer, the wheel odometer measuring the number of rotations the wheel goes while driving; and being displayed as an odometer reading placed into the dashboard of a car. Likewise, VO estimates the position of a vehicle by examining the changes induced in images recorded by its on-board cameras. These changes are induced by the motion of the vehicle and can be effectively examined under proper illumination.

A wheel odometer is usually affected by wheel movements due to sudden swerves or uneven surfaces. VO also comes with its own challenges.

VO techniques are often complemented by the use of a *global positioning system* (GPS). As GPS usually does not work in extensive tunnels or basements, VO provides useful additional input. The goal of VO is to solve egomotion estimation of a vehicle or robot, using recorded video data only.

Research for the estimation of the egomotion of a vehicle, using visual input only, started in 1980 [93]. Moravec contributed by presenting a first motion-estimation pipeline using a class of stereo VO algorithms. There have been many developments over the decades in the two fields of VO – monocular VO and stereo VO.

Monocular VO is a challenging class of VO as an accurate examination of a scene is not directly recoverable from monocular cameras. In both of these VO classes, the core components are selecting feature detectors and feature tracking to estimate different motion correspondences. Motion estimation can be done for 2D-to-2D, 3D-to-3D or 3D-to-2D correspondences. Given,

$$I_k^c : \Omega_c \subset \mathbb{R}^2 \to \mathbb{R} \tag{3.2}$$

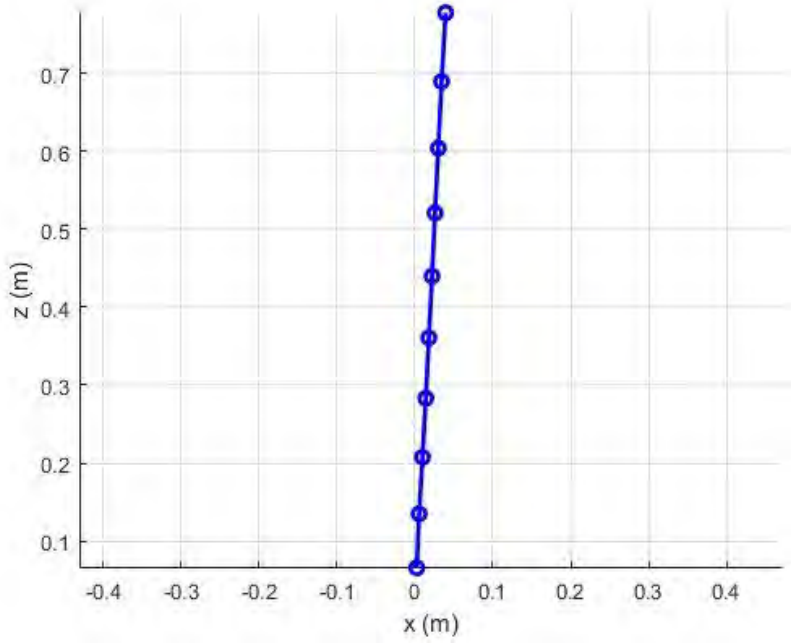

Figure 3.6: An example of recovered camera trajectory for `CCSAD urban sequence 1` frames $70 - 79$.

where I is the image of frame k and Ω_c is the image domain of camera c.

An N-camera VO problem is to generate a sequence of transforms $(T_1, ..., T_S)$ with a sequence of S images. Here the images are $(I_1^c, I_2^c, ...I_S^c)$ and $1 \leq c \leq N$. By solving a series of two-frame pose estimations subsequently, we obtain G_k, the "gradient" from the pose of the N cameras at time $k-1$ to their pose at time k. To be precise, G_k defines an affine transform from the pose at time $k-1$ to the pose at k, where an affine transformation is a transformation to preserve collinearity.

The VO problem is mainly solved using the following steps:

- It starts with pre-processing of new frames such as image undistortion, rectification and extraction of features.
- The image correspondence is then established by associating identified key points with the previously tracked elements. This correspondence is used to

solve the local transform of a camera.

- Once VO is solved, the new measures are integrated into the current system state. This process refines the structure of the scene.

The image correspondence acts like a bridge between the structure and motion of the camera. Mainly two types of techniques are used to establish correspondence in subsequently recorded images:

1. Direct intensity matching performs direct matching on pixel intensities and has been researched extensively in computer vision such as [94–96]. We use the *Kanade-Lucas-Tomasi* (KLT) method later in the thesis to detect some key points from a frame and track it as a new frame arrives. In KLT, we constrain the search space along the epipolar line when camera motion is obtained because it performs correspondence along an image gradient-guided path. By constraining, the dimensionality of the search space is reduced from 2D to 1D.

2. Feature descriptor matching is used when the camera pose is not known. In this case, sparse point correspondence between frames is established in feature space. The feature descriptor matching obtains a sample of accurate points even when the image displacement is very large. The epipolar constraint in feature space is difficult to enforce as feature transformation usually does not preserve the geometry of an image. Using a robust outlier algorithm, the incorrect correspondence points can be rejected before motion estimation (see Fig. 3.6).

3.3 Speeded up robust features

Speeded up robust features (SURF) is a computationally efficient algorithm for local similarity invariant representation in images [44]. Interesting keypoints are defined as salient features such as several other local descriptor-based techniques.

The SURF algorithm uses an approximated box filter second-order derivative (see Fig. 3.7) computation to locate extrema in the scale space, which is efficiently implemented using an integral image. The use of box filters enables SURF for fast real time application including object tracking. The integral image is used for calculating the accumulated intensity of an image for a pixel value $p = (x, y)$ as follows:

$$I_{int}(p) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j) \tag{3.3}$$

Here $I_{int}(p)$ is an integral image presenting the sum of all pixels in an image up to point p. SURF depends on the determinant of Hessian matrix for location and scale.

Given a point $p = (x, y)$, the Hessian matrix $M(x, \sigma)$ at scale σ is defined as follows:

$$M(p, \sigma) = \begin{bmatrix} C_{x,x}(p.\sigma) & C_{x,y}(p.\sigma) \\ C_{x,y}(p.\sigma) & C_{y,y}(p.\sigma) \end{bmatrix} \tag{3.4}$$

where the element of the matrix M is are the values produced by the four convolution masks. The (weighted) determinant of the Hessian matrix is calculated as follows:

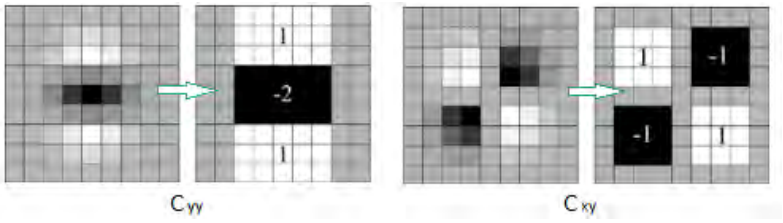

Figure 3.7: Approximations illustrated for $\sigma = 1.2$ (lowest scale) and 9×9 Gaussian partial derivative (Gaussians are discretised and cropped) in xy and yy. C_{yy} is the derivative in the y direction with approximation of SURF shown with green arrow. Similarly, C_{xy} is the derivative in the diagonal direction starting from lower left side to the upper side.

Figure 3.8: Original image on left side and transformed image on right side. The transformed image is resized at 0.7 times the original image and rotated at an angle of 10° in the counterclockwise direction.

$$M(p,\sigma)_{approx} = C_{x,x}(p.\sigma) \cdot C_{y,y}(p.\sigma) - (w \cdot C_{x,y}(p.\sigma))^2 \tag{3.5}$$

where w is a weight optimisation constant following [44]. The local maximum value of $M(p,\sigma)_{approx}$ provides a keypoint p. This keypoint is detected in a $3 \times 3 \times 3$ array of M-values.

To extract the SURF descriptor, a square region centred around the keypoint is constructed. This region is divided into 4×4 square sub-windows and for each of these smaller sub-windows a small number of features at regularly spaced sample points of 5×5 are computed. These wavelet responses are added for all the sub-windows. The SURF descriptor which is a 64-vector of floating point values combines local gradient information with weighted sums in sub-windows [99].

3.4 Random sample consensus

Random sample consensus (RANSAC) is a non-deterministic iterative algorithm to remove outliers during model fitting; in our case we apply RANSAC for robust plane fitting [97]. Random sampling picks a small set of points randomly and fits the model to them. Eventually, there will be a sample set of points that does not

contain (many) outliers anymore. The consensus set is the part of the model that helps to determine the set of data points that best fits with the model. The solution is the largest consensus set which fits the model under hypothesis.

The assumption of the RANSAC algorithm is that the training data consists of inliers explained with the model and outliers are erroneous and do not fit the model. It uses outliers while training the model to increase the final prediction error as it trains the model only using inliers while ignoring outliers. RANSAC uses small samples of data to train a model and enlarges this sample set by including samples within the error tolerance of the training model. For example, two data samples are enough to determine a linear function. The chosen sample, known as a consensus set, therefore trains the model using relatively smaller samples. This process is repeated for a predefined number of iterations to obtain a model with relatively least errors among all the generated models.

For example, in Fig. 3.9, we derive a geometric transformation from matched keypoints extracted using SURF for Fig. 3.8. In an image, the first keypoints are detected and SURF descriptors are extracted using each keypoint. Then two different samples of descriptors at time $k-1$ and k are compared to do the matching. Given this set of matching keypoints, we use RANSAC for outlier removal.

Figure 3.9: SURF keypoints matched on the left side including outliers. RANSAC for outliers removal for 1000 iterations on right side.

3.5 Lucas Kanade tracking

Object tracking is done to locate an object in successive frames. To understand the motion of the ego-vehicle, the features are detected and tracked in an image sequence. The goal is to select good features in a frame at time $k-1$ and track it in a new frame arriving at time k. This feature point tracking is known as sparse 2D correspondence problem. The *Kanade-Lucas-Tomasi* (KLT) algorithm [96] is a feature tracking algorithm based on the early work of Lucas and Kanade [94]. This algorithm tracks the location of some feature points in an image. KLT tracking is preferred due to its computational benefits in time critical applications.

The goal is to match a base window B_k of size $(2c+1) \times (2c+1)$ enclosing a keypoint p present in the base frame I with window $B_{k,\mathbf{d}}$ in the match frame J, where the procedure should be robust to allow for scaling, rotation and translation between B_k and $B_{k,\mathbf{d}}$. In Fig. 3.10, $\mathbf{d}$ shows the dissimilarity vector combines the components $\mathbf{t}$ (translation) and h (height scaling) of the centre of a keypoint p in base frame I to match frame J [55]. Figure 3.10 indicates the translation $\mathbf{t}$ by moving the pixel location of keypoint p into a new position defined by translation $\mathbf{t}$. is same in both frames. The KLT algorithm is robust because it approximates derivatives in both x and y directions.

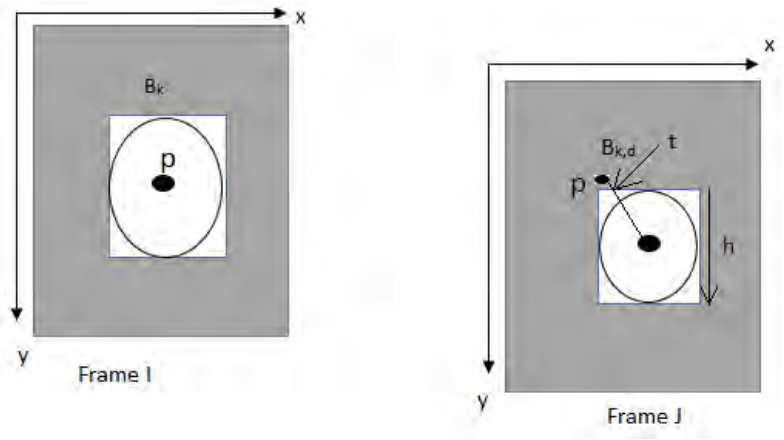

Figure 3.10: KLT tracking for pixel location p.

Translation.

To simplify, we assume that p at (x, y) is $(0, 0)$ and the goal is to calculate a trans-

lation vector $\mathbf{t} = [t.x, t.y]^\top$ where $J(x + t.x + i, y + t.y + j) \approx I(x + i, y + j)$. The aim is to approximate a minimum of following loss functions:

$$L(\mathbf{t}) = \sum_{i=-c}^{c} \sum_{j=-c}^{c} \Big[J(t.x + i, t.y + j) - I(B_k(i, j)) \Big]^2 \tag{3.6}$$

where $-c \leq i, j \leq c$ defines the relative locations in match window $B_{k,\mathbf{d}}$ and base window B_k.

General warping.

To calculate a dissimilar vector $\mathbf{d}$, we apply an interpolation to $J(B_{k,\mathbf{d}}(l))$ as warping does not map a pixel location onto a pixel location. It results from a warping of the pixel location $l = (i, j)$ with a dissimilar vector $\mathbf{d}$. Thus, to calculate $\mathbf{d}$, the aim is to minimise the following loss function:

$$L(\mathbf{d}) = \sum_{l} \Big[J(B_{k,\mathbf{d}}(l)) - I(B_k(l)) \Big]^2 \tag{3.7}$$

Iterative steepest-ascent algorithm.

Next we estimate the shift $s_\mathbf{d} = [s_1, s_2, ..., s_n]^\top$ similar to the mean-shift algorithm in image segmentation, assuming we are at a parameter vector $\mathbf{d} = [d_1, d_2, ..., d_n]^\top$ by minimising the following:

$$L(\mathbf{d} + s_\mathbf{d}) = \sum_{l} \Big[J(B_{k,\mathbf{d}+s_\mathbf{d}}(l)) - I(B_k(l)) \Big]^2 \tag{3.8}$$

To solve this approximation, a Taylor expansion of $J(B_{k,\mathbf{d}}(l))$ is used as follows:

$$J(B_{k,\mathbf{d}+s_\mathbf{d}}(l)) = J(B_{k,\mathbf{d}}(l)) + s_\mathbf{d}^\top \cdot \nabla J \cdot \frac{\partial B_{k,\mathbf{d}}}{\partial \mathbf{d}} + e \tag{3.9}$$

The term $s_\mathbf{d}^\top \cdot J \cdot \frac{\partial B_{k,\mathbf{d}}}{\partial \mathbf{d}} + e$ results in a scalar as we have scalar on left hand side. We use Jacobian matrix [98] of the warp calculated as:

$$\frac{\partial B_{k,\mathbf{d}}}{\partial \mathbf{d}}(l) = \begin{bmatrix} \frac{\partial B_{k,\mathbf{d}}(l).x}{\partial x} & \frac{\partial B_{k,\mathbf{d}}(l).x}{\partial y} \\ \frac{\partial B_{k,\mathbf{d}}(l).y}{\partial x} & \frac{\partial B_{k,\mathbf{d}}(l).y}{\partial y} \end{bmatrix} \tag{3.10}$$

Combining Equations (3.9) and (3.10), we obtain the minimization problem as:

$$\sum_{l} \left[J(B_{k,\mathbf{d}}(l)) + s_{\mathbf{d}}^{\top} \cdot \nabla J \cdot \frac{\partial B_{k,\mathbf{d}}}{\partial \mathbf{d}} - I(B_k(l)) \right]^2 \tag{3.11}$$

Derivation of Equation 3.11 with respect to $s_{\mathbf{d}}$ leads to:

$$2 \sum_{l} \left[\nabla J \cdot \frac{\partial B_{k,\mathbf{d}}}{\partial \mathbf{d}} \right]^{\top} \left[J(B_{k,\mathbf{d}}(l)) + s_{\mathbf{d}}^{\top} \cdot \nabla J \cdot \frac{\partial B_{k,\mathbf{d}}}{\partial \mathbf{d}} - I(B_k(l)) \right] = 0 \tag{3.12}$$

Considering 2×1 vector of zeroes on the right-hand side of Equation (3.12), and using the Hessian matrix:

$$H = \sum_{l} \left[\nabla J \cdot \frac{\partial B_{k,\mathbf{d}}}{\partial \mathbf{d}}^{\top} \right] \left[\nabla J \cdot \frac{\partial B_{k,\mathbf{d}}}{\partial \mathbf{d}} \right] \tag{3.13}$$

Using Equation (3.12) and (3.13), the shift vector $s_{\mathbf{d}}$ of the dissimilarity vector $\mathbf{d}$ is updated to $\mathbf{d} + s_{\mathbf{d}}$ by:

$$s_{\mathbf{d}}^{\top} = H^{-1} \sum_{l} \left[\nabla J \cdot \frac{\partial B_{k,\mathbf{d}}}{\partial \mathbf{d}}^{\top} \right] \left[I(B_k(l)) - J(B_{k,\mathbf{d}}(l)) \right] \tag{3.14}$$

3.6 Basics of transfer learning techniques

Deep learning is a prominent field of artificial intelligence, employing various distinct deep layered neural networks. Deep learning architecture has been in existence since the 1980s; however, it became famous due to recent advancements in hardware technologies. In this section, we discuss the basics of transfer learning techniques used in Chapter 5.

3.6.1 Convolutional neural networks

CNN represents the state of the art because of various improvements in the field of image segmentation, image localisation, and image classification. CNN has derived its inspiration from assumed working principles of the human brain (see Fig. 3.11). CNN is a class of deep learning algorithm which accepts an input image and shares learnable parameters. A basic CNN mainly consists of the following components:

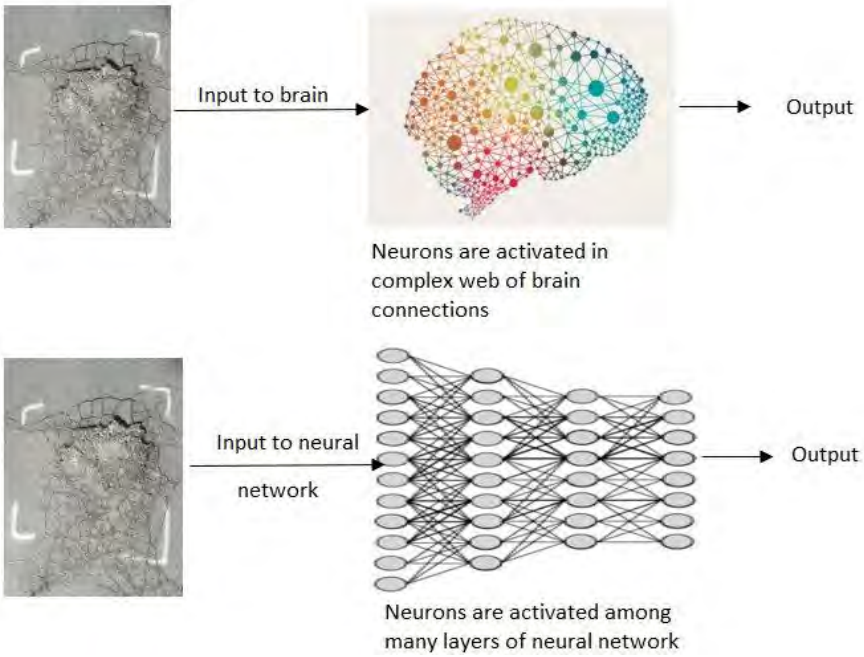

Figure 3.11: The processing of a deep neural network deriving its inspiration from human brain.

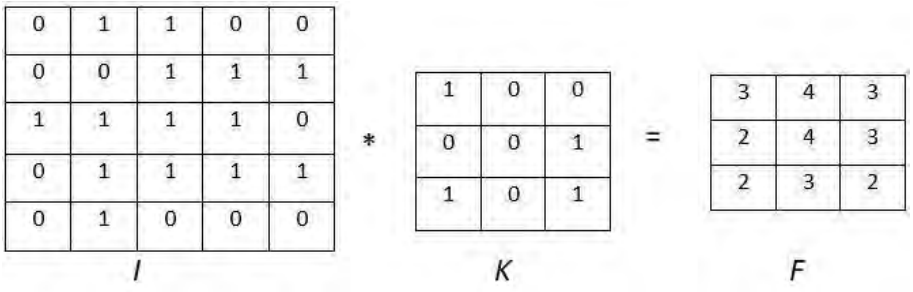

Figure 3.12: I is an input image, K is a kernel and F is an obtained feature map.

- **Convolution layer**.

  Convolution is a term used for the comparison of two functions $I * K$, here input image I is convolved with kernel function K resulting into feature map F.

  The kernel is mainly a collection of numbers known as weights (see Fig. 3.12, the second matrix of numbers, where kernel size is 3×3). This kernel uses a stride value, by which it slides over the image such as in Fig. 3.13, where the stride value is 1. The convolutional layer aims at extracting high-level features present in the image resulting in a (see Fig. 3.12, right matrix) feature map.

- **Pooling layer**.

  Pooling layers decrease the number of parameters and computation required by reducing the dimensionality of each feature map F while retaining important information. Figure 3.13 illustrates the concept of max pooling. This layer works by placing a 2×2 matrix over the feature map and picks the maximum value. This maximum value represents the most present feature in the patch. Average pooling is another type of pooling technique which picks the average value in a feature map.

- **Activation function layer**.

  In neural networks, it is difficult to know the right set of weights at the beginning. Backpropagation is an algorithm to approximate a function with an optimal set of weights. It uses a loss function to compare the predicted results with the actual results by determining the error caused by the individual

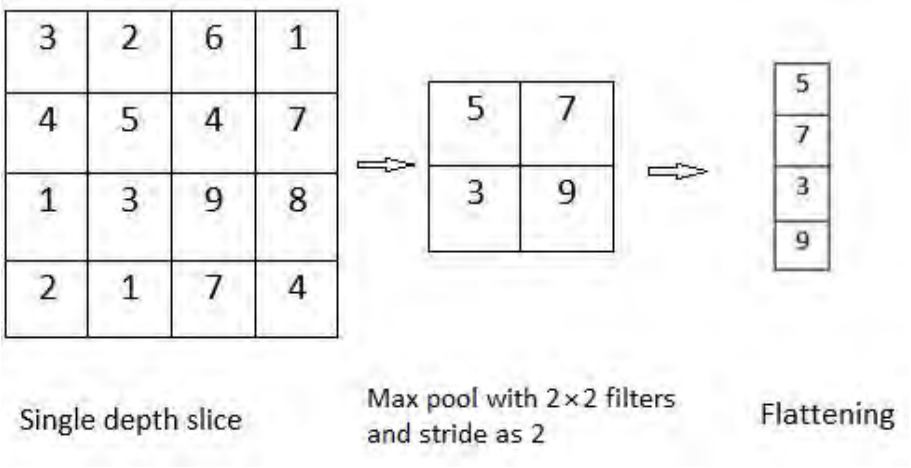

Figure 3.13: Max pooling and flattening.

weight. It does this by backpropagation through the network. The weights are also adjusted during this process. The activation function layer exists between the convolution and the pooling layer. This layer is responsible for transforming the input of summed weights of the node in the network into the output of that node. Activation functions are used to determine the output of the network by introducing non-linearity transformation in CNNs. This transformed output is sent to the next layer for processing. After several layers of convolution with the activation function and pooling layers, the reduced feature size is obtained with extracted complex features. We discuss activation functions later adopted in this research, i.e. *rectified linear unit* (ReLU) and leaky ReLU, which are widely preferred for deep network. The ReLU function is as follows:

$$R(x) = \max(0, x) \tag{3.15}$$

where x is an input value of the node. In the case of leaky ReLU, the negative value has some scope such as for $x \leq 0$ and it can be set as $0.01x$.

- **Flattening process**.

  During this step, the pooled feature map is flattened into a one-dimensional column vector in order to feed to the neural network for further processing (see Fig. 3.13).

Figure 3.14: Image illustrating the concept of knowledge transfer, where a dad is transferring his knowledge of riding skill to his young one.

3.6.2 Transfer learning

The learning and training process in traditional machine learning algorithms happen in isolation without the transfer of knowledge from one domain to other domain. Transfer learning is motivated by the capability of human beings to transfer knowledge (see Fig. 3.14). Transfer learning enables to leverage the knowledge gained from learned tasks and transfer it to the newer tasks. In computer vision, low-level features such as corners or edges can be shared among multiple tasks. The concept of transfer learning goes back to 1995 [100] and it is still an active area of research. *Andrew Ng* mentioned in his course [1] that "After supervised learning, transfer learning is the next driver of the machine learning success".

Transfer learning involves the notions of domain and task [101]. The knowledge gained from a source domain D_s is transferred to a target domain D_t.

A domain D is defined by two components $D = \mathbf{f}$ and $P(\mathbf{m})$ where $P(\mathbf{m})$ is the

[1] https://media.nips.cc/Conferences/2016/Slides/6203-Slides.pdf

marginal probability distribution over the feature space $\mathbf{f}$ and $\mathbf{m} = [x_1, x_2, ..., x_n]^\top \in \mathbf{f}$.

The $\mathbf{f}$ is the space of the data representations, x_i is the i-th component of vector $\mathbf{m}$. The sample of data used for training is denoted by $\mathbf{m}$. The source and the task domains are as follows:

$$D_s = \mathbf{f_s}, P_s(\mathbf{m}) \tag{3.16}$$

$$D_t = \mathbf{f_t}, P_t(\mathbf{m}) \tag{3.17}$$

where $\mathbf{f_s}$ and $\mathbf{f_t}$ are feature spaces and $P_s(\mathbf{m})$ and $P_t(\mathbf{m})$ are the probability distribution for the source and the target.

For each domain i.e. source s and target t, the task T is defined as $T = \mathbf{y}, P(\mathbf{c}|\mathbf{m})$ using label space $\mathbf{y}$ and a conditional probability distribution $P(\mathbf{c}|\mathbf{m})$. The $P(\mathbf{c}|\mathbf{m})$ is learned from training data having pairs of x_i in $\mathbf{m}$ and y_i in $\mathbf{y}$, where $\mathbf{y}$ is the set of all the labels.

Given D_s, T_s, the aim of transfer learning is to learn the T_t in D_t, using the knowledge gained from D_s and T_s, where $D_s \neq D_t$ or $T_s \neq T_t$:

$$T_s = \mathbf{y_s}, P_s(\mathbf{c}|\mathbf{m}) \tag{3.18}$$

$$T_t = \mathbf{y_t}, P_t(\mathbf{c}|\mathbf{m}) \tag{3.19}$$

where $\mathbf{y_s}$ and $\mathbf{y_t}$ are feature spaces and $P_s(\mathbf{c}|\mathbf{m})$ and $P_t(\mathbf{c}|\mathbf{m})$ are the probability distribution for the source and the target.

The transfer learning techniques are different depending upon the source and target domains [101]. Later in Chapter 5, we use *inductive transfer learning using parameter transfer*. In inductive transfer learning, the source and target domains are the same; however, tasks in both domains are different (for more details, please see [101].

At the granular level, the neural networks approximate a function by mapping input values to their corresponding values, which involves several arithmetic computations. This function becomes complex when used with non-linear transformations and stacks of layers. Using this functionality of machine learning, networks are trained to learn anything when provided with enough data and robust compu-

tation. Transfer learning is particularly useful when CNN cannot be trained from scratch, usually due to the lack of massive training datasets. To use transfer learning, one of the following two strategies is used:

1. **Pre-trained model as feature extractor**. The different features are learned at different layers of deep learning model architecture. A fully connected layer is added as last layer to get the final output. The various layered architecture such as ResNet [106] can be used as a feature extractor across many domains. To use a pre-trained model as an architecture, we replace the last layer according to the new task.

2. **Fine-tuning pre-trained model**. In a CNN, the process of feature extraction is coarse in earlier layers and becomes subtle in later layers. In the case of object detection, the later layers extract the information about the position of the object whereas, for object recognition, all layers of CNN serve the common purpose of extracting features of the object being identified. So, depending upon the new task, the layers of the deep learning model are retrained by optimizing different hyper-parameters (see Section. 3.6.3).

3.6.3 Hyperparameters

The models for source and target domains share some hyperparameters, which are some parameters of CNN that are usually tuned in order to have an unprecedented performance of the model being developed. Transfer learning often entails determining the hyperparameters that share an orthogonal relationship with the model. Some of the hyperparameters are:

- **Learning rate**. A learning rate or regularisation parameter defines the learning progress of a model to optimise its capacity. As shown in Fig. 3.15, a high learning rate accelerates the learning process by updating weights with larger values but does not converge, whereas a low learning rate is slow but converges. A model performing well on training data but not on testing data is a sign of - overfitting. A solution to avoid overfitting is to use regularisation techniques. Regularisation helps to generalise the prediction from training to test data. Using regularisation, we penalise the loss function while encouraging the learning algorithm to keep the weights small. Weights in a neural network are real values that are positive or negative and to regularise the weights

in a network, we use a regularisation parameter. At each iteration of training, the network penalises its weights using a loss functions as follows:

$$L(x, y) = \sum_{i=1}^{n} \Big(y_i - f(x_i) \Big)^2 \tag{3.20}$$

where $L(x, y)$ is the loss function, i is $\leq$ n with n as maximum number of iterations and $f(x_i)$ is:

$$f(x_i) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_m x_m$$

where w represents weights and x denotes a number of input variables.

With the increase in number m the complexity of the network increases. To avoid overfitting, w is kept smaller and while penalising the weights, some of the weights become close to zero. However, to ensure all the input variables are taken into account, a regularisation term is added to Equation (3.20). This regularisation term is added to avoid overfitting by keeping all the weights small. We consider:

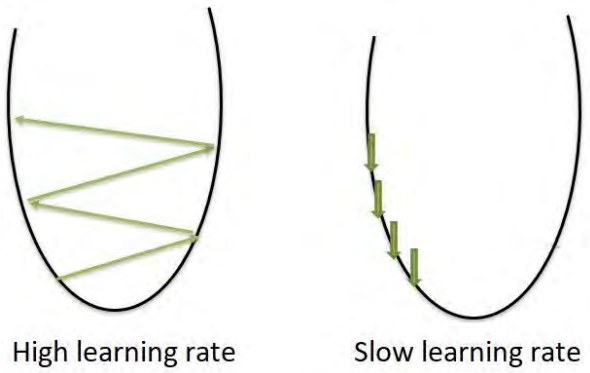

Figure 3.15: The left image with high learning rate shows gradient descent vector oscillating a lot and the right image with slow learning rate shows the gradient vector converging.

Figure 3.16: Gradient vector got stuck in local minima.

$$L(x, y) = \sum_{i=1}^{n} \Big(y_i - f(x_i) \Big)^2 + \lambda \sum_{i=1}^{n} \mid w_m \mid$$

or

$$L(x, y) = \sum_{i=1}^{n} \Big(y_i - f(x_i) \Big)^2 + \lambda \sum_{i=1}^{n} w_m^2$$

where

$$\lambda \sum_{i=1}^{n} \mid w_m \mid \tag{3.21}$$

represents regularisation term using L_1 regularisation which penalises the absolute value of weights, and

$$\lambda \sum_{i=1}^{n} w_m^2 \tag{3.22}$$

represents L_2 regularisation which penalises the sum of squared value of weights.

- **Learning momentum**. The purpose is to converge to global minima in order to reduce error for the predicted target and ground truth target. The error surface is not smooth, as shown in Fig. 3.15, instead it is usually comprised of many local minima. Figure 3.16 shows an algorithm that can get stuck in local minima.

To avoid such a situation, momentum value is used, which decides the direction of the gradient vector by preventing oscillations during optimisation of the weights. The learning momentum is usually kept higher in order to jump from local minima.

- **Epochs**. This is an integer number specifying the number of times the whole dataset is passed through the network while training.

The task of identifying the location of an object as well as predicting the class of an object in an image, is known as object detection. To detect an object in an image, an algorithm for CNN can be broadly divided into the following two types:

1. Classification based algorithms are used to generate several proposals in an image by sliding a fixed sized window at all the possible locations in the input image. The obtained proposals are fed to the image classifier to predict the class of an object. Usually, this algorithm is very slow because first various ROIs are selected in an image. Then these ROIs are classified using CNNs and predictions are made for every selected region. Mask R-CNN comes under the category of classification-based algorithms.

2. Regression based algorithms treat object detection as a regression problem by passing the whole image through CNN once and generating class probabilities. This algorithm predicts a continuous quantity of class probabilities as an output. YOLO network uses a regression-based algorithm. This algorithm is usually fast as the whole image is seen by the CNN in one run of the algorithm instead of dividing and passing several generated proposals of an input image.

3.7 Mask R-CNN

For the purpose of feature extraction, object detection and object localisation, multiple layers are combined to work together in a CNN such as a *region-based convolutional neural network* (R-CNN) by [102]. However, R-CNN involves computational expense as well as time-consuming training time. The R-CNN consists of three independent models including a CNN based feature extraction, an SVM classifier and finally a regression model for object location identification based on bounding boxes (also known as a *region of interest* (ROI)).

This problem of unifying three different models was solved and evolved into a new network known as Fast R-CNN [103]. Fast R-CNN improved the training time of R-CNN by introducing *region of interest pooling* (RoI Pool) layer. This layer shares the forward pass of an image and its subregions in a CNN. In this, the CNN features are obtained for each region using a corresponding region of a CNN 's feature map. Using max-pooling features are pooled for each image region. However, the region proposals were still generated by the selective search [104]. Hence, the training is still expensive.

Faster R-CNN removed this limitation of a generation of region proposals by a separate model and integrated the region proposal algorithm into the CNN model. Thus, Faster-R-CNN is a single, unified model composed of a *region-proposal network* (RPN) and Fast R-CNN with shared convolution feature layers. The Faster R-CNN reuses the same CNN feature which is calculated during the forward pass of the CNN in Fast R-CNN. So, Faster R-CNN generates region proposals using CNN features by adding a fully convolutional network on top of the CNN features, which is known as the Region Proposal Network. This network uses a sliding window over a CNN features map, which outputs bounding boxes and scores at each window.

Mask R-CNN [105] is an extension of Faster R-CNN to pixel-level instance segmentation (see Fig. 3.17). Mask R-CNN separated the classification and pixel-level mask prediction step and added a third branch to predict an instance level segmentation along with the other two branches of classification and localisation.

To predict a pixel level segmentation mask, a small fully-connected network is applied to each ROI. In Mask R-CNN, the features selected using RoI Pool layer were misaligned with the original image. So, the RoI Align layer is used for precise alignment of the mask with the original image.

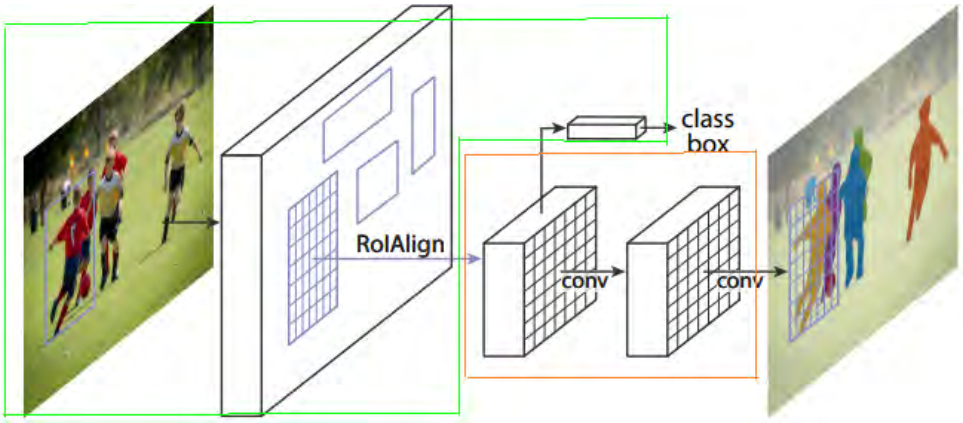

Figure 3.17: Mask R-CNN, source [105], where the green box encloses Faster R-CNN and the orange box encloses FCN.

The Mask R-CNN implementation uses a *residual neural network* (ResNet101) [106] and *feature pyramid network* [107] (FPN). A ResNet is a standard feature extractor which detects low-level features at early layers, and high-level features at later layers. The network accepts an image of 1024×1024 pixels. A smaller image is padded with zeroes to match with the expected image resolution. FPN is another improved feature extractor, a second pyramid that allows features at every level to have access to both lower- and higher-level features.

3.8 YOLOv2

YOLO is an object detection network which uses a single regression problem [108]. YOLO consists of 24 convolutional layers, which are followed by two fully connected layers (Fig. 3.18). YOLO has a limitation of one object rule which limits how close detected objects can be.

We have used *YOLO version 2*- (YOLOv2) [109] for our experiments in Chapter 5. YOLOv2 is better and faster than YOLO. YOLOv2 uses batch normalisation at all convolutional layers, thus eliminating the need of other regularisation techniques. The one less pooling layer in YOLOv2 produces higher resolution output from the network. YOLOv2 operates on an input image of 416×416 to have one single centre cell in a feature map. This is done because usually large objects occupy the centre of the image. Also, the fully connected layer is removed in YOLOv2. To predict bounding boxes, anchor boxes are used and for every anchor box, a class and objectness [110] are predicted. The anchor boxes help to remove the limitation of YOLO where spatially close objects are not detected. It is an improved version of

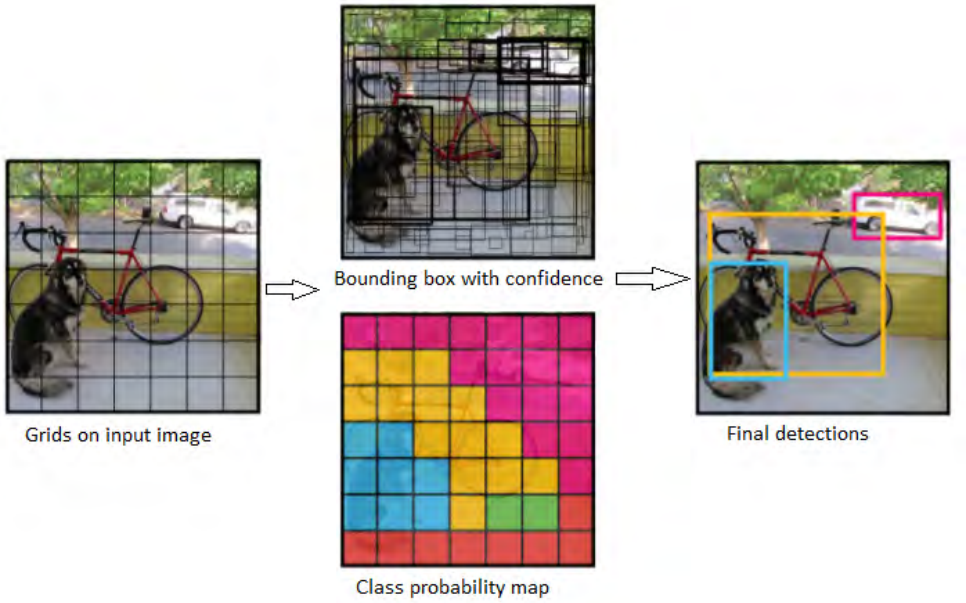

Figure 3.18: YOLO, source [108].

YOLO [108] as it generalises better over image size.

An input image of size $[n \times n]$ pixel is passed through a CNN and output is a vector of bounding boxes (b). The input image gets divided into a grid of $g \times g$ cells where, $g = \frac{n}{s^p}$ is computed from stride s and maxpool layers p.

For example, in YOLOv2 the input image size is $[n \times n] = [416 \times 416]$ pixels with $s = 2$ and $p = 5$ so, the grid cell size is 13.

These grid cells produce N bounding boxes along with their confidence scores. The next step in YOLO is to perform non-max suppression, which is a process of removing bounding boxes with low object probability and highest shared area.

The bounding box consists of five numeric predictions: confidence, x, y, width, and height where, the confidence score represents *intersection over union* `IoU` between predicted and ground truth box and x, y are coordinates centre of the box relative to the grid cell, width and height are relative to the input.

During testing the confidence score represents how likely and accurately the bounding box has the object. YOLO produces several bounding boxes per grid cell, but only one of them is responsible for the object being detected. So, the bounding box with the higher `IoU` is selected. The loss function in YOLO mainly comprises of classification, localisation and confidence loss.

3.9 Datasets

Authors of [66] state that "there is no uniform road damage dataset available openly, leading to the absence of a benchmark for road damage detection". Existing datasets on websites of the KITTI [46], EISATS [111] projects, or of Middlebury College [112] have been recorded in countries where distress on roads typically occurs rarely, and they are not benchmarks for road distress. Under these circumstances, we decided on the use of the following data and images from each dataset is shown in Fig. 3.19:

1. *CCSAD*. J. B. Hayet et al. [113] has introduced a dataset of *challenging sequences for autonomous driving* (CCSAD) that is exceptionally utilitarian to execute strategies for detection of road potholes in Mexico. The CCSAD dataset has been split into four parts, `Colonial Town Streets`, `Urban Streets`, `Avenues and Small Roads`, and `Tunnel Network` which accounts for 500 GB of data that incorporates calibrated and rectified pairs of stereo images, videos and meta-data.

The *CCSAD* `Urban Streets` dataset is an extensive collection of road potholes. The data has been acquired at 20 fps using two Basler Scout scA1300-32fm firewire greyscale cameras mounted on the roof of a car. The image resolution of `CCSAD` is 1096×822.

2. `DLR`. This dataset has been recorded while using the *integrated positioning system* (IPS) [114, 115], developed by the German Aerospace Centre (DLR), installed on a car. The collected dataset accounts for 288 GB with image dimensions as $1360 * 1024$.

3. `Japan`. This dataset comprises of $163,664$ road images of dimensions 600×600 collected in seven different cities of Japan [66]. The dataset contains $9,053$ damaged-road images and $15,435$ instances of damaged road surfaces such as (mainly) cracks and (rarely) potholes. Images are captured at an interval of one second under different weather and lighting conditions.[2]

4. `Sunny`. Authors of [116] provided a dataset of $48,913$ images of size $3,680 \times 2,760$ recorded using a GoPro camera, mounted inside a car on its windscreen. The camera was set to a 0.5 second time lapse mode and car was moving at an average speed of 40 km/h while scanning the road surface. The total data available is 2.70 GB.

5. `PNW`. PNW is an extensive video recorded on the *Pacific northwest highway* [117].

[2] In late 2018, these data were used in a competition, see `bdc2018.mycityreport.net/overview/`.

It shows the highway with patches of snow and water. We used $19,784$ extracted frames of dimension 1280×720 from this video.

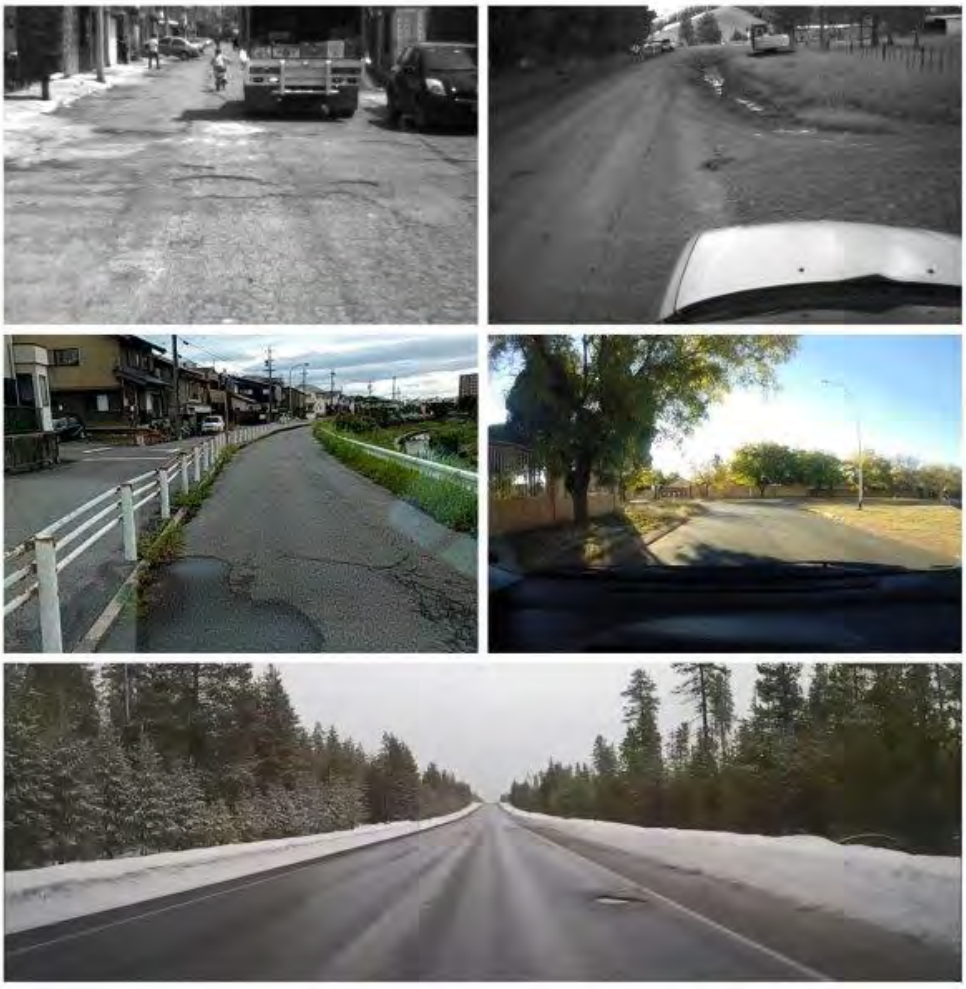

Figure 3.19: Top left - *CCSAD*, Top right - *DLR*, middle left - Japan, middle right- Sunny, bottom - PNW.

Chapter 4

Stereo-vision based methods

This chapter proposes two stereo-vision based novel techniques for pothole detection. The first section presents a single-frame stereo-vision based method and the second section presents a multi-frame fusion based method. The third section discusses the value of the proposed methods for road-surface distress analysis. The chapter ends with a brief summary. The two novel methods have been published in [118] in 2017 and in [119] in 2018.

4.1 Single-frame stereo-vision based method - *SV1*

Stereo cameras are considered to be cost-effective compared to other sensors. Their application involves effective and accurate disparity computation using disparity images calculated from a stereo frame (as discussed in Chapter 3).

This section presents a single-frame stereo vision based method; the presentation of the method and the experiments use the CCSAD dataset for illustration and verification. The CCSAD dataset was recorded in Mexico showing various examples of road surface distress [113]; (see Fig. 4.1 for examples of the used dataset). An example of calculated disparity maps for frames shown in Fig. 4.1 is shown in Fig. 4.5 by setting the maximum disparity value as 128.

The proposed method starts with deriving a disparity map from the left and right frames of a stereo pair. For example, on the CCSAD dataset, see a calculated disparity map in Fig. 4.2. Based on the shown disparity map, two columns are profiled at $x = 200$ (green) and $x = 400$ (red) and shown in the fused image in Fig. 4.2.

From the profiles, we see in Fig. 4.3 the distressed region around $y = 330$ shows a sudden change in disparity values, due to an unevenness on the road. To examine such changes, we differentiated the disparity map along the y-axis, as shown in

Figure 4.1: Intensity images captured by the left camera as available in the CCSAD dataset.

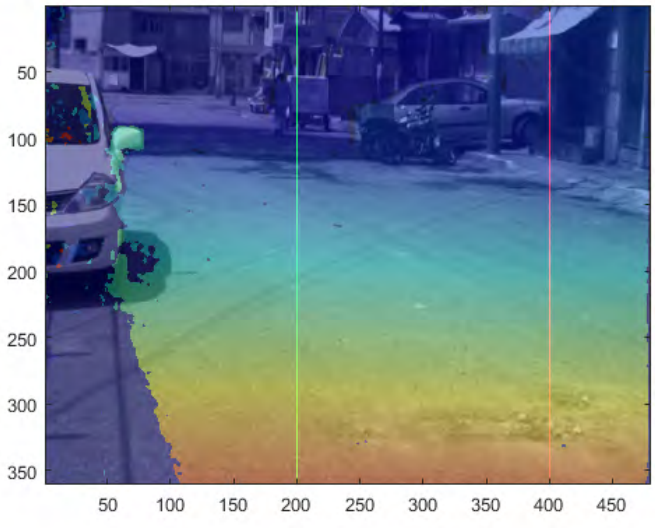

Figure 4.2: Calculated disparity map using the CCSAD left and right frames of the stereo frame illustrated in the upper left of Fig. 4.1.

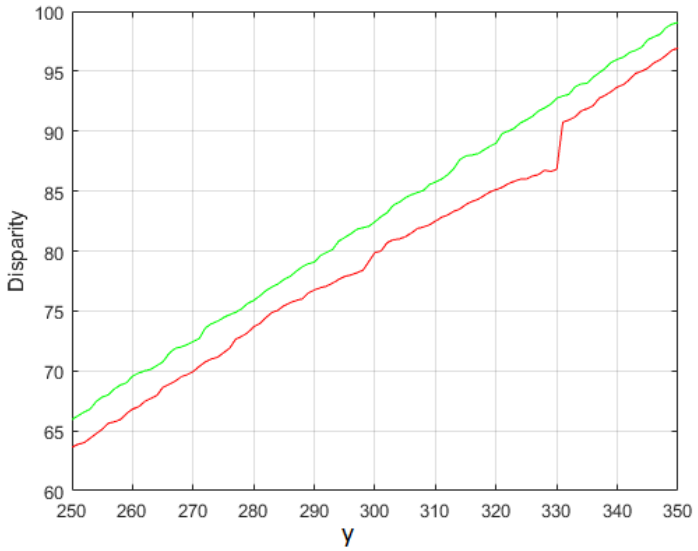

Figure 4.3: Distressed region showing an unevenness on the road.

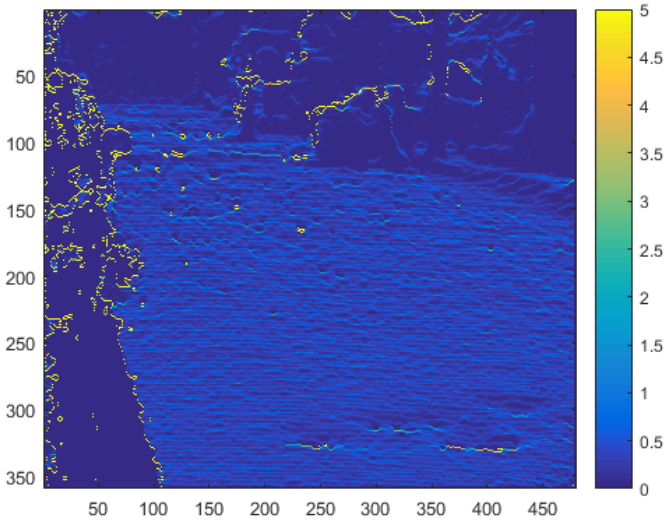

Figure 4.4: Disparity map differentiated along the y- axis.

Fig. 4.4. Such findings motivated us to detect discontinuities on road manifolds based on changes in the disparity map, without back-projecting disparities into 3D space. This is done to avoid the transformation from xyd to XYZ space which is nonlinear [120].

We propose a strategy that performs *road-plane modelling* directly in the image-disparity space, without back-projecting a disparity image into 3D space. The modelling is based on a RANSAC process that finds the *dominating plane* [121] for locating potholes being below surface level. In short, we address this dominating-plane method by *SV1* (i.e. stereo-vision method 1).

We also formulate an alternative implementation using a common y-disparity technique based on straight-line estimation in y-disparity space [122]. Both implementations are evaluated on the `CCSAD` urban dataset; our experiments indicate a better accuracy of our dominating-plane method in terms of pothole detection compared to the linear y-disparity technique.

4.1.1 Planar road surface approximation

This section details the two approaches briefly mentioned above of estimating the road manifold from disparity images using 3D planes in disparity space, or of line fitting in y disparity space.

3D planes in disparity space

Consider a plane $a_0X + a_1Y + a_2Z + a_3 = 0$ in 3D Euclidean space with plane coefficients $a_0, \ldots, a_3 \in \mathbb{R}$. A point (X, Y, Z) in 3D space is mapped onto an image pixel (x, y) following the pinhole projection model:

$$x = f_x \cdot \frac{X}{Z} + x_c, \quad y = f_y \cdot \frac{Y}{Z} + y_c \tag{4.1}$$

where (f_x, f_y) are the focal lengths, and (x_c, y_c) is the principal point.

Two calibrated and horizontally rectified pinhole cameras introduce a disparity space, where every pixel (x, y) in the (say) left image is mapped to $(x - d, y)$ in the right image via $d \in [0, d_{\text{max}})$, the disparity value bounded by d_{max}. The disparity-to-depth conversion follows

$$Z = f_x \cdot \frac{b}{d} \tag{4.2}$$

where b is the length of the baseline (connecting the focal points of the two cameras) in world units.

Planarity in 3D space is well preserved in the image-disparity space (x, y, d) by the related conversion. It can be verified by first substituting (4.1) into the plane equation, resulting in

$$a_0 \cdot \frac{Z}{f_x}(x - x_c) + a_1 \cdot \frac{Z}{f_y}(y - y_c) + a_2 Z + a_3 = 0 \tag{4.3}$$

and then (4.2) into (4.3) producing

$$a_0 \cdot \frac{x - x_c}{f_x} + a_1 \cdot \frac{y - y_c}{f_y} + a_2 + a_3 \frac{d}{b f_x} = 0 \tag{4.4}$$

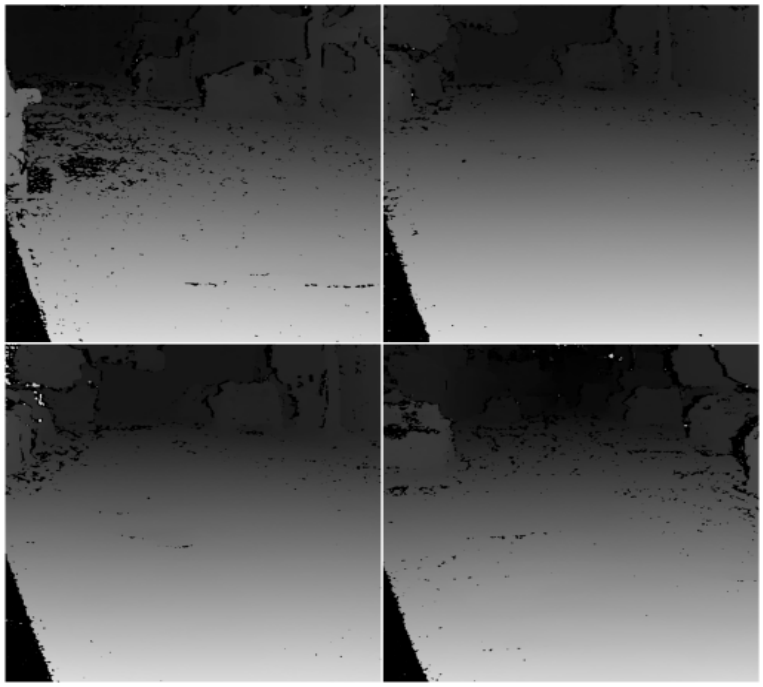

Figure 4.5: Disparity maps computed by OpenCV's SGBM stereo matcher for stereo frames illustrated in Fig. 4.1 using a grey-level key for visualising different disparities.

This equation can be rewritten as

$$a'_0 x + a'_1 y + a'_2 d + a'_3 = 0 \tag{4.5}$$

in terms of $a'_0 = (bf_y)a_0$, $a'_1 = (bf_x)a_1$, $a'_2 = f_y a_3$ and $a'_3 = (bf_x f_y)a_2 - (bf_y x_c + bf_x y_c)$, which also models a plane, but in the image-disparity space. [Note that a_2 and a_3 move basically into a'_3 and a'_2 from Eq. (4.4) to Eq. (4.5).]

Based on the described linearity, we find the road plane without any need of back-projecting a disparity map into the 3D Euclidean space by applying the following RANSAC procedure.

RANSAC-based plane fitting

A plane is not uniquely parametrized by (4.5), as any multiple of (a'_0, a'_1, a'_2, a'_3), in fact, defines the same plane. In this study, we adopt the normal-offset parametrization where a plane is uniquely defined by a unit 3-vector $\mathbf{n}$ and an offset $\delta \in \mathbb{R}_+$, where

$$\delta = \frac{|a'_3|}{\sqrt{{a'_0}^2 + {a'_1}^2 + {a'_2}^2}} \tag{4.6}$$

and

$$\mathbf{n} = \frac{\delta}{a'_3} \cdot (a'_0, a'_1, a'_2)^\top \tag{4.7}$$

A point $\mathbf{p} = (x, y, d)^\top$ is in a plane $(\mathbf{n}, \delta)$ if and only if

$$\mathbf{p}^\top \mathbf{n} - \delta = 0\,, \tag{4.8}$$

and the signed distance from an off-plane point $\mathbf{p}$ to the plane is defined by

$$\varepsilon(\mathbf{p}, \mathbf{n}, \delta) = \mathbf{p}^\top \mathbf{n} - \delta \tag{4.9}$$

If $\varepsilon > 1$ then the point is above the plane, with an up-vector defined by $\mathbf{n}$; $\varepsilon < 1$ means that the point is below the plane.

As point clouds converted from the real-world data are noisy, so we define a threshold $\varepsilon_{\text{max}} \in \mathbb{R}_+$ such that a point is considered to be an *inlier* with respect to a plane hypothesis $(\hat{\mathbf{n}}, \hat{\delta})$ if

$$|\varepsilon(\mathbf{p}, \hat{\mathbf{n}}, \hat{\delta})| \leq \varepsilon_{\text{max}} \tag{4.10}$$

According to (4.10), we deploy a RANSAC process. It first draws a minimum set

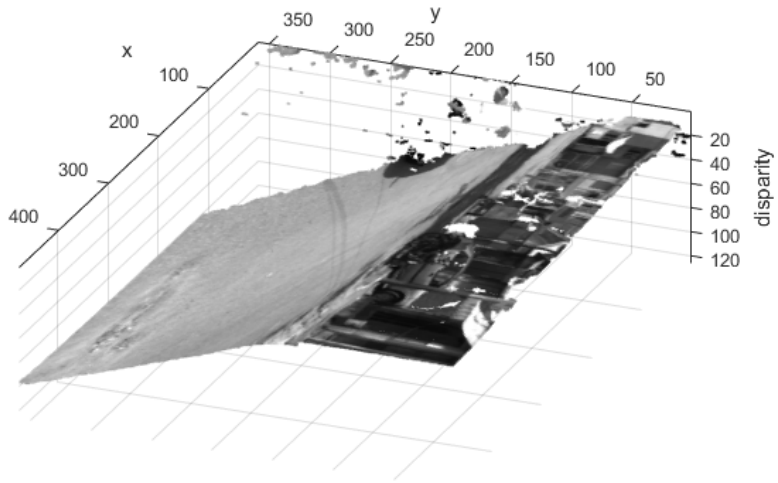

Figure 4.6: Rendering of a scene in image-disparity space.

of points from the whole image, based on which the best-fit-plane problem is solved. The hypothesis is verified by finding all the inliers from the image. A hypothesis,

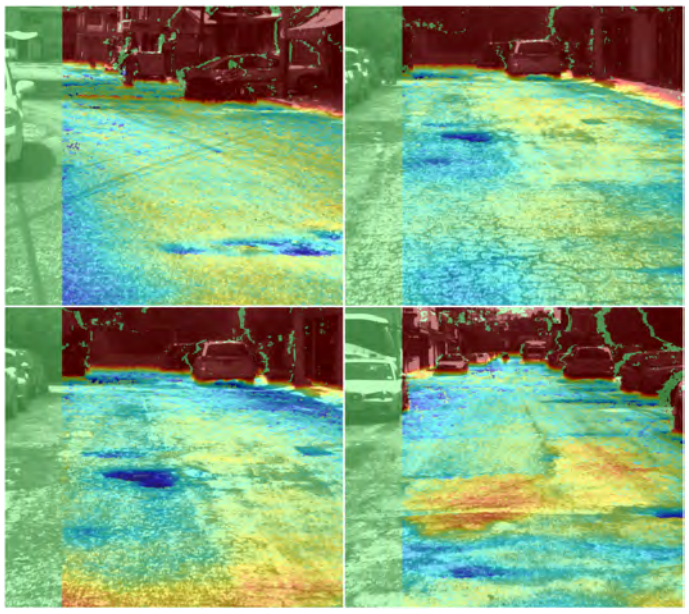

Figure 4.7: Signed distances to the best-fit plane. Note that several regions have negative distances, meaning that they are below the found plane.

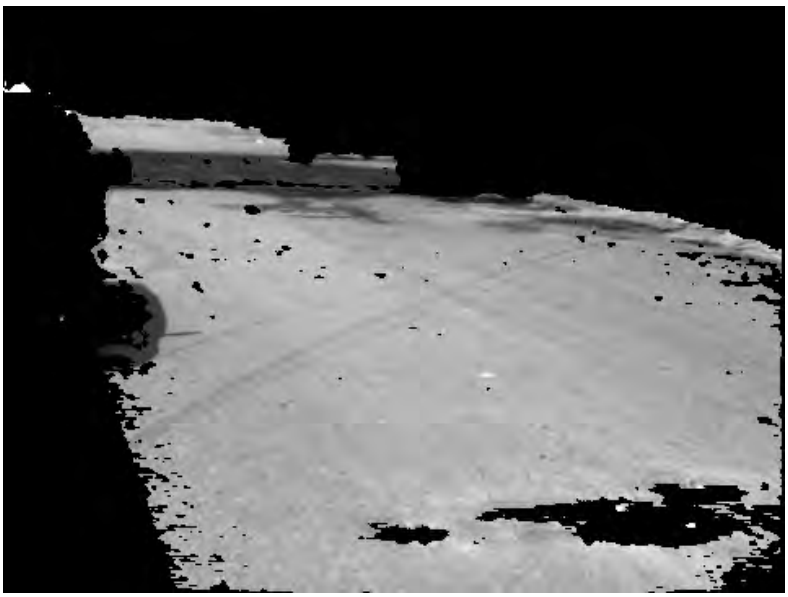

Figure 4.8: Road manifold supported by the inliers with ε_{max} set to 1.

supported by significantly many inliers (say, 50%) is considered to be a candidate model. The process is repeated for a predefined number of iterations. In the end, the candidate which is supported by the highest number of inliers wins the selection. A 3D representation of the disparity map in the image-disparity space is visualized by Fig. 4.6.

Following the described RANSAC process, a *dominating plane* is found. We further calculate the signed distance from each pixel to the plane and have it rendered in Fig. 4.7, where the blue colour represents the pixels being below the road surface. The inliers are considered to be the road manifold, as shown in Fig. 4.8.

4.1.2 Line fitting in y-disparity space

The RANSAC plane-fitting process can be reduced to a 2D line fitting problem under certain conditions. For y-disparities, readers are referred to the original source [122] (which used v for vertical coordinates), or a current application in [123].

A y-disparity map is computed by using an image D of a disparity map as follows:

$$V(y,d) = \text{card}\{x : 1 \leq x \leq N_{\text{cols}} \wedge D(x,y) = d\} \tag{4.11}$$

where N_{cols} is the width of the input images in pixels, and $d \in [0, d_{\max})$ is a disparity bounded by the maximum value $d_{\max}$. $V(y,d)$ gives the number of pixels sharing the same disparity d in the y-th row of the disparity map.

Using this, we implemented a y-*disparity*-based road manifold detection technique. The line found in the y-disparity space can be used to define a threshold for each row so we can do a row-wise threshold to find the road manifold.

Here we have a y-disparity space (on the left) constructed from the disparity map **D** on the right 4.9. Here the green cut $d = f(y)$ in the y-disparity space indicates the disparity of road manifold for each row. Then the green cut is expanded in the image space to obtain the following mask M 4.10.

Here the values in each row are filled by the disparity of the green curve at the same row in the y-disparity space, i.e. $M(x,y) = f(y)$. Then we compared the

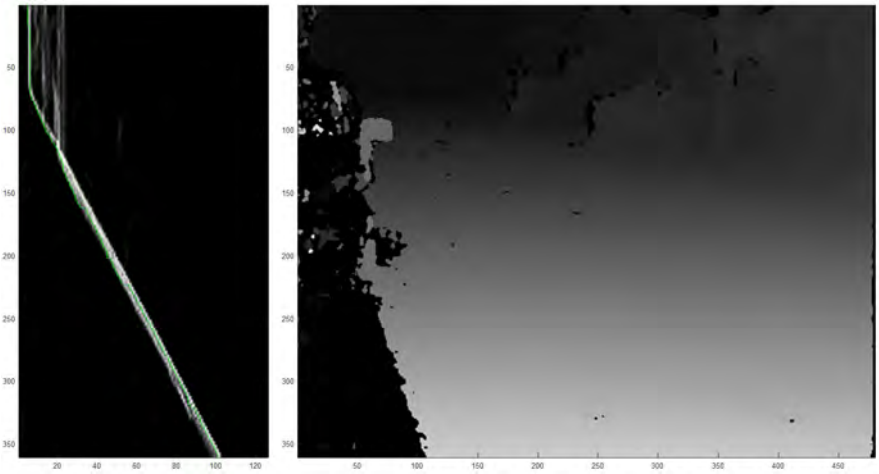

Figure 4.9: y-disparity map computed from the disparity map on the right.

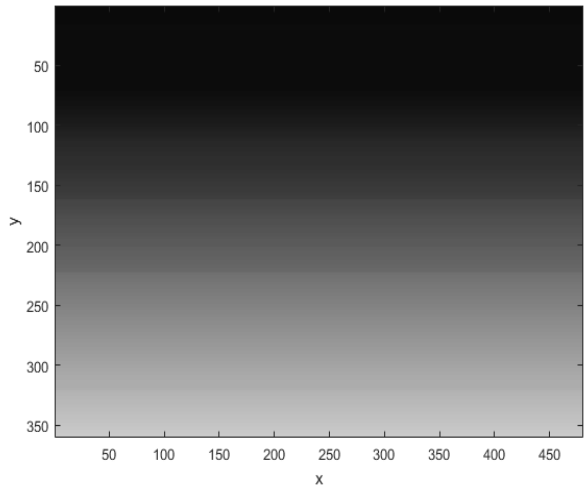

Figure 4.10: Green cut shown in Fig. 4.9 is expanded to obtain this mask M.

disparity map with the resulting mask; if the difference is less than a tolerance, say 5 pixels, we consider the pixel on the road manifold. Such a thresholding process produces the binary image R, see Fig. 4.11. Fig. 4.12 is obtained after applying the mask to it.

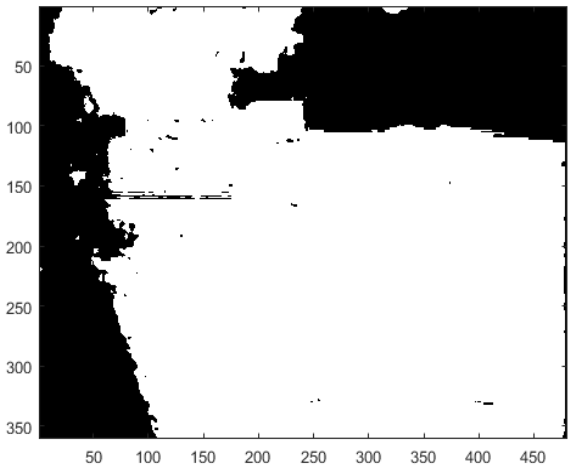

Figure 4.11: Obtained binary image R after thresholding.

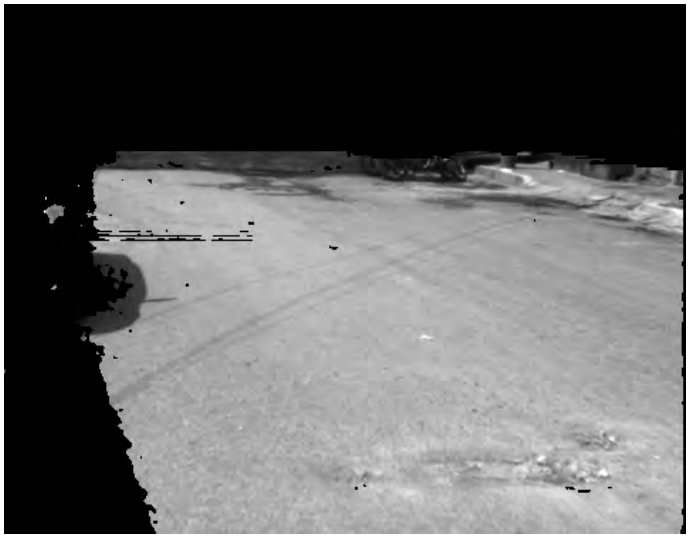

Figure 4.12: Obtained image after applying mask. Note the pixels from y= 0 to 100 are also masked out as they are too far.

Now, as the potholes are lower than the road manifold, we find them by identifying those pixels below the surface of the road. To identify these pixels, we cal-

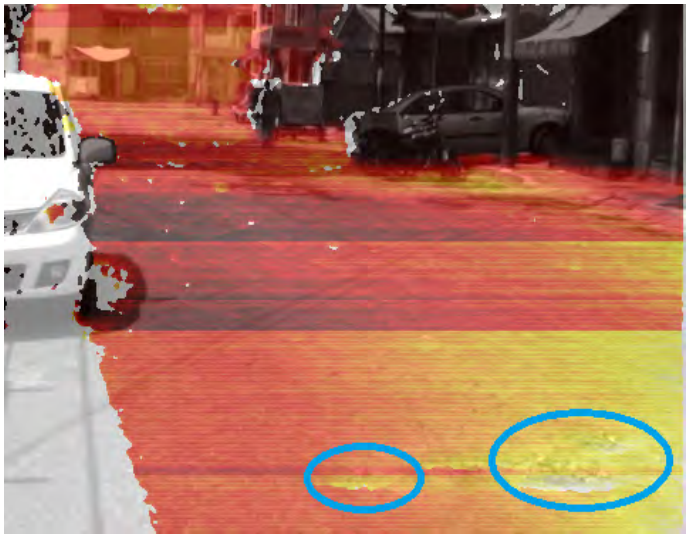

Figure 4.13: The brighter pixels indicate disparities that significantly smaller than the road manifold's disparities.

culated the difference $M - D$ and have it visualised with the intensity image 4.13. Here the brighter pixels indicate disparities that are significantly smaller than the road manifold's disparities. We marked two regions of interest enclosed in a circle. The rationale is based on the fact that the surface of a pothole is slightly farther than the surface of road in the same row. (We may find a pencil and do some drawing to verify this.)

This encouraged us to implement the same following the RANSAC way to improve the found curve, which is not smooth. That possibly causes the awkward stairwell looking artefact in the Fig. 4.13.

Therefore, a RANSAC-based line-fitting process is used to implement a variation of the general sparse 2D line-fitting technique for y-disparity maps. As values in the map define a density distribution, we make direct use of Eq. 4.11 to weight the fitness for each line hypothesis that takes (y, d) as an inlier. The process is closely related to the well-known Hough transform method [55]. Figure 4.14 visualises a computed y-disparity map, and the best-fit model found in 100 iterations. The estimated line model $d = f(y)$ is then applied to the disparity map D. A pixel (x, y) is considered to show a 3D point on the road if $|D(x, y) - f(y)|$ is smaller than a predefined threshold, say 1 pixel. An epsilon map defined by the signed difference this way is shown in Fig. 4.15; for detected road pixels (see Fig. 4.16). Compare with Fig. 4.8.

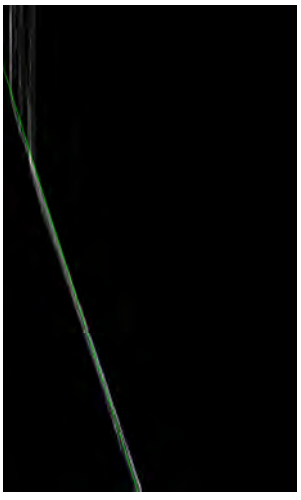

Figure 4.14: y-disparity map with a line fit for the lower envelope.

Figure 4.15: Obtained epsilon map using best-line fit in y-disparity space.

Figure 4.16: Obtained road manifold using the alternative y-disparity approach.

4.1.3 Pothole detection using *SV1*

The extracted road manifold is used to guide the pothole detection process. Pixels lower than the detected plane, for more than 1 unit, are considered to be pothole candidates and worthy of further investigation.

To remove noise from the pothole candidates, we apply a threshold operation to remove shallow pothole candidates, which is then followed by a morphological opening operation [124, 125] to fractional pieces in the binary mask.

Observing that a pothole pixel location (x, y) usually shares similar gradients with its neighbourhood in the disparity image, we perform an 8-adjacency connectedness analysis [124, 125] to group pixels in the disparity image masked by the filtered pothole candidates. We remove connected small regions to suppress the detection of minor distress. Some regions larger than a reasonable size are also removed if they are edges of a curved road surface.

The proposed pothole-detection methods, using either 3D plane estimation or y-disparities, have been implemented in MATLAB version R2017b (using the built-in image processing and computer vision toolbox).

We choose sequences from the `Urban Streets sequence 1` category as it presents an extensive collection of road distresses. It is remarkable how consistently the potholes are being identified in the given challenging CCSAD. To evaluate comparatively the performance of both techniques, we use `precision` and `recall`.

Table 4.1 shows the scores that have been calculated pixel by pixel compared to the ground truth. Since the dataset does not come with pothole ground truth, it was established by two independent markers, with each marker's labelling shown in a different colour in Fig. 4.17.

The comparison between the performance of the 3D-plane-estimation method with the y-disparity line fit shows the capability of the 3D-plane technique of pro-

Table 4.1: Classification measures in percentage.

Measure	Plane fit	Line fit
$Precision$	62.0	27.8
$Recall$	25.8	8.2
TNR	99.6	99.7
$Accuracy$	98.2	97.9

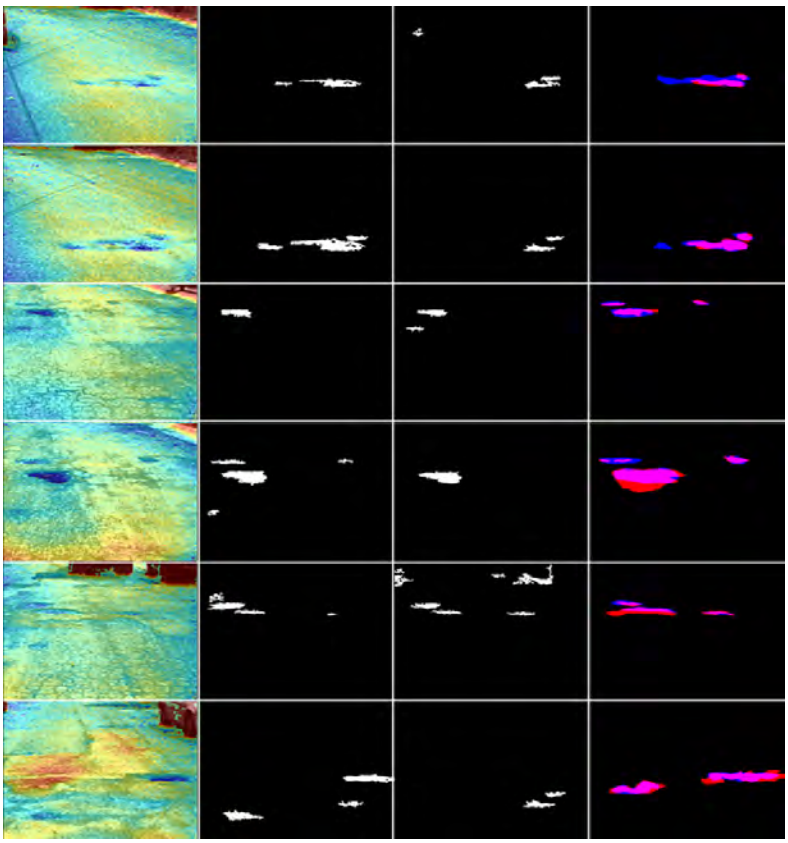

Figure 4.17: *From left to right*: Epsilon map, potholes using plane fitting, potholes using y-disparity line fitting, manually labelled ground truth. The ground truth was established independently by two markers, with each marker's labelling, or differences between labelling, shown in different colours.

viding good performances; a detailed analysis of results shows an overall accuracy of 98% with 62% `precision` and 25% `recall` in detecting potholes using the proposed 3D-plane technique.

Experiments illustrated by Fig. 4.17 are for randomly selected 6 stereo images of `CCSAD`. Using the 3D-plane technique, it remains difficult to distinguish between major cracks and potholes. Therefore, more processing is needed if such a decision is required.

4.1.4 Summary

Section 4.1 presented two variants of a pothole detection method by using a computationally efficient and robust disparity calculation algorithm.

To obtain the road manifold, we either fit a 3D plane in the image-disparity space using RANSAC, or we use line fitting for the lower envelope of y-disparities.

Potholes have been modelled considering the fact that they would be always lower than the estimated road surface. Using the fitted 3D plane for the road manifold, potholes have been identified with reasonable accuracy, and with less accuracy when using y-disparity for the line fit and road manifold estimation. A reason might be that the cameras, when recording the `CCSAD` data set, have been slightly tilted about the car's forward coordinate axis. Because this cannot be avoided in general, the 3D plane fit appears to be the more appropriate way to go. Because the road surface is not perfectly angled to the camera's y-axis, the y-disparity model assumes the slope of the road manifold doesn't change w.r.t. the y-axis. The sample images we have been working on so far; however, do not follow that principle. A clear evidence can be found in the y-disparity map, which shows a dispersed distribution of disparities as visualised by 4.18.

The shift of road slope can also be found in the per-row profiles of the disparity map, see Fig. 4.19. Obtained results are altogether promising and the information obtained using the 3D-plane technique can be used to alert drivers as well as for a

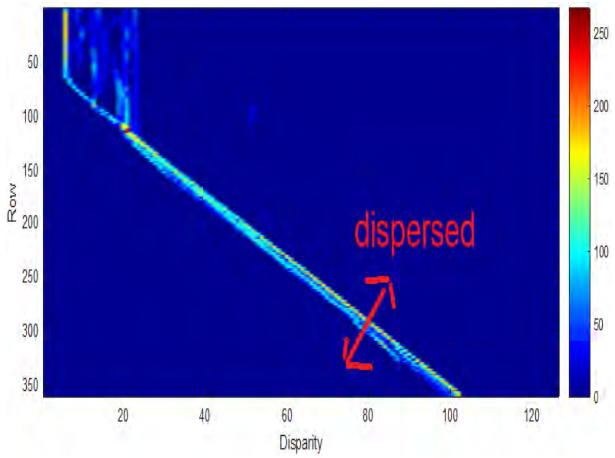

Figure 4.18: Visualisation of dispersed distribution of disparities.

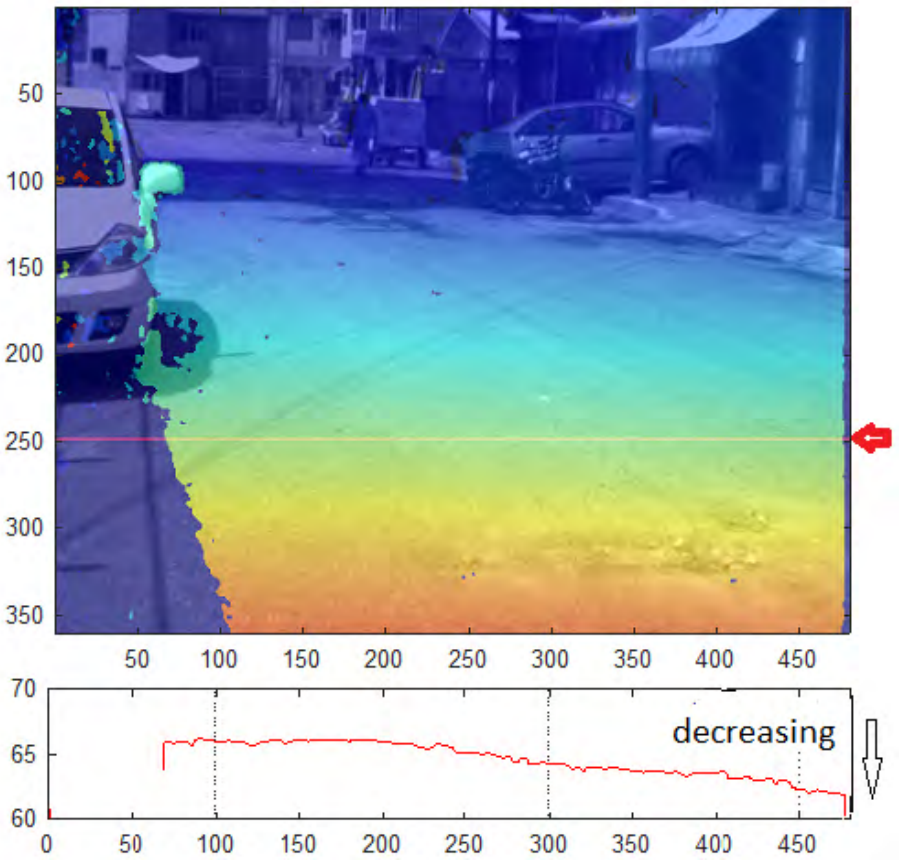

Figure 4.19: The profile shows the disparity values along row 250. The disparity drops as the road is not perfectly angled.

preliminary maintenance report for roads. The proposed 3D-plane technique has a better performance when combined with multiple frames and tracking of pothole features. The next Section 4.2 experiments with multiple frame accumulation and tracking features.

4.2 Multi-frame fusion based method - *SV2*

The novelty of the proposed method is credited by the accumulation of multiple-frame 3D reconstructions which are properly aligned to a road-centred coordinate system. A 3D plane-fitting technique is first carried out to approximate the road manifold at the beginning of accumulation, followed by the construction of a digital elevation model (for the road being analysed) from multiple frames that are integrated using a stereo visual odometry algorithm. Potholes are detected as "valleys" from the built digital elevation model by means of connectedness analysis.

In this section, we describe a novel method to build a digital elevation model of the road from multiple stereo frames, by extending the approach described in the previous section. We address this multi-frame fusion based method using *SV2*.

4.2.1 Digital elevation model

It is computationally inefficient to process the complete 3D model of an environment. Therefore, we model the geometry of a scene using the *digital elevation model* (DEM) index representation, a regular grid of squares representing heights to a zero-plane.

Given G a set of 3D points, the construction of a DEM M is as follows. For each point $(X, Y, Z) \in G$ within a defined range of interest $X \in [X_{\text{min}}, X_{\text{max}}]$ and $Z \in [Z_{\text{min}}, Z_{\text{max}}]$, a corresponding cell $(i, j) \in \mathbb{Z}^2$ is found by

$$i = \left[\frac{Z - Z_{\text{min}}}{W}\right], \quad j = \left[\frac{X - X_{\text{min}}}{W}\right] \tag{4.12}$$

where W is a chosen size of grid such that every cell spans an area of $W \times W$ in the space, and $[x]$ denotes the nearest integer to real number x.

The value of $M(i, j)$ is decided by all the points assigned to cell (i, j). Thanks to the known structural information of the road, it allows the elevation to be properly defined with respect to the road manifold. In this study, we use the averaged signed distances of cell points to the road plane to build $M(i, j)$.

We also deployed a *principal component analysis* (PCA) technique to find a rigid transformation that normalises point cloud G in a way that the z-axis aligns to the primal axis of the road and y-coordinates of each point being the signed distance to the road plane (i.e. the y-axis is parallel to $\mathbf{n}$, the normal vector of the plane) after the

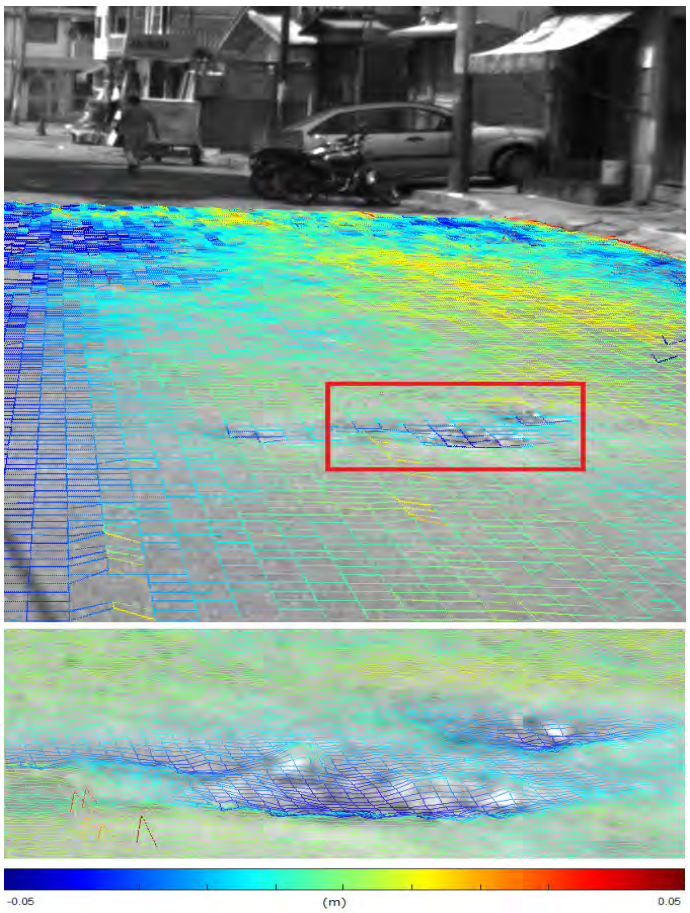

Figure 4.20: Visualisation of a digital elevation model that shows elevation profile of the road surface in a cell resolution of 10×10 cm^2 (top) and a closer look on a pothole in an enhanced resolution of 1×1 cm^2 (bottom).

transformation is applied to G. We define the transformed space as the *road-centred space* (RCS), on which the DEM is constructed. The adoption of RCS allows the road to be modelled in a view-independent manner, as the transformation is intrinsic to the geometry of the road being analysed.

4.2.2 Visual odometry

To accumulate 3D data measured in different frames, their poses have to be recovered with respect to the reference coordinate system. In this study, we use *visual odometry* (VO), a technique to recover the camera's egomotion[1] from a video sequence, to achieve this goal. The state-of-the-art implementations use either direct intensity matching or feature descriptor matching to establish inter-frame correspondences [93]. In this study, we implemented a 3-stage hybrid model as described in this section.

In the first stage, some key points are detected from a frame and tracked as a new frame arrives, by means of the *Kanade–Lucas–Tomasi* (KLT) algorithm. The tracking yields a set of the image correspondences $(x, y) \rightarrow (x', y')$ where (x, y) denotes the location of a key point in the last frame, and (x', y') is its tracked location in the new frame. Given the triangulated coordinates of (x, y) in the last frame, a set of 3D-to-2D mapping $(X, Y, Z) \rightarrow (x', y')$ is acquired.

Given such a mapping, an initial pose $(\mathbf{R}, \mathbf{t})$ consisting of $\mathbf{R} \in SO(3)$, a rotation matrix, and $\mathbf{t} \in \mathbb{R}^3$ a translation vector, is computed using an efficient *perspective-from-n-point* (PnP) algorithm [126]. To enhance the robustness, we deploy a RANSAC strategy similar to the one described in Section 4.1.1 on the established correspondences to exclude outliers.

In the second stage, a nonlinear refinement is carried out to improve $(\mathbf{R}, \mathbf{t})$. We further extract feature descriptors for the detected key points. A descriptor matching process is then performed in the feature space. The initial pose estimate is used to filter outliers. We use the estimate and camera intrinsic to derive an *essential matrix*, based on which the Sampson distance is calculated for each pair of matched key points (see [127] for details). Subject to the filtered correspondences, the reprojection error is minimized in the sum-of-squares form:

$$\varphi_{\text{RPE}}(\mathbf{R}, \mathbf{t}) = \sum_{i}^{\mathcal{P}} \left\| \pi \left(\mathbf{R} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + \mathbf{t} \right) - \begin{bmatrix} x'_i \\ y'_i \end{bmatrix} \right\|^2 \tag{4.13}$$

where $\mathcal{P}$ denotes a set of sparse 3D-to-2D correspondences and $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ symbolises the pinhole projection function defined by (4.1). The function is minimised

[1] ego, Latin, means self. Egomotion, therefore, means self-motion of the camera.

using the *Levenberg-Marquardt* (LM) method [128]. In this work, we adopt SURF feature detector and descriptor extractor [44] for the second stage.

In the third stage, we fine-tune the optimised pose estimate following a photometric error minimisation process. It warps I_L, the left image of last frame, to the current frame using a pose hypothesis being tuned, and iteratively reduces the sum-of-squares difference in intensities, between the warped image and I'_L the left image observed in the current frame. In particular, the objective function is defined as

$$\varphi_{\text{INT}}(\mathbf{R}, \mathbf{t}) = \sum_{i}^{\mathcal{Q}} \left(I'_L(\pi \left(\mathbf{R} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + \mathbf{t} \right)) - I_L(x_i, y_i) \right)^2 \tag{4.14}$$

where $\mathcal{Q}$ is a set of pixels with valid depth data. Note $\mathcal{Q}$ has many more elements than $\mathcal{P}$ thus useful to further improve the estimation. We use LM algorithm again to approach a local- minima of (4.14), starting with the pose previously minimised subject to (4.13).

4.2.3 Weight assignment and multi-frame fusion

To improve the robustness of the DEM construction process, weighted averaging is adopted. We derive the weight for each data point by evaluating the quality of its disparity. In this study, we adopt a block-wise correlation-based evaluation on a disparity map solved by a stereo matcher.

The evaluation first reconstructs the left image from the right image I_R and a computed disparity map D. Let I'_L be the reconstructed image, it follows:

$$I'_L(x, y) = I_R(x - D(x, y), y) \tag{4.15}$$

Given the observed right image I_L, the block-wise correlation is defined as

$$C(x, y) = \sum_{p \in A(x,y)} \frac{(I_L(p) - \mu_{xy}) \left(I'_L(p) - \mu'_{xy}\right)}{\sigma_{xy}\sigma'_{xy}} \tag{4.16}$$

where $A(x, y)$ denotes neighbours centring at pixel (x, y), μ_{xy} and σ_{xy} are local mean and standard deviation calculated from $A(x, y)$ in image I_L, respectively, and μ'_{xy} and σ'_{xy} are those calculated from the reconstructed image I'_L.

For a good disparity estimate in (x, y), the correlation $C(x, y)$ will be close to 1, while an inaccurate estimate will lead to low coefficient as low as -1. We use normalised indicator $W(x, y) = (C(x, y) + 1)/2$ to weight each point during the accumulation process.

For multi-frame integration, we first solve camera poses with respect to a reference frame using the VO technique described in Section 4.2.2. After point clouds from different frames are aligned to the reference frame, we further transform them to RCS, using the rigid transformation solved by means of PCA.

In Fig. 4.21 accumulations over 1, 2, 5, 10, and 20 frames of a tested sequence are rendered. As can be seen, as depth data from more frames are integrated, the resulted DEM becomes denser and presents fewer missing cells. Compare to the single-frame approach described in Section 4.1, the multi-frame DEM approach not only models the road manifold in a more reliable way, but also provides more accurate geometric measures such as depth and size of each pothole.

4.2.4 Pothole detection using $SV2$

This section describes the process to identify defects on the road surface from grids of 2.5D DEM introduced in the previous section.

Potholes are indicated by those major valleys present in the height map. Due to occlusion, these valleys usually come with missing cells. As can be seen in Fig. 4.21, the edge closer to the camera of each deep road distress is occluded and cannot be measured despite the multi-frame accumulation technique. Hole-filling is therefore required before locating these valleys. In this study, we deploy bilinear interpolation to decide the height of a missing cell from its neighbours.

After filling the holes in the height map, we locate local minima by comparing each cell to its 8-connected neighbourhood. From each of these extrema, a region growing search is performed. The search floods to a cell's neighbours that are either lower, equal or slightly higher (within a tolerable range) than the cell. If the growth of a region reaches another region grown from a different starting cell, two regions are merged to form a larger valley. Such a process is repeated until all the extrema have been grown. Note a Gaussian filter is applied prior to the search process to accelerate it by suppressing non-significant valleys.

Following the region growing search, a size check is carried out. Regions that are either under- or over-sized are rejected. Those under-sized regions are usually

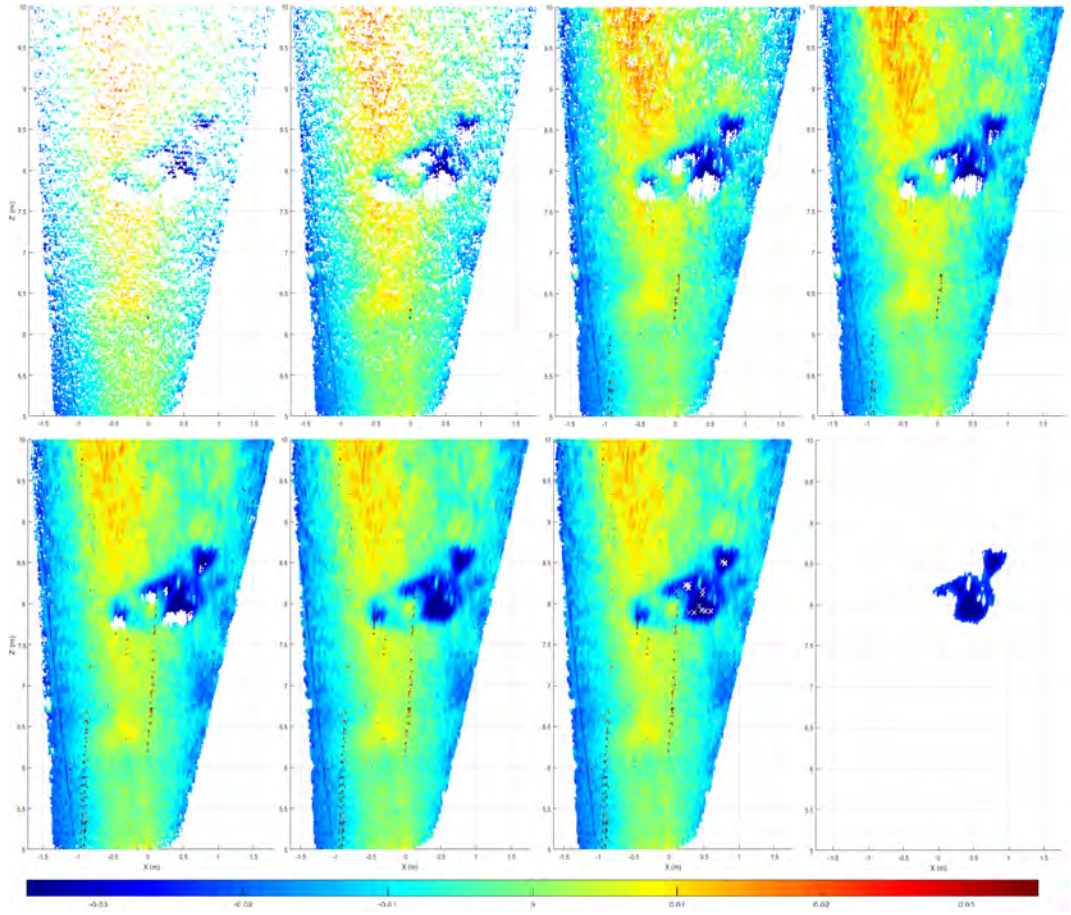

Figure 4.21: Here the vertical axis is z(meters) ranges from 5 to 10 and the horizontal bar ranges from -1.5 to $+1.5$. These are the road surface profiles. Top row, from left to right: profile from 1, 2, 5, and 10 subsequent frames. Bottom row, from left to right: profile from 20 accumulated frames before and after hole filling, identified local minima (as annotated by white crosses) and extracted valleys.

Figure 4.22: Outline of potholes identified from a multi-frame DEM, after projecting the valley cells into image domain and triangulating the vertices using alpha shapes.

small road distresses caused by unevenness, and over-sized regions are structures of the road drainage system on the sides. We also remove shallow regions that are not causing safety issues. In the end of this stage, potholes are identified.

To visualise the detected potholes in an image, we project the vertices of valley cells into the image domain. These vertices are then triangulated using 2D *alpha shapes* [129] to form polygons (either convex or non-convex). Figure. 4.22 shows the outline of a detected pothole.

The tested implementation accumulates frames in captured one second. The working volume is set to 10 metres wide and ranges from 5 to 15 metres in front of the vehicle.

The multi-frame DEM approach is compared with a single-frame plane based method. The detected potholes are shown in Fig. 4.23.

The proposed method is able to correctly identify all 14 potholes in the tested frames. These pairs are of identical size for calculating disparities among left and right images in the i-th (left image) and j-th (right image), respectively. In the best case, 92.5% of pixels from detected potholes are also annotated by the labellers, and more than 50% of labelled pixels are correctly detected. It also achieves an overall

Table 4.2: Evaluation of the proposed *SV2* and *SV1* approaches described in Section 4.1 and Section 4.2 in percentages.

Frame	Precision		Recall		Accuracy	
	SV1	*SV2*	*SV1*	*SV2*	*SV1*	*SV2*
0078	**67.4**	65.2	46.6	**53.2**	99.4	**99.4**
0093	57.3	**64.0**	**69.2**	61.4	99.1	**99.2**
0278	**78.2**	63.1	41.0	**52.9**	**99.6**	99.5
0304	76.5	**92.5**	**72.8**	57.6	99.4	**99.5**
0547	37.1	**81.5**	24.0	**34.7**	**98.8**	99.2
0935	24.2	**69.7**	45.1	**49.1**	**98.9**	99.6
Overall	45.8	**67.4**	45.8	**51.2**	99.1	**99.5**

accuracy of 99.5%. Compared to the single-frame approach, the multi-frame DEM shows improvements of 32.1% and 10.6% in terms of `precision` and `recall` rate. The results are listed in Table 4.2.

Figure 4.23: Tested frames and detected potholes. The potholes found using *SV1* and *SV2* approaches are marked in red and green respectively, while the manually labelled ground truths are marked in blue.

4.3 Summary

In Section 4.2, we presented a novel digital road profiling tool using a multi-frame depth data integration technique for DEM. The construction of DEM is performed in the view-independent road-centred coordinate system derived from the geometry of the road itself, and depth data from multiple frames are registered to the frame using a three-stage VO technique. The proposed approach can be applied to monocular vision, inertial-monocular vision [130], and monocular-LiDAR systems [131], and have great potential in road surface digitisation for a number of real-world applications [132].

Chapter 5

Deep-learning based methods

In this chapter, pothole detection has been achieved by using state-of-the-art deep learning based methods. Obtained results can be used for determining primary maintenance (fixing of potholes) or can be embedded into an application to alert drivers. The first method identifies potholes at the pixel level and produces quite an accurate mask which can be further used for analysis or annotation. The second method facilitates pothole identification using video data and the developed model has the potential for real-time scenarios. Conducted experiments aim to assess the performance of two state of the art object detection networks to detect potholes in our limited computational environment.

5.1 Computing setup

The assumption of machine learning algorithms that source and target data must be of the same feature space and distribution is not valid in real-world applications. Authors of [133] report that data under different distribution and feature space benefit from transfer learning techniques. An objective of this study is to implement and test CNN object detectors, to see how accurately a CNN trained on general source data can perform on a specific target task. This task of pothole detection is also coupled with varying illumination and weather conditions. These objectives altogether determined the selection criteria for the CNN and datasets. The available hardware and software setup influence the performance of a CNN, (see Table. 5.1 for environment setup) where *LM1* is used for transfer learning with Mask R-CNN and *LM2* is used for transfer learning with YOLOv2. As the environment setups for both networks are different, we chose not to use execution time as an evaluation metric. Execution time metric requires standardisation and the same environment

Table 5.1: Computing information for *LM1* and *LM2*.

Network	GPU	RAM	Framework	Machine
LM1	GTX 1080	32 GB	TensorFlow, Keras, Python, OpenCV	HP Z440 E5-1680v4 (local workstation)
LM2	Tesla K80	12 GB	C, CUDA, OpenCV, Python	Google Colab (free cloud service)

setup to produce one exact number, which is not possible in our case. We do report about training time for both of the techniques.

5.1.1 Ground truth annotation

Previous studies have demonstrated that "there is no uniform road damage dataset available openly, leading to the absence of a benchmark for road damage detection" [66]. Manual labelling of irregular potholes for marking ground truth is a laborious and tedious task. For pixel-wise annotation in an image, it takes around $30-50$ seconds to annotate a smallest pothole depending upon the choice of tool and network. Annotating a pothole using a bounding box is difficult as a pothole can be of any shape: circular, longitudinal or multiple potholes adjacent to each other (see Fig. 5.1).

Furthermore, the potholes marked in purple colours can be perceived as one big pothole or several smaller potholes. For instance, see Fig. 5.2, two potholes in the left image, six potholes in the right image (both images are the same), other potholes are not marked here, but considered in experiments. We put our best efforts into tackling the above mentioned two issues in this study.

5.1.2 Training and testing datasets

The dataset is the essential part when training a CNN. A well architectured CNN can achieve reasonably good results only when provided with a proper training and testing dataset. For a training dataset, we used a collection of 247 images, where 50 of the `CCSAD's urban sequence` 1, 100 of `DLR`, 48 `Japan` and 49 of `Sunny` dataset. The validation dataset comprised of 50 frames are also selected using the same datasets. We excluded 6 frames from `CCSAD's urban streets sequence`

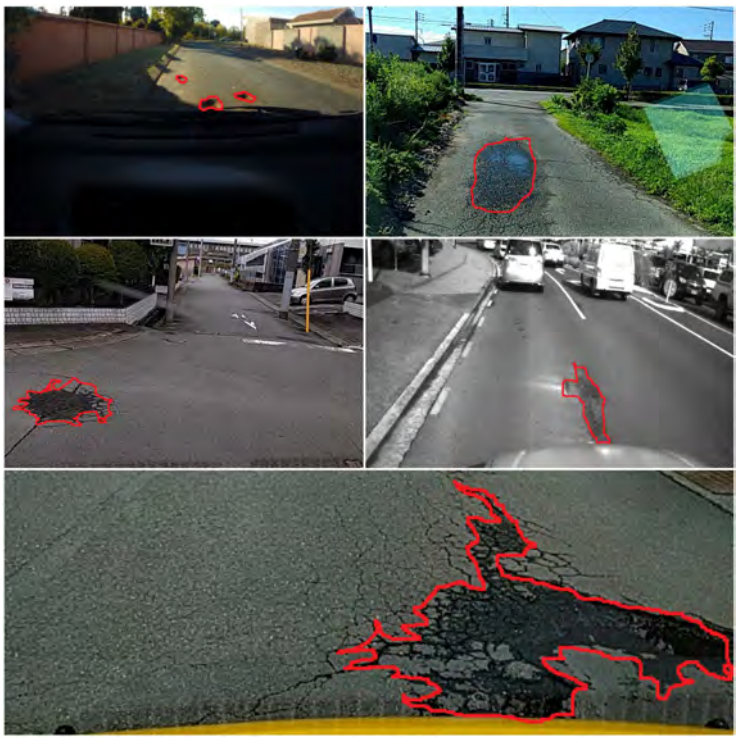

Figure 5.1: Images from `sunny` (first row first image), `Japan` (first row second image, second row first image and last row) and `DLR` (second row second image) showing the complex polynomial shape of potholes such as irregular circular, longitudinal.

1 for comparison purpose with *SV2*. The frames for training, validation and testing datasets are chosen on the basis of challenging scenarios such as shadow, varying illumination and weather conditions present in the scene. Data augmentation techniques are used to compensate for less training data. The test dataset is comprised of 50 images from `CCSAD's sequence 2` and a weather challenging `PNW` dataset. The weights trained on the *COCO* [134] dataset are used as the base model and these weights are adapted to identify potholes.

5.2 Using transfer learning with Mask R-CNN - *LM1*

To identify potholes at the pixel level, we use transfer learning with Mask R-CNN (*LM1*). To preserve the aspect ratio of uniform size $1,024 \times 1,024$, zero padding is added to the top and bottom of an image, as shown in Fig. 5.3, top left. It is done to support the originality of different datasets used in our experiments, as resizing the original image would shrink or expand the original pothole. Accordingly, this would require changes in ground truth while labelling different frames from datasets. The working of our *LM1* model is composed of a three-stage framework. The first stage scans the whole image for generating proposals, the second stage classifies the proposals, creates bounding boxes, and the third stage produces masks of an object. The detailed working is below:

Stage 1: Region proposal network. The backbone of *LM1* model uses a *Residual neural network* (ResNet101) [106] and *Feature pyramid network* (FPN) [77]. A ResNet is a standard feature extractor which detects low-level features at first layers, and high-level features at later layers. The FPN feeds the ResNet with extracted feature maps. *Region proposal network* (RPN) is a lightweight neural network that scans over the backbone's feature map. A sliding window is used to generate anchors that are typically boxes distributed over the image area. Its convolutional nature handles

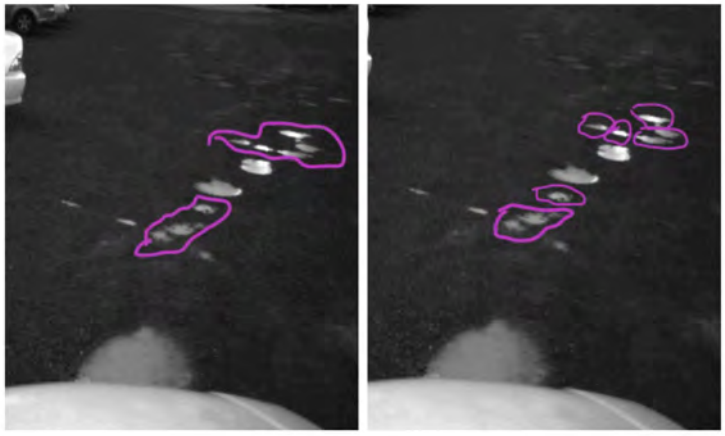

Figure 5.2: Annotation challenge while labelling ground truth (Images from `DLR` dataset).

the sliding window operation, and this is very fast on a GPU. The output of the RPN is an anchor class (see Fig. 5.3, top row, second image) at different scales and a bounding-box refinement.

We fine-tune the RPN end-to-end for a region proposals, initialised by a pre-trained CNN image classifier and IoU > 0.7 and < 0.3 define positive or negative samples, respectively. A small $n \times n$ window slides over the convolved feature map of the entire image. The anchor is produced to predict the multiple regions, at each sliding position. We used 256 anchors per image for RPN training.

Stage 2: Refined bounding boxes. Bounding boxes are mapped precisely to the regions of an image using an improved RoIAllign layer of the network for pixel-level segmentation. It removes the harsh quantisation of the RoIPool layer to encapsulate the extracted features with the input correctly. This stage accepts the refined anchors from the RPN and classifies the anchors, as shown in Fig. 5.3, top row, third image. A refined bounding box with a final detection is shown in Fig. 5.3, bottom row, first image. It trains the object detection model by using the proposals obtained by RPN.

Stage 3: Instance masks. The mask branch is a CNN that accepts positive regions

(a) Padded image (b) Predictions of RPN (c) Background with dotted, and pothole with solid anchors

(d) Target of RPN (e) Mini mask (f) Final detection

Figure 5.3: Illustration of *LM1 framework*.

```
Configurations:
BACKBONE                       resnet101
BACKBONE_SHAPES                [[256 256]
 [128 128]
 [ 64  64]
 [ 32  32]
 [ 16  16]]
BACKBONE_STRIDES               [4, 8, 16, 32, 64]
BATCH_SIZE                     2
BBOX_STD_DEV                   [0.1 0.1 0.2 0.2]
DETECTION_MAX_INSTANCES        100
DETECTION_MIN_CONFIDENCE       0.9
DETECTION_NMS_THRESHOLD        0.3
GPU_COUNT                      1
IMAGES_PER_GPU                 2
IMAGE_PADDING                  True
IMAGE_SHAPE                    [1024 1024    3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.001
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               100
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           pothole
NUM_CLASSES                    2
POOL_SIZE                      7
POST_NMS_ROIS_INFERENCE        1000
POST_NMS_ROIS_TRAINING         2000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE              1
RPN_BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD              0.7
RPN_TRAIN_ANCHORS_PER_IMAGE    256
STEPS_PER_EPOCH                100
TRAIN_ROIS_PER_IMAGE           200
USE_MINI_MASK                  True
USE_RPN_ROIS                   True
VALIDATION_STEPS               50
WEIGHT_DECAY                   0.0001
```

Figure 5.4: *LM1* model configurations.

as input generated by the classifier during Stage 2 and predicts a low resolution 28×28 soft mask for it(see Fig. 5.3, bottom row). A soft mask differs from a binary mask as float numbers represent a soft mask and hold more details. We set the object class name as "pothole".

5.2.1 Hyper-parameters setting

For the *LM1* model, per pixel manually labelled ground truth images and left-right flips for data augmentation are used. A grid search is employed to find an optimal learning rate value. The learning rate values from 1 to 0.000001 are used and we realised that the majority of the learning rates failed to train *LM1* model. A low learning rate such as 0.00001 or a very high one such as 1 never converged, thus causing instability while updating weights.

So we used a learning rate of 0.001 as it helps to avoid the problem of an exploding gradient. The L2 regularisation technique is used in *LM1* to prevent overfitting [135]. We train the network using stochastic gradient descent with a learning momentum of 0.9 to identify an object class as a pothole. A snapshot of *LM1* architecture is shown in Fig. 5.4.

The batch size is 2 and the network is trained for 30 epochs, which took 14 hours on our Ge Force GTX 1080 GPU. The *LM1* model achieved an overall `precision` and `recall` on the testing dataset as 88.7 and 84.6 respectively. .

Loss. Loss function is used to measure how inaccurate the obtained output is compared to the desired outcome. In the *LM1* model, the loss is defined and calculated using the following equations [105]:

$$LM1_{loss} = LM1_{class} + LM1_{bbox} + LM1_{mask} \tag{5.1}$$

where

$$LM1_{class} = \frac{1}{N_{class}} \sum_i -a_i^* log a_i - (1 - a_i^*) log(1 - a_i) \tag{5.2}$$

$$LM1_{bbox} = \frac{\lambda}{N_{bbox}} \sum_i a_i^* \cdot L_1^{smooth}(b_i - b_i^*) \tag{5.3}$$

$$LM1_{mask} = -\frac{1}{m^2} \sum_{1 \leqslant i,j \leqslant m} [l_{ij} log \hat{l}_{ij}^t + (1 - l_{ij}) log(1 - \hat{l}_{ij}^t)] \tag{5.4}$$

where $LM1_{class}$ is a loss function over two classes, $LM1_{bbox}$ is bounding box loss, and $LM1_{mask}$ is a mask loss. N_{class} is a normalisation parameter, a_i, a_i^* are predicted and ground truth probability of an object being detected, respectively.

λ is a learning rate constant and b_i, b_i^* are predicted and ground truth four coordinate values. l_{ij} is the label of cell(i, j) in the true mask and $\hat{l}_{ij}^t$ is predicted value

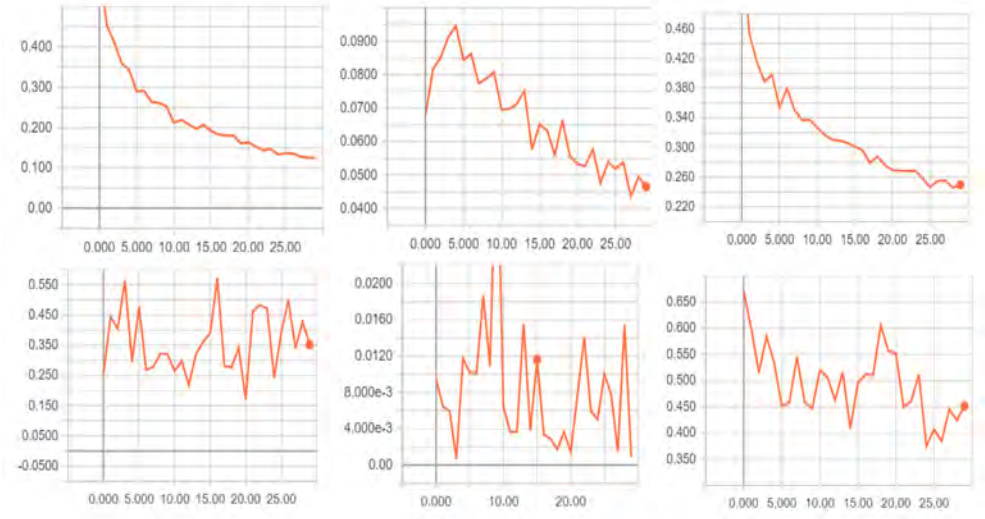

Figure 5.5: $LM1_{box} + LM1_{class} + LM1_{mask}$ during training (upper row) and validation (bottom row). Here the horizontal axis represents the number of epochs and the vertical axis represents loss.

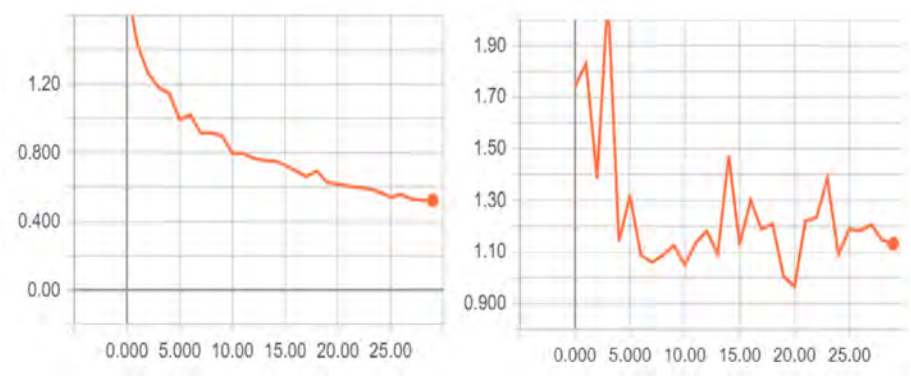

Figure 5.6: $LM1_{loss}$ during training and validation. Here the horizontal axis represents the number of epochs and the vertical axis represents loss.

of same cell for ground-truth class t.

The $LM1_{class} + LM1_{bbox} + LM1_{mask}$ during training and validation is shown in Fig. 5.5. The plots in this figure show that the parameters for the models are chosen sensibly. The parameters are converged to a stable set of weights, which is a desirable property for a model at the end of a training run. The average training and validation losses in Fig. 5.6 show that, after a certain number of iterations, loss does not increase further.

The loss values shown in Figs. 5.7, 5.8, and 5.9 show decreasing loss values at iterations of epochs 1, 15 and 30. The decreasing loss values are highlighted with

```
Epoch 1/30

 90/100 - loss: 1.4874 - mrcnn_class_loss: 0.0782 - mrcnn_bbox_loss: 0.5189 - mrcnn_mask_loss: 0.3788
 91/100 - loss: 1.4841 - mrcnn_class_loss: 0.0776 - mrcnn_bbox_loss: 0.5182 - mrcnn_mask_loss: 0.3786
 92/100 - loss: 1.4819 - mrcnn_class_loss: 0.0770 - mrcnn_bbox_loss: 0.5170 - mrcnn_mask_loss: 0.3799
 93/100 - loss: 1.4802 - mrcnn_class_loss: 0.0763 - mrcnn_bbox_loss: 0.5151 - mrcnn_mask_loss: 0.3825
 94/100 - loss: 1.4776 - mrcnn_class_loss: 0.0760 - mrcnn_bbox_loss: 0.5142 - mrcnn_mask_loss: 0.3823
 95/100 - loss: 1.4756 - mrcnn_class_loss: 0.0755 - mrcnn_bbox_loss: 0.5143 - mrcnn_mask_loss: 0.3825
 96/100 - loss: 1.4749 - mrcnn_class_loss: 0.0755 - mrcnn_bbox_loss: 0.5131 - mrcnn_mask_loss: 0.3829
 97/100 - loss: 1.4681 - mrcnn_class_loss: 0.0751 - mrcnn_bbox_loss: 0.5111 - mrcnn_mask_loss: 0.3817
 98/100 - loss: 1.4658 - mrcnn_class_loss: 0.0750 - mrcnn_bbox_loss: 0.5100 - mrcnn_mask_loss: 0.3824
 99/100 - loss: 1.4615 - mrcnn_class_loss: 0.0747 - mrcnn_bbox_loss: 0.5093 - mrcnn_mask_loss: 0.3819
100/100 - loss: 1.4590 - mrcnn_class_loss: 0.0742 - mrcnn_bbox_loss: 0.5105 - mrcnn_mask_loss: 0.3808
```

Figure 5.7: First epoch

```
Epoch 15/30
 90/100 - loss: 0.3446 - mrcnn_class_loss: 0.0212 - mrcnn_bbox_loss: 0.0871 - mrcnn_mask_loss: 0.1678
 91/100 - loss: 0.3484 - mrcnn_class_loss: 0.0216 - mrcnn_bbox_loss: 0.0879 - mrcnn_mask_loss: 0.1699
 92/100 - loss: 0.3510 - mrcnn_class_loss: 0.0217 - mrcnn_bbox_loss: 0.0894 - mrcnn_mask_loss: 0.1704
 93/100 - loss: 0.3509 - mrcnn_class_loss: 0.0219 - mrcnn_bbox_loss: 0.0891 - mrcnn_mask_loss: 0.1705
 94/100 - loss: 0.3501 - mrcnn_class_loss: 0.0217 - mrcnn_bbox_loss: 0.0888 - mrcnn_mask_loss: 0.1704
 95/100 - loss: 0.3507 - mrcnn_class_loss: 0.0218 - mrcnn_bbox_loss: 0.0885 - mrcnn_mask_loss: 0.1708
 96/100 - loss: 0.3507 - mrcnn_class_loss: 0.0217 - mrcnn_bbox_loss: 0.0883 - mrcnn_mask_loss: 0.1711
 97/100 - loss: 0.3507 - mrcnn_class_loss: 0.0219 - mrcnn_bbox_loss: 0.0883 - mrcnn_mask_loss: 0.1712
 98/100 - loss: 0.3497 - mrcnn_class_loss: 0.0219 - mrcnn_bbox_loss: 0.0880 - mrcnn_mask_loss: 0.1709
 99/100 - loss: 0.3497 - mrcnn_class_loss: 0.0221 - mrcnn_bbox_loss: 0.0877 - mrcnn_mask_loss: 0.1709
100/100 - loss: 0.3483 - mrcnn_class_loss: 0.0220 - mrcnn_bbox_loss: 0.0874 - mrcnn_mask_loss: 0.1703
```

Figure 5.8: Second epoch

```
Epoch 30/30

 90/100 - loss: 0.1877 - mrcnn_class_loss: 0.0134 - mrcnn_bbox_loss: 0.0365 - mrcnn_mask_loss: 0.1114
 91/100 - loss: 0.1872 - mrcnn_class_loss: 0.0134 - mrcnn_bbox_loss: 0.0365 - mrcnn_mask_loss: 0.1113
 92/100 - loss: 0.1875 - mrcnn_class_loss: 0.0132 - mrcnn_bbox_loss: 0.0368 - mrcnn_mask_loss: 0.1114
 93/100 - loss: 0.1870 - mrcnn_class_loss: 0.0132 - mrcnn_bbox_loss: 0.0367 - mrcnn_mask_loss: 0.1112
 94/100 - loss: 0.1869 - mrcnn_class_loss: 0.0130 - mrcnn_bbox_loss: 0.0368 - mrcnn_mask_loss: 0.1111
 95/100 - loss: 0.1867 - mrcnn_class_loss: 0.0130 - mrcnn_bbox_loss: 0.0366 - mrcnn_mask_loss: 0.1111
 96/100 - loss: 0.1868 - mrcnn_class_loss: 0.0131 - mrcnn_bbox_loss: 0.0366 - mrcnn_mask_loss: 0.1110
 97/100 - loss: 0.1869 - mrcnn_class_loss: 0.0130 - mrcnn_bbox_loss: 0.0366 - mrcnn_mask_loss: 0.1112
 98/100 - loss: 0.1865 - mrcnn_class_loss: 0.0131 - mrcnn_bbox_loss: 0.0364 - mrcnn_mask_loss: 0.1111
 99/100 - loss: 0.1866 - mrcnn_class_loss: 0.0131 - mrcnn_bbox_loss: 0.0365 - mrcnn_mask_loss: 0.1110
100/100 - loss: 0.1874 - mrcnn_class_loss: 0.0133 - mrcnn_bbox_loss: 0.0368 - mrcnn_mask_loss: 0.1112
```

Figure 5.9: Third epoch

purple colour marking. The $LM1_{loss}$ at the start of epoch 1 is 2.7060 and at the end of 30 epochs it is 0.1874. Each epoch consists of 100 iterations and in the above figures, last iterations values form $90 - 100$ are shown for epochs 1, 15 and 30. The decreasing loss values are evidence that selected hyper parameters work in a stable manner while updating various parameters in the *LM1* model during training.

5.2.2 False detections of *LM1*

Frames shown in Fig. 5.10 and evaluation measures listed in Table 5.2 are from the validation dataset. All the images from the validation dataset show potholes are identified with a high level of accuracy.

Figure 5.10 shows that, for the validation dataset pothole instances are correctly identified while a false positive was detected in the third image (from the bottom). Under bright sunshine, a tree is miss-classified as a pothole in this case which could

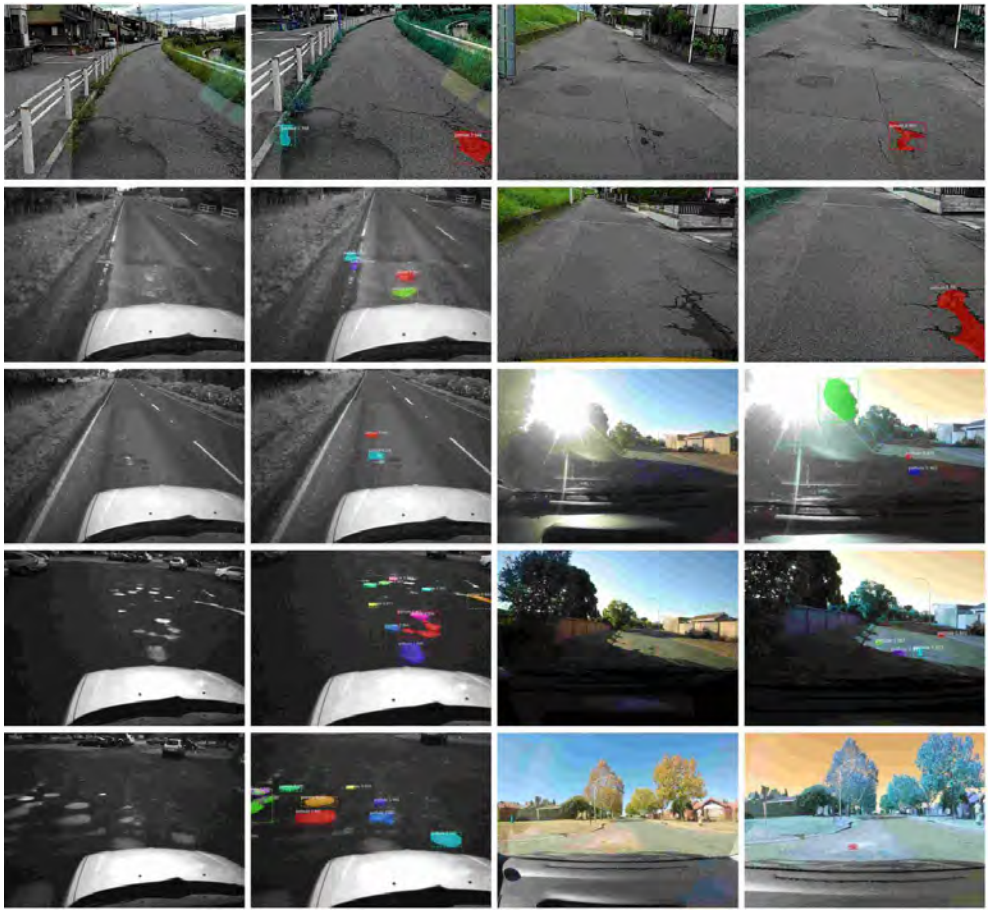

Figure 5.10: Detected "potholes" from the validation dataset using *LM1*, shown in two columns with original image on the left and predicted results on the right. *Top to bottom, left to right*. Ten frames in order as listed in Table 5.2.

Table 5.2: Evaluation measures (in percentage) for the validation dataset using technique *LM1* as shown in Fig. 5.10.

Dataset	Frame	Precision	Recall
Japan	20170912135214	72.8	65.7
DLR	472000	67.7	88.3
DLR	572000	100.0	100.0
DLR	749000	100.0	92.2
DLR	449000	91.0	36.1
Japan	20170906135035	76.8	86.9
Japan	20170906135037	96.4	72.8
Sunny	G0010116	73.9	26.6
Sunny	G0010118	100.0	100.0
Sunny	G0011873	78.5	50.0

be excluded by identifying a ground manifold first.

A pothole in Fig. 5.11 presents a complex situation of a pothole filled with water and tree shadow (this is frame number 3074). Examples of false detections using *LM1* are shown in Fig. 5.12. Here frames in the top two rows are from continuous frame number 3070 to 3076. Frames in this range have only one pothole yet it has

Figure 5.11: Pothole marked in red colour presents a very complex situation as it is filled with water and shadow of a tree.

been detected accurately in frame number 3073 (second-row first frame). The bottom row shows frames where some potholes are identified on the side of the road, especially where a region is mainly a combination of snow and different concrete colours. In general, the *LM1* results are good, keeping in mind that Mask R-CNN has been trained on an entirely different pothole dataset. Many interesting observations of false detections are presented in this section. The reasons for false-positives resulting in these frames is because the *LM1* model tried to identify a pothole which might be a patch or an emerging pothole. As a pothole has no defined shape or features, it is hard to identify and label them manually.

Figure 5.12: Examples of false detections from test `PNW` dataset using *LM1*. The detection of potholes including false positives varies from 3070 to 3076 frames shown in top two rows.

5.2.3 Obtained results using *LM1*

In order to measure the accuracy of our models, we calculate standard classification measures `precision` and `recall`. On a per-pixel basis, `Precision` measures the correctness among all positive pothole instances, `recall` measures how many positive pothole instances are successfully identified among all positive pothole instances. Figures. 5.13 and 5.14 show some randomly selected frames from testing dataset and Table 5.3 shows `precision` and `recall` values for these frames.

Figure 5.13: Detected road potholes of `CCSAD Urban sequence` 2 in upper figure *LM1*, with original image on the left and obtained results on the right. *Top to bottom, left to right*: Frames in order as listed in Table 5.2.

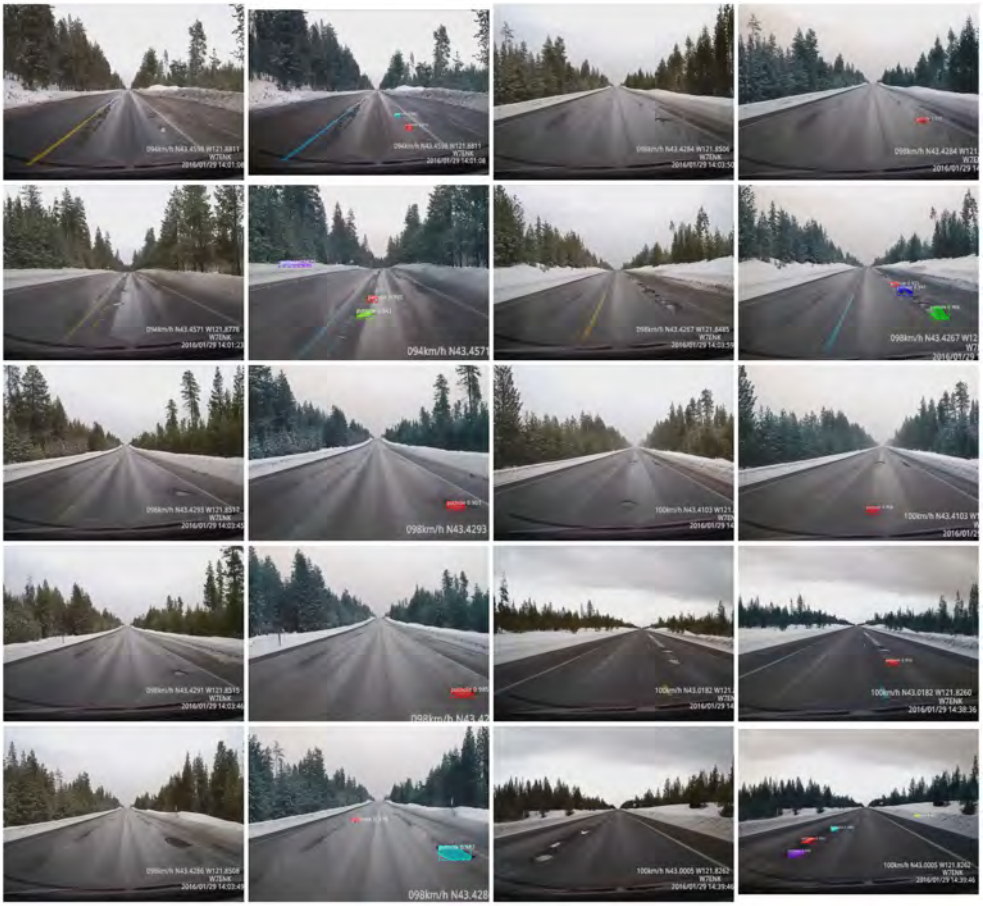

Figure 5.14: Detected road potholes of PNW dataset using *LM1*, with original image on the left and obtained results on the right. *Top to bottom, left to right*: Frames in order as listed in Table 5.2.

This method misclassifies a pothole in PNW frame 720. The reason is that the network identified a region as a pothole, which is a darker region on the side of the pothole. The precision and recall values in Table 5.2 also shows that precision drops drastically for frames 779 and 282. The reasons for false-positives in these two frames are that our network tried to identify a pothole which might be a patch or an emerging pothole. However, the *LM1* method has great potential to identify potholes whether they are dry, or filled with water or snow. The overall precision

Table 5.3: Evaluation metric for test dataset; in percentage.

Dataset	Frame	Precision	Recall
CCSAD 2	282	60.3	92.4
CCSAD 2	345	96.4	80.2
CCSAD 2	382	88.6	79.1
CCSAD 2	740	100.0	94.1
CCSAD 2	779	42.0	86.9
CCSAD 2	831	100.0	56.3
CCSAD 2	958	100.0	83.7
CCSAD 2	1020	82.7	77.6
CCSAD 2	1043	100.0	65.0
CCSAD 2	1119	100.0	97.8
PNW	266	100.0	100.0
PNW	720	41.2	76.6
PNW	1710	84.7	81.7
PNW	1756	82.9	84.3
PNW	1844	100.0	100.0
PNW	1871	100.0	100.0
PNW	2159	100.0	70.7
PNW	2832	100.0	67.1
PNW	18343	100.0	100.0
PNW	19388	96.9	100.0
Overall		88.7	84.6

and `recall` for randomly chosen 50 frames from our testing dataset are 88% and 84% respectively.

5.3 Comparision of *LM1, SV1, SV2*

We comparatively tested the *LM1* and *SV1, SV2* techniques. Results of *LM1*, are great improvement compared to the same example-frames as shown in Figs. 5.15 and 5.16 when using the *SV2* or *SV1* approaches.

Table 5.4 lists results for a few frames of the tested `CCSAD Urban Sequence 1`, comparing pixel-wise detected potholes with the manually labelled ground truth.

Figure 5.15: The potholes found using *SV1* and *SV2* approaches are marked in red and green respectively, while the manually labeled ground truths are marked in blue.

Figure 5.16: The potholes found using *LM1* model and frames are same as Fig. 5.15.

The table shows that there is a case for frame 304 where the *SV2* method provides a *slightly* better result. In general, this is very hard to identify and annotate whether there is actually a pothole(see Fig. 5.17). It could benefit by having more than one human annotator. This also demonstrates the general observation that the *LM1* method outperforms the *SV1* method in a majority of cases, and typically by providing *much* better results.

Figure 5.17: Pothole candidate marked with orange colour is hard to identify for annotation.

Table 5.4: Comparative evaluation of the proposed *LM1, SV1* and *SV2* in the table (in percentages).

Frame	Precision			Recall		
	LM1	SV1	SV2	LM1	SV1	SV2
0078	**96.2**	67.4	65.2	**92.2**	46.6	53.2
0093	**93.1**	57.3	64.0	**93.7**	69.2	61.4
0278	**84.6**	78.2	63.1	**94.1**	41.0	52.9
0304	80.7	76.5	**92.5**	**84.4**	72.8	57.6
0547	**89.0**	37.1	81.5	**95.6**	24.0	34.7
0935	**95.4**	24.2	69.7	**97.1**	45.1	49.1
Means	**89.8**	45.8	67.4	**92.8**	45.8	51.2

5.4 Using Transfer learning with YOLOv2 - *LM2*

For real-time detection of potholes, we used transfer learning using another object detector, *You only look once* (YOLO), addressed in this work as *LM2*. The training dataset for *LM2* is the same as for *LM1*. The annotation format is different in the case of *LM2*, which is a bounding box and not a mask. The layer architecture screen-shot of *LM2* [136] is shown in Fig. 5.18.

```
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   416 x 416 x   3   ->   416 x 416 x  32
    1 max          2 x 2 / 2   416 x 416 x  32   ->   208 x 208 x  32
    2 conv     64  3 x 3 / 1   208 x 208 x  32   ->   208 x 208 x  64
    3 max          2 x 2 / 2   208 x 208 x  64   ->   104 x 104 x  64
    4 conv    128  3 x 3 / 1   104 x 104 x  64   ->   104 x 104 x 128
    5 conv     64  1 x 1 / 1   104 x 104 x 128   ->   104 x 104 x  64
    6 conv    128  3 x 3 / 1   104 x 104 x  64   ->   104 x 104 x 128
    7 max          2 x 2 / 2   104 x 104 x 128   ->    52 x  52 x 128
    8 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
    9 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128
   10 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256
   11 max          2 x 2 / 2    52 x  52 x 256   ->    26 x  26 x 256
   12 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
   13 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
   14 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
   15 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256
   16 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512
   17 max          2 x 2 / 2    26 x  26 x 512   ->    13 x  13 x 512
   18 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
   19 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
   20 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
   21 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512
   22 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024
   23 conv   1024  3 x 3 / 1    13 x  13 x1024   ->    13 x  13 x1024
   24 conv   1024  3 x 3 / 1    13 x  13 x1024   ->    13 x  13 x1024
   25 route  16
   26 reorg              / 2    26 x  26 x 512   ->    13 x  13 x2048
   27 route  26 24
   28 conv   1024  3 x 3 / 1    13 x  13 x3072   ->    13 x  13 x1024
   29 conv     30  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x  30
```

Figure 5.18: *LM2* layer architecture.

This consists of 22 convolutional layers and 5 maxpool layers at layers $1, 3, 7, 11, 17$. The route layer at 25 combines the output of finer grained features from the previous output of layers 16. The output of layer 26 and 24 is concatenated using a reorganization layer at 25.

5.4.1 Hyper-parameters setting

We started with setting the input image subdivision value as 8, but due to a resulting high memory requirement, we changed it to 32. We used 64 images per batch[1]. To optimize the produced weights, this *LM2* model uses three loss values as functions as

$$LM2_{loss} = LM2_{class} + LM2_{bbox} + LM2_{conf} \tag{5.5}$$

$$LM2_{class} = \sum_{i=0}^{g^2} I_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}(c))^2 \tag{5.6}$$

$$\begin{aligned} LM2_{bbox} &= \lambda_{bbox} \sum_{i=0}^{g^2} \sum_{j=0}^{b} I_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \\ &\quad \lambda_{bbox} \sum_{i=0}^{g^2} \sum_{j=0}^{b} I_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \end{aligned} \tag{5.7}$$

$$\begin{aligned} LM2_{conf} &= \sum_{i=0}^{g^2} \sum_{j=0}^{b} I_{ij}^{obj} [(c_i - \hat{c}_i)^2 + \\ &\quad \lambda_{nobj} \sum_{i=0}^{g^2} \sum_{j=0}^{b} I_{ij}^{nobj} [(c_i - \hat{c}_i)^2 \end{aligned} \tag{5.8}$$

where $LM2_{class}, LM2_{bbox}, LM2_{conf}$ are classification, bounding box and confidence loss, respectively. I_i^{obj} is one when a pothole is present otherwise it is 0. $p_i(c)$ is conditional probability for class c in cell i. $\hat{p}$ is conditional class probability. (x, y)

[1]The batch size in *LM2* is bigger compared to batch size as 2 in *LM1*, this is because of the implementation of the different architecture of used frameworks for *LM1* and *LM2*

is the predicted and $(\hat{x}, \hat{y})$ is the actual bounding box position, similarly (w, h) is width and height of the bounding box. λ is a constant to penalize bounding box predictions. c is the confidence score associated with the bounding box predictor and $\hat{c}$ is the IoU of the predicted to the ground truth bounding box.

Initially, the learning rate was set to 0.01. However, after $1,000$ iterations, the average loss kept on increasing. Therefore, we used 0.0001 for learning rate to avoid fast model divergence to unstable gradients. A snapshot of loss values at different

```
Region Avg IOU: 0.247562, Class: 1.000000, Obj: 0.435531, No Obj: 0.451517,
Region Avg IOU: 0.202149, Class: 1.000000, Obj: 0.424553, No Obj: 0.452533,
Region Avg IOU: 0.212077, Class: 1.000000, Obj: 0.553082, No Obj: 0.452353,
Region Avg IOU: 0.169687, Class: 1.000000, Obj: 0.466995, No Obj: 0.451441,
Region Avg IOU: 0.173869, Class: 1.000000, Obj: 0.420677, No Obj: 0.452847,
Region Avg IOU: 0.232709, Class: 1.000000, Obj: 0.347968, No Obj: 0.452136,

 1: 14.246600, avg loss 14.246600, Learning Rate: 0.0001, Momentum: 0.9, 64
```

Figure 5.19: Highest $LM2_{loss}$ value in the first stage (highlighted as **avg loss).**

```
Region Avg IOU: 0.608993, Class: 1.000000, Obj: 0.375313, No Obj: 0.001983,
Region Avg IOU: 0.767309, Class: 1.000000, Obj: 0.619221, No Obj: 0.001983,
Region Avg IOU: 0.574684, Class: 1.000000, Obj: 0.404246, No Obj: 0.002939,
Region Avg IOU: 0.171185, Class: 1.000000, Obj: 0.029530, No Obj: 0.001456,
Region Avg IOU: 0.629128, Class: 1.000000, Obj: 0.646827, No Obj: 0.002987,
Region Avg IOU: 0.609112, Class: 1.000000, Obj: 0.537753, No Obj: 0.002036,
Region Avg IOU: 0.497971, Class: 1.000000, Obj: 0.586254, No Obj: 0.003800,
Region Avg IOU: 0.531319, Class: 1.000000, Obj: 0.011546, No Obj: 0.001719,
Region Avg IOU: 0.850017, Class: 1.000000, Obj: 0.799140, No Obj: 0.003186,
Region Avg IOU: 0.732077, Class: 1.000000, Obj: 0.746667, No Obj: 0.005210,

566: 0.228596, avg loss 0.260599 , Learning Rate: 0.001, Momentum: 0.9, 36224
```

Figure 5.20: Decreasing $LM2_{loss}$ value in middle stage with learning rate 0.01.

```
Region Avg IOU: 0.783547, Class: 1.000000, Obj: 0.738331, No Obj: 0.003051,
Region Avg IOU: 0.154456, Class: 1.000000, Obj: 0.126468, No Obj: 0.001499,
Region Avg IOU: 0.598334, Class: 1.000000, Obj: 0.721760, No Obj: 0.002554,
Region Avg IOU: 0.737833, Class: 1.000000, Obj: 0.711250, No Obj: 0.001746,
Region Avg IOU: 0.650270, Class: 1.000000, Obj: 0.669002, No Obj: 0.002364,
Region Avg IOU: 0.520425, Class: 1.000000, Obj: 0.563937, No Obj: 0.007412,
Region Avg IOU: 0.875487, Class: 1.000000, Obj: 0.805815, No Obj: 0.002039,
Region Avg IOU: 0.637811, Class: 1.000000, Obj: 0.614071, No Obj: 0.013960,

1132: 0.243277, avg loss 0.209380 , Learning Rate: 0.001, Momentum: 0.9, 72448
```

Figure 5.21: Last stage with lowest $LM2_{loss}$ value.

stages of *LM2* model training is shown in Fig. 5.21. The average loss drops from 14.246600 to 0.209380 in the first 100 iterations; therefore, after 100 iterations, we increased the learning rate from 0.0001 to 0.001. The learning momentum is set as 0.9.

5.4.2 False detections of *LM2*

For the real-world applicability of *LM2*, the different false positives and false negatives are studied. The failed predictions of potholes generally do not have negative consequences. This is avoidable by having camera recordings of mainly the road ahead while ignoring anything above and on the side of the road. Figures 5.22 and 5.23 show various false detections, both false positives and false negatives. The proposed *LM2* model enclosed the utility hole (see in Fig. 5.22, second row) with a

Figure 5.22: False detections identified using *LM2* from `CCSAD` test dataset.

Figure 5.23: False detections identified using *LM2* from PNW test dataset.

bounding box, the reason might be the road around the utility hole does not appear smooth.

Another challenge is the shadow of the tree (see in Fig. 5.22, first row) at the end tips that might be because of the darker region. Notably, there is no false detection (see in Fig. 5.22, third row, first image) due to the shadow of the tree. Figure 5.23 shows some interesting false positive examples identified particularly in some frames (not all) such as snow, tree and marked arrows. These false positives are avoidable by training it using some false negatives or by embedding this model into another road scene understanding model.

Figure 5.24: Example of some randomly selected frames and detected road potholes using *LM2*.

5.4.3 Obtained results using *LM2*

We trained the network for around $8,000$ iterations and saved the weights at every 1000 iterations. The *mean average precision* mAP [137] value for different iterations at $5,400$, $6,400$, $7,400$ are examined. As the mAP value of $5,400$ iterations was higher than other iteration weights and also $Loss_{LM2}$ did not decrease further from 0.090189 after five thousand iterations, using Early Stopping Point, the model was selected after $5,400$ iterations. A sample of frame is shown in Fig. 5.24 from the testing dataset.

Table 5.5: Evaluation measures (in percentage) for *LM2* results shown in Fig. 5.10; note that bb1 represents bounding box 1, and bb2 bounding box 2 for different instances of potholes.

Frame	IoU	Frame	IoU
122	66	8464	68
4667	66	8527(bb1)	76
4685	79	8527(bb2)	79
4691	90	8540	55
7044(bb1)	76	8664	67
7044(bb2)	75	8874(bb1)	45
7335	60	8874(bb2)	55
7337	60	10064	83
7476	52	10595	90
8133	51	12601	89
Averaged			69

Video testing using *LM2*.

The *LM2* is fast enough to process videos at a fps rate of $25, 29, 31$. The short video results of PNW test dataset using *LM2* can be seen using this link - https://vimeo.com/337886918. This video is 29 fps. Another video [138] tested is also available at https://vimeo.com/340338524.

The Table 5.5 lists IoU values for some of the randomly chosen frames, see Fig. 5.24. The mean is 69% because to annotate potholes which are always of irregular shape, using the bounding box is complicated. Using the *LM2* method, the developed model is applicable for real time scenarios. In Table 5.5 IoU is greater than 0.5 (which is a standard value), we can say that results are promising. We also tested 50 randomly selected PNW frames at different IoU values 0.5, 0.6 and 0.7 and the obtained average precision values are 60%, 52% and 45%, respectively.

The PNW test frame gets divided into the same number of grids as selected during the training period, i.e. 13×13. The model can predict multiple bounding boxes in each grid, so we kept one with the highest IoU value. This leads to the enforcement of spatial diversity in making predictions.

The originality of our work lies in focusing on pothole identification under challenging illumination and weather conditions. The recent advancements in the area of deep learning hugely improved the field of object detection. The *LM2* model

saves identified pothole images separately, which can be used for analysing to plan in advance to patch up potholes. This will help to automate the laborious and expensive task of manual pothole identification. Though the accuracy of the *LM2* model is not as high as *LM1* model, it still shows the interesting trade-off between speed and accuracy.

5.5 Summary

The prediction accuracy of both *LM1* and *LM2* can be increased by using more training datasets. However, annotating potholes is a tedious task, especially pixel-wise, which requires a lot of time. The *LM2* model has the potential of real-world application as supported by video experiments. For pothole detection, we had implemented the models with a relatively small dataset. By studying the false detections of both the models, the additional training dataset can be selectively chosen to make the *LM2* model ready for commercial use.

We utilised modern approaches for object detection and demonstrated that pothole identification could be automated in real time. This work also fills the gap of different datasets recording under different scenarios. We found out that the *LM1* method performs better than the *LM2* method with high accuracy, but it is much slower than *LM2*. In this study *LM1* served as a proof of concept and encouraged the implementation of the *LM2* model. However, with more training, labelled datasets and use of more than one GPU, the reliability of the models can be increased. Our models allow for future possibilities in numerous ways, such as using the output of the *LM1* method as an annotated image to train the *LM2* method. With more training data, there is definitely a scope for trying *YOLO version* 3 (YOLOv3) [139]. YOLOv3 is an incremental improvement of YOLOv2 and consists of 53 convolution layers.

While this research provides promising steps toward pothole identification, one could extend these models to extract a variety of other metrics such as depth and size of identified potholes. The reported research was motivated by collaboration with *Northland Innovation Centre*'s N3T project; see [140–142] for joint publications so far, also including the *German Aerospace Centre* (DLR) in the context of their IPS [114, 115].

Chapter 6

Conclusion

The gravity of pothole related accidents can be understood by the increase in the number of accidents around the world due to potholes. In this research, four different techniques are proposed and tested against one another. Each technique has its own benefits and can provide different pathways to a number of applications. The *LM1* model can identify a pothole under challenging weather conditions with high `precision` and `recall` whereas the *LM2* model is capable of real time pothole identification. Further work using a more comprehensive dataset and analysis including false positives is needed for the applicability of the model in the real world. The *SV2* model can identify potholes and road manifolds when used with stereo vision cameras. The findings that we have presented here suggest that it is very difficult to define the irregular shape of a pothole which further makes it difficult to annotate ground truth. This in turns presents a complex process of matching results with ground truth. To date, there is no platform or benchmark available for pothole identification. As a result of conducting this research, we put forward six datasets specifically for pothole identification and discuss applications of two different areas of research such as computer vision and deep learning. It would be fruitful to pursue further research in order to combine the output of *LM1* for annotating pothole data and use it to train more *LM2* in order to increase its accuracy for real time purposes.

Bibliography

[1] S. P. Reddit, "Road melts around NZ under brutal summer heat", www.newshub.co.nz/home/new-zealand/2019/01/roads-melt-around-new-zealand-under-brutal-summer-heat.html, January 26, 2019.

[2] "The pothole facts", www.pothole.info/the-facts, last accessed - July 07, 2019.

[3] S. Williams "Compensation requests fall on deaf ears", www.neighbourly.co.nz/e-edition/east-bays-courier/27329, January 17, 2018.

[4] C. M. Phili, "Deadly pits: Potholes claimed 11,386 lives during 2013-16", www.timesofindia.indiatimes.com/india/deadly-pits-potholes-claimed-11386-lives-during-2013-16/articleshow/60774243.cms, September 21, 2017.

[5] J. Horowitz, "All roads lead to Rome, where potholes will destroy your tires" , www.nytimes.com/2018/03/25/world/europe/italy-rome-potholes.html, March 25, 2018.

[6] J. Guildford, "Christchurch the pothole capital of New Zealand", www.stuff.co.nz/the-press/news/100847641/christchurch-the-pothole-capital-of-new-zealand/, February 04, 2018.

[7] D. O'Carroll, "For the love of pizza, Domino's is now fixing potholes in roads", www.stuff.co.nz/motoring/104643123/for-the-love-of-pizza-dominos-is-now-fixing-potholes-in-roads, June 13, 2018.

[8] V. Mucibabic, "420 million budget for potholes welcome but it is not enough", www.openaccessgovernment.org/420-million-budget-for-potholes-welcome-but-it-is-not-enough/53805/, October 30, 2018.

[9] D. Knuth, T.L. Larrabee, and P.M. Roberts, "Mathematical Writing", Mathematical Association of America, ISBN: 088385063X, 1989.

[10] Jaguar Land Rover, "Pothole detection technology annunced by Jaguar land rover" , www.landrover.com/experiences/news/pothole-detection.html, last accessed - July 02, 2019.

[11] J. Michener, "A smarter Suspension", www.corporate.ford.com/innovation/solving-the-bumpy-commute.html, last accessed - July 04, 2019.

[12] Magic Body Scan, www.mercedes-benz.com/en/mercedes-benz/innovation/magic-body-control/, September 13, 2013.

[13] Fugro roadware, Canada, www.roadware.com/applications/, last accessed - July 05, 2019.

[14] Greenwood - profiling roads, www.greenwood.dk/road.php, last accessed - July 05, 2019.

[15] Clifton associates Ltd., "Pothole Identification, Assessment and Repair Guidelines", www.suma.org/img/uploads/documents/communities_of_tomorrow/Pothole%20Guidelines.pdf, 2012.

[16] B. -H. Lin, and S. -F. Tseng, "A predictive analysis of citizen hotlines 1999 and traffic accidents: A case study of Taoyuan city", in Proc. *Int. Conf. Big Data Smart Computing*, pp. 374–376, 2017.

[17] D. Santani, J. Njuguna, T. Bills, A. W. Bryant, R. Bryant, J. Ledgard, and D. G. -Perez, "CommuniSense: Crowdsourcing road hazards in Nairobi", in Proc. *Int. Conf. Human-Computer Interaction Mobile Devices Services*, pp. 445–456, 2015.

[18] Fixmystreet, `fixmystreet.org.nz/`, last accessed - July 05, 2019.

[19] C.-W. Yi, Y. -T. Chuang, and C. S. Nian, "Toward crowdsourcing-based road pavement monitoring by mobile sensing technology", *J. Trans. Intelligent Transportation Systems*, 16(4), 1905–1917, 2015.

[20] A. Mednis, G. Stardins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using smartphones with accelerometers", in Proc. *Int. Conf. Distributed Computing Sensor Systems Workhops*, pp. 1–6, 2011.

[21] H. Kong, J. Y. Audibert, and J. Ponce, "General road detection from a single image", *J. Image Processing*, 19(8): 2221–2220, 2010.

[22] C. Koch and I. Brilakis, "Pothole detection in asphalt pavement images", *J. Advanced Engineering Informatics*, 25(3): 507–515, 2011.

[23] X. Ai, Y. Gao, J.G. Rarity, and N. Dahnoun, "Obstacle detection using U-disparity on quadratic road surfaces", in Proc. *Int. Conf. Intelligent Transportation Systems*, pp. 1352–1357, 2013.

[24] F. Oniga, S. Nedevschi, M. M. Meinecke, and T. B. To, "Road surface and obstacle detection based on elevation maps from dense stereo", in Proc. *Int. Conf. Intelligent Transportation Systems*, pp. 859–865, 2007.

[25] M. Xiangyang, L. Lin, T. Nan, C. Liu, and J. Xiuhan, "Laser-based system for highway pavement texture measurement", in Proc. Int. Conf. Intelligent Transportation Systems, pp. 1559–1562. 2003.

[26] J. Laurent, M. Talbot, and M. Doucet, "Road surface inspection using laser scanners adapted for the high precision 3D measurements of large flat surfaces", in Proc. *Int. Conf. Recent Advances*, pp. 303–310, 1997.

[27] U. Spagnolini and V. Rampa, "Multitarget detection/tracking for monostatic ground penetrating radar: Application to pavement profiling", in Proc. *IEEE Tran. Geoscience Remote Sensing*, pp. 383–394, 1999.

[28] J. Ren and D. Liu, "PADS: A reliable pothole detection system using machine learning", in Proc. *Int. Conf. Smart Computing and Communication*, pp. 327–338, 2016.

[29] M. Ghadge, D. Pandey, and D. Kalbande, "Machine learning approach for predicting bumps on road", in Proc. *Int. Conf. Applied Theoretical Computing Communication Technology*, pp. 481–485, 2015.

[30] M. Badurowicz, T. Cieplak, and J. Montusiewicz, "On-the-fly community-driven mobile accelerometer data analysis system for road quality assessment", *J. Applied Computer Science*, 12(4): 18–27, 2016.

[31] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring", in Proc. *Int. Conf. MSAS*, pp. 29–39, 2008.

[32] K. Chen, M. Lu, X. Fan, M. Wei, and J. Wu, "Road condition monitoring using on-board three-axis accelerometer and GPS sensor", in Proc. *Int. Conf. Communications Networking*, pp. 1032–1037, 2011.

[33] N. Kalra, G. Chugh, D. Bansal, "Analyzing driving and road events via smartphone", *J. Computer Applications*, 98(12): 5–9, 2014.

[34] M. Fekry, A. Hamdy, and A. Atia, "Anti-Bump: A bump/pothole monitoring and broadcasting system for driver awareness", in Proc. *Int. Conf. Human-Computer Interaction*, pp. 561–570, 2013.

[35] Y. Ren, G. Wen, and X. Li, "An SVM based algorithm for road disease detection using accelerometer", *J. Electrical Engineering Computer Science*, 11(9): 5169–5175, 2013.

[36] K. Georgieva, C. Koch, and M. König, "Wavelet transform on multi-GPU for real-time pavement distress detection", *J. Computing Civil Engineering*, 99–106, 2015.

[37] K. Doycheva, C. Koch, and M. König, "Implementing textural features on GPUs for improved real-time pavement distress detection", *J. Real-Time Image Processing*, 1–12, 2016.

[38] A. Tedeschi and F. Benedetto, "A real-time automatic pavement crack and pothole recognition system for mobile Android-based devices", *J. Advanced Engineering Informatics*, 32: 11–25, 2017.

[39] R. M. Haralick, K. Shanmugam and I. Dinstein, "Textural features for image classification", *J. Systems Man Cybernetics*, 3(6): 610–621, 1973.

[40] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, "*Data Mining: Practical Machine Learning Tools and Techniques*", 3rd ed., Morgan Kaufmann, Burlington, 2011.

[41] S. -K. Ryu, T. Kim, and Y. -R. Kim, "Image-based pothole detection system for ITS service and road management system", *J. Mathematical Problems Engineering*, doi: 10.1155/2015/968361, 2015.

[42] L. Powell, and KG. Satheeshkumar, "Automated road distress detection", in Proc. *Int. Conf. Emerging Technological Trends*, pp. 1–6, 2016.

[43] V. A. Bashkar and G. T. Manohar, "Surface pothole depth estimation using stereo mode of image processing", *J. Advance Research Engineering Technology*, 4: 1169–1177, 2016.

[44] H. Bay, T. Tinne, and L. V. Gool, "Surf: Speeded up robust features", in Proc. *European conference on computer vision*, pp. 404-417, 2006.

[45] Z. Ying, G. Li, X. Zang, R. Wang, and W. Wang, "A novel shadow-free feature extractor for real-time road detection", in Proc. *Int. Conf. Pervasive Ubiquitous Computing*, pp. 611–615, 2016.

[46] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset", *J. Robotics Research*, 32 (11): 1231–1237, 2013.

[47] T. Veit, J.-P. Tarel, P. Nicolle and P. Charbonnier, "Evaluation of road marking feature extraction", in Proc. *Intelligent Transportation Systems*, pp. 12-15, 2008.

[48] M. V. Thekkethala, R. S., S. J. Varughese, V. Mohan, and G. Titus, "Pothole detection and volume estimation using stereoscopic cameras", in Proc. *Int. Conf. Mixed Design Integrated Circuits Systems*, pp. 47–51, 2016.

[49] A. Akagic, E. Buza, and S. Omanovic, "Pothole detection: an efficient vision-based method using RGB color space image segmentation", in Proc. *Information Communication Technology, Electronics and Microelectronics*, pp. 1104–1109, 2017.

[50] Q. Li, M. Yao, X. Yao, and B. Xu, "A real-time 3D scanning system for pavement distortion inspection", *J. Measurement Science Technology*, 21(1): 015702, 2009.

[51] X. Yu and E. Salari, "Pavement pothole detection and severity measurement using laser imaging", in Proc. *Int. Conf. Electro Information Technology*, pp. 1–5, 2011.

[52] K. K. Vupparaboina, R. R. Tamboli, P. M. Shenu, and S. Jana, "Laser-based detection and depth estimation of dry and water-filled potholes: A geometric approach", in Proc. *Nat. Conf. Communications*, pp. 1–6, 2015.

[53] R. Fan, J. Jiao, J. Pan, H. Huang, S. Shen and M. Liu, "Real-time dense stereo embedded in a UAV for road inspection", in Proc. *Int. Conf. CVPR* 2019.

[54] Y. Pan, X. Zhang, M. Sun, and Q. Zhao, "Object-based and supervised detection of potholes and cracks from the pavement images acquired by UAV" *Int. Archives Photogrammetry, Remote Sensing Spatial Information Sciences* doi: 10.5194/isprs-archives-XLII-4-W4-209-2017, 2017.

[55] R. Klette, "*Concise Computer Vision: An Introduction into Theory and Algorithms*", Springer, London, 2014.

[56] T. Garbowski and T. Gajewski, "Semi-automatic inspection tool of pavement condition from three-dimensional profile scans", *J. Intelligent Transportation Systems*, 172: 310–318, 2017.

[57] Femat project, `www.fematproject.pl/index.html` last accessed - July 05, 2019.

[58] A. Rasheed, K. Kamal, T. Zafar, and S. Mathavan, and M. Rahman, "Stabilization of 3D pavement images for pothole metrology using the Kalman filter", in Proc. *Int. Conf. Intelligent Transportation Systems*, pp. 2671–2676, 2015.

[59] T. Shen, G. Schamp, M. Haddad, "Stereo vision based road surface preview", in Proc. *Int. Conf. Intelligent Transportation Systems*, pp. 1843–1849, 2014.

[60] A. Mikhailiuk and N. Dahnoun, "Real-time pothole detection on TMS320C6678 DSP", Proc. *Int. Conf. Imaging Systems Techniques*, pp. 123–128, 2016.

[61] Z. Zhang, X. Ai, C. K. Chan, and N. Dahnoun, "An efficient algorithm for pothole detection using stereo vision", in Proc. *Int. Conf. Acoustics, Speech and Signal Processing*, pp. 564–568, 2014.

[62] M. Staniek, "Neural networks in stereo vision evaluation of road pavement condition", in Proc. *Int. Symp. Non-Destructive Testing Civil Engineering*, pp. 15–17, 2015.

[63] B. Cyganek and J. P. Siebert, "*An Introduction into 3D Computer Vision Techniques and Algorithms*", John Wiley & Sons, UK, 2011.

[64] H. Song, K. Baek, and Y. Byun, "Pothole detection using machine learning", *J. Advanced Science Technology Letters*, 150: 151–155, 2018.

[65] C. Szegedy, V. Vanhoucke, S. Ioffe, J. SHlens, and Z. Wojna, "Rethinking the inception architecture for computer vision", in Proc. *Int. Conf. CVPR*, pp. 2818–2826, 2016.

[66] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, "Road damage detection using deep neural networks with images captured through a smartphone", *J. Computer-Aided Civil Infrastructure Engineering*, 33(12): 1127-1141, 2018.

[67] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed accuracy trade-offs for modern convolutional object detectors", in Proc. *Int. Conf. Computer Vision Pattern Recognition*, pp. 73100–7311, 2017.

[68] A. G. Howard, M.Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications", *J. Computer Research Repository*, arXiv: 1704.04861, 2017.

[69] A. Zhang, K. Wang, B. Li, E. Yang, X. Dai, Y. Peng, Y. Fei, Y. Liu, J. Q. Li, and C. Chen, "Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network", *J. Computer-Aided Civil Infrastructure Engineering*, 32(10): 805–819, 2017.

[70] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation", *J. IEEE Trans. Pattern Analysis Machine Intelligence*, 39(12): 2481 - 2495, 2017.

[71] N. D. Hoang, "An artificial intelligence method for asphalt pavement pothole detection using least squares support vector machine and neural network with steerable filter-based feature extraction", *J. Adv. Civil Eng*, doi: 10.1155/2018/7419058, 2018.

[72] Y. J. Cha, C. Wooram, and O. Büyüköztürk, "Deep learningbased crack damage detection using convolutional neural networks", *J.ComputerAided Civil and Infrastructure Engineering*, 32(5): 361-378, 2018.

[73] J. Bray, B. Verma, X. Li, and W. He, "A neural network based technique for automatic classification of road cracks", in Proc. *International Joint Conference on Neural Networks* , pp. 907–912, 2006.

[74] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network", doi: 10.1109/ICIP.2016.7533052, 2016.

[75] K. E. An, S. W. Lee, S. K. Ryu, and D. Seo, "Detecting a pothole using deep convolutional neural network models for an adaptive shock observing in a vehicle driving", in Proc. *Consumer Electronics*, pp. 1–2, 2018.

[76] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network", in Proc. *Int. Conf.Computer Vision Pattern Recognition*, pp. 2881–2890, 2017.

[77] G. Lin, A. Milan, C. Shen, and I. D. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation", in Proc. *Int. Conf. Computer Vision Pattern Recognition*, pp. 1925–1934, 2017.

[78] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters - Improve semantic segmentation by global convolutional network", in Proc. *Int. Conf. Computer Vision Pattern Recognition*, pp. 1743–1751, 2017.

[79] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", in Proc. *Int. Conf. Computer Vision Pattern Recognition*, pp. 3431–3440, 2015.

[80] L. -C Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs", *J. IEEE Trans. Pattern Analysis Machine Intelligence*, 40(4): 834–848, 2018.

[81] L. k. Suong and k. Jangwoo, "Detection of Potholes Using a Deep Convolutional Neural Network", *J. Universal Computer Science*, 24(9): 1244–1257, 2018.

[82] V. Pereira, S. Tamura, S. Hayamizu, and H. Fukai, "A Deep Learning-Based Approach for Road Pothole Detection in Timor Leste", in Proc. *Int. Conf. Service Operations and Logistics, and Informatics*, pp. 279–284, 2018.

[83] K. E. An, S. W. Lee, S. K. Ryu, and D. Seo, "Detecting a pothole using deep convolutional neural network models for an adaptive shock observing in a vehicle driving", in Proc. *Int. Conf.Consumer Electronics*, pp. 1–2, 2018.

[84] Y. Bhatia, R. Rai, V. Gupta, N. Aggarwal, and A. Akula, "Convolutional neural networks based potholes detection using thermal imaging", *J. King Saud University, Computer and Information Sciences*, doi: 10.1016/j.jksuci.2019.02.004, 2019.

[85] H. Youquan, W. Jian, Q. Hanxing, Z. Wei, and X. Jianfang, "A research of pavement potholes detection based on three-dimensional projection transformation", in Proc. *Int. Conf. Image and Signal Processing*, pp. 1805–1808, 2011.

[86] Y. -H. Tseng, S. -C. Kanga, J. -R. Changb, and C. -H. Leea, "Strategies for autonomous robots to inspect pavement distresses", *J. Automation Construction*, 20(8): 1156–1172, 2011.

[87] F. Seraj, B. J. V. D. Zwaag, A. Dilo, T. Luarasi, and P. Havinga, "Roads: A road pavement monitoring system for anomaly detection using smart phones", in Proc. *Int. Wksh. Modeling Social Media*, pp. 128–146, 2014.

[88] T. Naidoo, D. Joubert, T. Chiwewe, A. Tyatyantsi, B. Rancati, and A. Mbizeni, "Visual surveying platform for the automated detection of road surface distresses", in Proc. *Conf. Sensors MEMS Electro-Optic Systems*, doi.org: 10.1117/12.2066116, 2014.

[89] Y. -W. Hsu, J. W. Perng, and Z. -H. Wu, "Design and implementation of an intelligent road detection system with multisensor integration", in Proc. *Int. Conf. Machine Learning Cybernetics*, pp. 219–225, 2016.

[90] J. Lin and Y. Liu, "Potholes Detection Based on SVM in the Pavement Distress Image", in Proc *Int. Conf. Distributing Computing Applications Business Engineering Science*, pp. 544–547, 2010.

[91] A. Kulkarni, N. Mhalgi, S. Gurnani, and N. Giri, "Pothole detection system using machine learning on android", *J. Emerging Technology Advanced Engineering*, 4(7): 360–364, 2014.

[92] C. Zhang and A. Elaksher, "An unmanned aerial vehicle-based imaging system for 3D measurement of unpaved road surface distresses", *J. Computer-Aided Civil Infrastructure Engineering*, 27(2): 118–129, 2012.

[93] D. Scaramuzza and F. Fraundorfer, "Visual odometry" tutorial on *IEEE Robotics Automation Magazine*, 8(4): 80–92, 2011.

[94] B. D. Lucas and K. Takeo, "An iterative image registration technique with an application to stereo vision", in Proc. *Int. Conf. Artificial Intelligence*, pp. 674–679, 1981.

[95] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion", in Proc. *Scandinavian Conf. Image analysis*, pp. 363–370, 2003.

[96] C. Tomasi and T. Kanade, "Detection and tracking of point features", *J. Computer Vision*, 1991.

[97] M. A. Fischler, and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", *J. Communications ACM*, 24(6): 3810–395, 1981.

[98] D. Gale and H. Nikaido, "The Jacobian matrix and global univalence of mappings", *J. Mathematische Annalen*, 159(2): 81–93, 1965.

[99] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", in Proc. *Int. Conf. Computer Vision Pattern Recognition*, pp. 511–518, 2001.

[100] NIPS 1995 workshop, "Knowledge Consolidation and Transfer in Inductive Systems", `www.cs.cmu.edu/afs/cs/project/cnbc/nips/NIPS95/Workshops.html`, 1995.

[101] S. J. Pan, Q. Yang, "A survey on transfer learning", *J. IEEE Transactions on knowledge and data engineering*, 22(10): 1345–59, 2010 Oct.

[102] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in Proc. *Int. Conf. Computer Vision Pattern Recognition*, pp. 580–587, 2014.

[103] R. Girshick, "Fast R-CNN", in Proc. *Int. Conf. Computer Vision*, pp. 1440–1448, 2015.

[104] J. R. R. Uijlings,K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders, "Selective search for object recognition", *J. computer vision*, 104(2): 154–71, 2013.

[105] H. Kaiming, G. Georgia, D. Piotr, and G. Ross, "Mask R-CNN", in Proc. *Int. Conf. Computer Vision Pattern Recognition*, pp. 2961–2969, 2017.

[106] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in Proc. *Int. Conf. Computer Vision Pattern Recognition*, pp. 770–778, 2018.

[107] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection", in Proc. *Int. Conf. Computer Vision Pattern Recognition*, p. 2117–2125, 2017.

[108] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", in Proc. *Int. Conf. Computer Vision Pattern Recognition*, pp. 779–788, 2016.

[109] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger", in Proc. *Int. Conf. Computer Vision Pattern Recognition*, pp. 7263–7271, 2017.

[110] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the Objectness of Image Windows", *J. Pattern Analysis and Machine Intelligence*, 34(11): 2189-2202, 2012.

[111] T. Vaudrey, C. Rabe, R. Klette, and J. Milburn, "Differences between stereo and motion behavior on synthetic and real-world stereo sequences", in Proc. *Int. Conf. Image Vision Computing*, pp. 1–6, 2008.

[112] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth", in Proc. *German Conf. Pattern Recognition*, pp. 31–42, 2014.

[113] R. Guzmán, J. -B. Hayet, and R. Klette, "Towards ubiquitous autonomous driving: The CCSAD dataset", in Proc. *Int. Conf. Computer Analysis Images Patterns*, LNCS 9256, pp. 582–593, 2015.

[114] A. Börner, D. Baumbach, M. Buder, A. Choinowski, I. Ernst, E. Funk, D. Grießbach, A. Schischmanow, J. Wohlfeil, and S. Zuev, "IPS - A vision-aided navigation system", *J. Advanced Optical Technologies*, 6(2): 121–129, 2017.

[115] D. Grießbach, D. Baumbach, and S. Zuev, "Stereo-vision-aided inertial navigation for unknown indoor and outdoor environments", in Proc. *Int. Conf. Indoor Positioning Indoor Navigation*, pp. 709–716, 2014.

[116] S. Nienaber, M.J. Booysen, and R.S. Kroon, "Detecting potholes using simple image processing techniques and real-world footage", in Proc. *Southern African Transport Conference*, 2015.

[117] PNW dataset, `www.youtube.com/watch?v=BQo87tGRM74`, last accessed - July 05, 2019.

[118] A. Dhiman, H.-J. Chien, and R. Klette, "Road surface distress detection in disparity space", in Proc.*Int. Conf. Image Vision Computing New Zealand*, pp. 1–6, 2017.

[119] A. Dhiman, H.-J. Chien, and R. Klette, "A multi-frame stereo vision-based road profiling technique for distress analysis", in Proc.*Symposium Pervasive Systems Algorithms Networks* , pp. 7–14, 2018.

[120] W. Khan, "Accuracy of stereo-based object tracking in a driver assistance context", PhD thesis, Computer Science Department, Auckland University, 2013.

[121] K. Kawamoto and R. Klette, "Dominant plane estimation", *CITRTR-88, Computer Science Department, University of Auckland*, 2001.

[122] R. Labayrade, D. Aubert, and J.-P. Tarel. "Real time obstacle detection in stereovision on non flat road geometry through v-disparity representation," in Proc. *IEEE Intelligent Vehicles Symp.*, pp. 646–651, 2002.

[123] N. H. Saleem, H. -j Chien, M. Rezaei, and R. Klette, "Improved stixel estimation based on transitivity analysis in disparity space," in Proc. *Int. Conf. Computer Analysis Images Patterns*, pp. 28–40, 2017.

[124] J. Serra, "*Image Analysis and Mathematical Morphology*", Orlando: Academic Press, 1983.

[125] R. Klette and A. Rosenfeld, "*Digital Geometry*"' San Francisco: Morgan Kaufmann, 2003.

[126] V. Lepetit, F. Moreno-Noguer, and P. Fua, " Epnp: An accurate o (n) solution to the pnp problem", *J. computer vision*, 81(2): 155, 2009.

[127] H.-J. Chien and R. Klette, "Regularised energy model for robust monocular egomotion estimation," in Proc. *Int. Joint Conf. Computer Vision Imaging Computer Graphics: Theory Applications*, pp. 361–368, 2011.

[128] K.A. Levenberg, "Method for the solution of certain non-linear problems in least squares", *J. The Quarterly Applied Math*, 2: 164–168, 1944.

[129] H. Edelsbrunner, D.G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane", *J. IEEE Trans. Information Theory*, 29(4): 551–559, 1983.

[130] Y. Liu, Z. Chen, W. Zheng, H. Wang, and L. Jianguo, "Monocular Visual-Inertial SLAM: Continuous Preintegration and Reliable Initialization", *J. Sensors* 17(11): 2613, 2017.

[131] H. J. Chien, R. Klette, N. Schneider, and U. Franke, "Visual odometry driven online calibration for monocular lidar-camera systems", in Proc. *Int. Conf.pattern recognition*, pp. 2848–2853, 2016.

[132] N. DÁpuzzo, "Overview of 3D surface digitization technologies in Europe", *J. Three-Dimensional Image Capture and Applications*, 52(4): 546–560, 2006.

[133] S. Pan and Q. Yang, "A Survey on Transfer Learning", *J. IEEE Trans. Knowledge Data Engineering*", 22(10): 1345-1359, 2010.

[134] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common objects in context", in Proc.*European Conf. Computer Vision*, pp. 740–7552014.

[135] J. Schmidhuber, "Deep learning in neural networks: An overview", *J. Neural networks*, (61): 85–117, 2015.

[136] J.Redmon, "Darknet: Open Source Neural Networks in C", `www.pjreddie.com/darknet/`, 2013–2016.

[137] Jonathan Hui, "mAP (mean average precision) for object detection", medium.com, March 07, 2018.

[138] Sudbury, Canada pothole Video, www.youtube.com/watch?v=SyIQirLZB7A, last accessed - July 05, 2019.

[139] J. Redmon and A. Farhadi. "Yolov3: An incremental improvement", *arXiv*, 1804.02767, 2018.

[140] D. Baumbach, H. Zhang, S. Zuev, J. Wohlfeil, M. Knoche and R. Klette, "GPS and IMU Require Visual Odometry for Elevation Accuracy", in Proc. *Advanced Video and Signal-based Surveillance*, pp. 1–6, 2018.

[141] I. Ernst, H. Zhang, S. Zuev, M. Knoche, A. Dhiman, H.-J. Chien, and R. Klette, "Large-scale 3D Roadside Modelling with Road Geometry Analysis: Digital Roads New Zealand", in Proc. *Pervasive Systems, Algorithms and Networks*, pp. 15–22, 2018.

[142] H. Zhang, I. Ernst, S. Zuev, A. Börner, M. Knoche, and R. Klette, "Visual odometry and 3D point clouds under low-light conditions", in Proc. *Image Vision Computing New Zealand*, pp. 1–6, 2018.

Index