A Hybrid Approach to Condition Assessment of Paved Roads

By William Sprague

A thesis
submitted to the Faculty of Graduate Studies
in partial fulfilment of the requirements for the
Degree of Master of Science in
Civil Engineering

Supervisor
Dr. Ehsan Rezazadeh Azar

Lakehead University
Thunder Bay, Ontario

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Road reconstruction is drastically more expensive than an appropriate maintenance program. Performing targeted maintenance requires a comprehensive inventory of road conditions. To create and maintain a road condition inventory, many cities use expensive, purpose-built road profiling vehicles. With the abundance of smartphones, and the variety of sensors they hold, there is an opportunity to collect road condition data more pervasively and at a lower cost. This study shows that smartphones can be used to collect road surface data for the purpose of creating an inventory of road conditions. The data sources within the phone are its onboard accelerometer and camera. Readings from these sensors are taken simultaneously while the phone is mounted to the windshield of a vehicle driving over paved city streets. The collected data was used for measuring road surface roughness and automatically identifying specific defects, such as cracks and potholes.

Road roughness is quantified using the range between acceleration values and pairing them with readings from the smartphone's GPS antenna to create roughness intensity maps. Defect identification is done by a convolutional neural network that can label the pixels of an image it receives into one of several defect pixel label classes. The neural networks used for this study were trained using a manually labelled image set of several hundred images. The novel hybrid approach used in this study combines the accelerometer data with the camera footage. The data are synchronized so that frames can be extracted from the video footage based on anomalies in the acceleration data. Peaks in the acceleration data are used as a trigger to extract and analyze images from the video footage. This approach to defect identification can be done at a much lower computational burden than if all the video footage was processed by a convolutional neural network.

The defect identification results show strong performance in terms of identifying locations of defects, as determined by the acceleration data, and identifying the defects at the identified locations themselves. Our study found that an unmodified vehicle with a smartphone mounted to its windshield can generate data that our system, developed in MATLAB, can find over 60% of road defects. This is achieved in just one pass over the road surface.

Acknowledgements

My deepest gratitude goes to Dr. Ehsan Rezazadeh Azar. He has been much more than a supervisor and has shown me the power of mentorship. This thesis would not exist without him, and I am honoured to call myself his student.

I must also thank my family. Their love and support has been unwavering.

Table of Contents

Abstract III

Acknowledgements IV

List of Figures VIII

List of Tables XI

Chapter 1: Introduction 1

1.1 Background and Research Motivation 1

1.2 Research Objectives 2

1.3 Research Methodology 2

1.4 Thesis Organization 3

Chapter 2: Literature Review 4

2.1 Road Structures and Surfaces 4

2.1.1 Types of Road Surfaces 4

2.1.2 Road Anatomy 5

2.1.3 Pavement Distress 7

2.2 Road Roughness 10

2.2.1 Road Surface Quality 11

2.2.2 Quantifying Road Roughness 12

2.2.3 Indexing Drawbacks 15

2.2.4 Measuring Road Roughness 15

2.3 Smartphone-based Vehicle Telematics 17

2.3.1 Types of Sensors 18

2.3.2 Drawbacks of Smartphone-based Data Collection 20

2.3.3 Data Utilization 21

2.3.4 Challenges 30

2.4 Computer Vision 31
2.4.1 Visual Surface Analysis 31
2.4.2 Image Preprocessing 33
2.4.3 Image Features 35
2.4.4 Defect Detection 37
2.5 Pavement Management Systems 38
2.5.1 Components of a PMS 39
2.5.2 Implementing a PMS 39
2.5.3 PMS Software 40
Chapter 3: Research Methodology 42
3.1 Data Collection 42
3.1.1 Equipment & Setup 43
3.1.2 Driving Routes 47
3.2 Preprocessing 48
3.2.1 Motion Data Preprocessing 48
3.2.2 Video Preprocessing 49
3.2.3 Image Preprocessing 50
3.3 Motion Analysis Process 51
3.3.1 Roughness Metrics 51
3.3.2 Anomaly Detection 54
3.4 Image Analysis Process 57
3.4.1 Thresholding 58
3.4.2 Deep Learning Models 66
3.5 Data Integration 75
3.5.1 Data Synchronization 78

3.5.2 Frame Extraction 78
3.5.3 Geotagging 80
3.5.4 Image Defect Identification 81
3.5.5 KML Data Packaging 82
Chapter 4: Results and Discussion 85
4.1 Data Replicability 85
4.1.1 Replicability of Video Footage 90
4.2 Motion Results 91
4.2.1 Roughness Mapping 91
4.2.2 IRI Challenges 92
4.3 Vision Results 94
4.3.1 Thresholding Results 94
4.3.2 Evaluating CNN Performance 98
4.3.3 CNN Results 102
4.4 Integration Results 109
4.4.1 Integration Example 1 110
4.4.2 Integration Example 2 113
4.5 Challenges 116
Chapter 5: Conclusion 119
5.1 Summary and Conclusions 119
5.2 Limitations of the Research 120
5.3 Recommendations for Future Work 121
Appendix A – MATLAB Code 132
Appendix B – Extended Data & Tables 1647

List of Figures

Figure 1. Types of road surfaces used for infrastructure 5
Figure 2. General anatomy of a road cross-section (Hjort et al. 2008) 6
Figure 3. The idealization of flexible (left) and rigid (right) pavement structures (Lay 2009) 7
Figure 4. Examples of raveling on an asphalt road (Hoang 2019b) 10
Figure 5. Examples of potholes on paved roads 10
Figure 6. Road surface texture at different scales. Left: A rough surface with small-scale regularity; Right: An irregular road with a mix of roughness features at different scales 11
Figure 7. Basic elements of road texture and their impact on driving (Cackler et al. 2006) 12
Figure 8. The quarter car model used in the International Roughness Index (Loizos and Plati 2008) 14
Figure 9. A road surface profiler (Shamsabadi 2019) 17
Figure 10. Triaxial acceleration measurements from a smartphone 20
Figure 11. Acquire, Analyze, Apply—the approach to using smartphone data for a PMS (Douangphachanh and Oneyama 2013b) 21
Figure 12. Left: windshield-mounted smartphone (Souza et al. 2018); Right: dashboard- and windshield-mounted phones (Yeganeh et al. 2019) 22
Figure 13. Acceleration before (left) and after (right) the signal has passed through a maximum absolute value filter (Harikrishnan and Gopi 2017) 24
Figure 14. Threshold-based image segmentation for a pothole (Koch et al. 2011) 34
Figure 15. Identifying the sky in an image using two image processing operations (Canuma 2018) 35
Figure 16: Gradient orientations (left) and their distributions (right) for the SIFT feature descriptor (Treiber 2010) 36
Figure 17. Functions of a pavement management system 39
Figure 18. Example of a PMS user interface (Shamsabadi 2019) 41
Figure 19. Overview of the pavement condition assessment system, from data collection to display 42
Figure 20. Equipment setup used for data collection 44
Figure 21. The six degrees of freedom captured by the phone's motion sensors 45

Figure 22. Example of viewfinder and data provided by Timestamp Camera app 47
Figure 23. Data collection driving routes in Thunder Bay 48
Figure 24: Region of interest within a video frame captured during testing 51
Figure 25. Original acceleration data (top) condensed to a series of MMR values (bottom). 53
Figure 26. A geographic density plot using the MMR roughness metric 54
Figure 27. Percentiles for the entire set of acceleration values collected. 55
Figure 28. Closer view of a pothole-induced spike in acceleration. 56
Figure 29. Acceleration time series plus detected anomalies circled in red. 57
Figure 30. An example of image segmentation with thresholding. 59
Figure 31. An image of a crack in the road and a threshold mask laid on top of it 61
Figure 32. An example of how choosing the right threshold value can affect mask quality 62
Figure 33. The original grayscale image, an equalized version, and their histograms 63
Figure 34. Contrast enhanced image using gamma correction, plus its histogram 64
Figure 35. Mask derived from a contrast-enhance image using AdaptiveThreshold function 64
Figure 36. Image segmented first with a threshold followed by erosion. 65
Figure 37. Basic process of semantic segmentation using a CNN as a classifier 66
Figure 38. Example of working within the MATLAB Image Labeler application 70
Figure 39. Top: Original image; Bottom: Fully labelled image with five classes 71
Figure 40. Steps to integrate the motion and video data 77
Figure 41. Values manually collected to determine speed-based correction factor. 80
Figure 42. Example data display of KML files loaded into Google MyMaps 83
Figure 43. Comparison of acceleration signals for two different vehicles driving over the same stretch of a road twice 86
Figure 44. Average of the four plots in Figure 43 86
Figure 45. Variances of acceleration signals taken over 100-point range 87
Figure 46. Opposite and modified acceleration series compared to a control trial. 90
Figure 47. Roughness map of the selected Thunder Bay streets coloured using MMR intensity 91
Figure 48. Comparison of variance and MMR as roughness metrics 92
Figure 49. Plots comparing the same 20-meter stretch of a road before and after paving 93
Figure 50. Damped free vibration behaviour noticed in the data collector 94
Figure 51. Comparing an image ground truth and adaptive threshold 95

Figure 52. Histogram of results comparing thresholded image masks to the ground truth labels 96
Figure 53. An example of image segmentation with thresholding 97
Figure 54 . Possible outcomes in a binary classification problem 100
Figure 55: Comparing scores for the Crack class in the image training set 102
Figure 56. Class BF scores for different CNN architectures 105
Figure 57. Class BF scores for CNNs trained on modified training data 106
Figure 58. Example of a segmented image with low BF score 109
Figure 59. A difficult to detect transverse crack in the road 111
Figure 60. Automatically and manually identified defect locations along a short route 112
Figure 61. Combination of defects identified automatically and manually 113
Figure 62. Cracks in the road that could cover multiple extracted frames 114
Figure 63. Defect locations automatically detected based on time between acceleration peaks 115
Figure 64. Defect identification in extracted frames compared to the manual review of the video 116

List of Tables

Table 1: Types of pavement distress (Fwa 2005; Hjort et al. 2008; Walker et al. 2002) 8
Table 2. Types of road roughness measuring devices 16
Table 3. Roughness classification and indexing research..... 26
Table 4. Acceleration-based road anomaly detection research..... 28
Table 5. Detection methods used for various pavement defects..... 37
Table 6. Sensors plus their frequency and precision..... 46
Table 7. Summary of outputs of the SensorLog iOS app 46
Table 8. Variables remaining after feature elimination plus their description 49
Table 9. Labels used to classify pixels on frames collected from video footage..... 68
Table 10. Image feature classifications and their pixel labels 72
Table 11. Network architectures used for convolutional neural networks 75
Table 12. Vehicle specifications for comparing vibration patterns between different vehicles ... 85
Table 13. Correlation coefficients for different vehicles driving the same road section..... 89
Table 14. Distribution of labels across training and test dataset images (all values in %)..... 99
Table 15. Results of the classifiers trained using the original label definitions 103
Table 16. Differences in training of CNNs evaluated for defect detection 104
Table 17. Results of the classifiers trained using the modified label definitions. 106
Table 18. Overall performance metrics for a ResNet50 classifier. 107
Table 19. Class-based evaluation metrics for an image segmented by the ResNet50 classifier. 107

Chapter 1: Introduction

1.1 Background and Research Motivation

Governments are responsible for maintaining serviceability of the aging road networks, but public finances are limited. Road reconstruction is drastically more expensive than an appropriate maintenance program (Ahmed et al. 2015). Monitoring road conditions can help target maintenance efforts and reduce the need to perform more serious road rehabilitation work.

Efforts to monitor road conditions, such as rating road surface roughness and cracking, have traditionally been done by engineering firms (Sayers 1998). They typically use specialized vehicles with high-precision sensors (see Section 2.2.4.1) to collect road condition data, but the vehicles and technicians that operate them are expensive. This can discourage governments and municipalities to perform frequent road condition monitoring or forego it altogether (Hjort et al. 2008).

In recent years, as smartphone usage has grown, so too has the popularity of using them as low-cost road condition measuring devices (Bhoraskar et al. 2012; Forslöf and Jones 2015; Souza et al. 2018). Much of that research used the smartphone's built-in accelerometer to measure its response to the road surface profile. Some work has also been done to identify specific defects, such as potholes (see Table 1), based on the acceleration response (Eriksson et al. 2008; Ngwangwa et al. 2010). Computer vision methods have also advanced in recent years, and they have been emerged as a popular approach to identifying defects on the road surface (Koch et al. 2015).

In particular, deep learning-based models have shown impressive defect identification results (Cao et al. 2020; Liu et al. 2020; Mei and Gül 2020). These models have been able to overcome some of the challenges faced by other approaches to asphalt defect detection, such as non-uniform lighting or global vs. local pixel intensity differentiation (Gopalakrishnan et al. 2017). Advances in deep learning methods are compounded by the proliferation of smartphones that can quickly and easily capture images of the road surface. This supplies more data for training the deep learning models. Even though the modern smartphones contain the sensors necessary to simultaneously collect motion data and visual data, there is a lack of research on methods of combining this data for performing road condition assessment.

There is an opportunity to combine the data generated by the different sensors in a smartphone, and to use that data to build useful reporting on the conditions of a road network. This hybrid approach, plus the ability to perform more frequent condition monitoring, using less expensive equipment and fewer labour hours represent a challenge to the traditional methods of data collection.

1.2 Research Objectives

The objective of this research is to describe road conditions using data collected by a smartphone while driving. The smartphone provides motion and image data from its onboard accelerometer and camera, respectively. With this data, an end user can view and interact with parts of it to interpret the condition of the road network. The research objective will be achieved by accomplishing the following:

- Use a smartphone to capture road condition data, including geotagged video and motion data, while driving
- Use the motion data to show how the roughness changes along a road
- Extract key frames from the video based on anomalies identified in the motion data
- Train a convolutional neural network to segment road defects in the extracted video frames
- Visualize road roughness and defect data in a format useful for road maintenance planners

1.3 Research Methodology

The methodology conducted to perform this research occurred in five steps. The first phase was data collection, which involved understanding the different sensors in the smartphone and how to set it up for optimally collecting motion data and video footage. The next step was preprocessing, which included any work done to modify the collected data before it is analyzed. Steps three and four were both for analyzing the data. Step three was concerned with how the motion data from the accelerometer can be used to describe overall road conditions and road defect locations. Step four was focused on analyzing the data from the video footage collected, mainly how to identify road defects in the frames that make up the video footage. Finally, in step five the motion data and video footage were integrated into a single map interface that an end user would interact with.

1.4 Thesis Organization

This thesis includes five chapters and an appendix. Chapter 1 has introduced the need for this research, its objectives and methodology. Next, Chapter 2 will provide background information on topics of road surface quality, smartphone-based vehicle telematics, computer vision, and pavement management systems. The literature review presents the state of research on these topics and informs the basis of the methodology. Chapter 3, the research methodology, describes the execution of work that has been conducted in pursuit of this research's objectives. The results of the work, and a discussion about the findings they have led to, make up the contents of Chapter 4. Conclusions and recommendations for future work are discussed in Chapter 5. The appendix contains the information relevant to the thesis deemed not suitable for the main body of the document, mostly large tables and computer codes.

Chapter 2: Literature Review

In their 2017 Infrastructure Report Card for roads, the American Society of Civil Engineers estimated that driving on roads in poor condition costs U.S. motorists, in aggregate, over $100 billion per year in related repairs and costs—about $500 per driver (ASCE 2017). Using taxes and tolls, governments levy additional costs to motorists for the maintenance and construction of roads. The cost of maintaining a road to an adequate level of quality can be a quarter of the cost of reconstruction (Ahmed et al. 2015). As infrastructure costs outstrip government funding, there is a strong incentive for reliable, inexpensive condition assessment methods to be developed (Hicks et al. 1999).These methods can help governments organize and prioritize their road repair burden so that funds and resources will be best appropriated. To understand how, this literature review will answer the following questions:

1. What are the features and condition characteristics of roads?
2. How is the condition of a road measured?
3. What technology can be used to collect road condition data?
4. What methods can be used to analyze that data?
5. And how can the analyzed data be used for asset management?

2.1 Road Structures and Surfaces

This section discusses the structure and materials used for paved roads. This will contextualize the types of pavement distresses (Section 2.1.3) and, later, how indirect sensor data can be used to describe road conditions (Section 2.3).

2.1.1 Types of Road Surfaces

Any established route that accommodates transportation between two or more places can be considered a road. Given such an ambiguous definition, it can be useful to classify roads based on their composition, which may vary drastically depending on their intended use. For example, the functional requirements of a transcontinental highway are vastly different than a temporary access road for a construction project. The material used for the road surface is dictated in part by how it will be used. There are a few ways to classify roads based on their surface type, as shown in Figure 1.

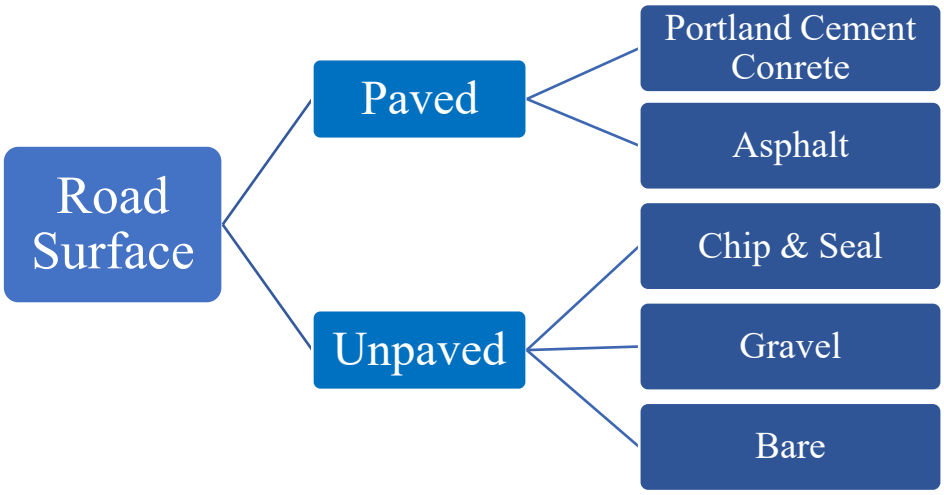

Figure 1. Types of road surfaces used for infrastructure

The surface can first be distinguished as either paved or unpaved. If it is paved, the surface layer aggregates are held together with a binding agent. This could be Portland cement paste or, in the case of asphalt, a bituminous material (Lay 2009). Paved road surfaces are the most rigid, but, due to construction and maintenance costs, they are also the most expensive.

Unpaved roads lack a chemically-bounder upper layer, making them more flexible and less expensive. Chip and seal unpaved roads, as well as gravel, still make use of a surface layer, but construction and maintenance costs are reduced. A bare unpaved road is the most flexible and the least expensive surface. But because an unpaved road surface is merely a graded (and sometimes compacted, too) subgrade surface, the frequency of maintenance required may be higher than other surfaces, depending on the amount of traffic and other factors (such as freeze thaw cycles).

2.1.1.1 Soil Types

The performance of the overlaying pavement is influenced by the conditions of the soil below. In general, the pavement will exhibit fewer signs of visible distress when the soil is mostly granular, as opposed to soils with higher plasticity (Yoder and Witczak 1991).

2.1.2 Road Anatomy

A well-performing road should be designed with three perspectives in mind: the structure, function, and maintenance of the road (Thompson and Visser 2000). First, consider the road

structure. Most paved roads in North America have a general cross-section that includes four distinct layers, as shown in Figure 2.

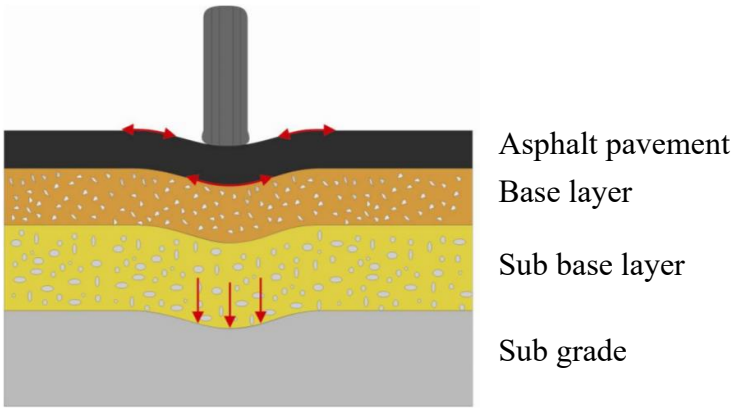

Figure 2. General anatomy of a road cross-section (Hjort et al. 2008)

Subgrade, the lowest layer of the road cross-section, is comprised of undisturbed soil or compacted material, including rock. The purpose of defining the subgrade layer during construction is to ensure that the above base and sub-base layers have a stiff foundation for their placement (Lay 2009). The layer above subgrade, the sub-base layer, provides a working layer for vehicles and equipment to travel over without disturbing the subgrade (Lay 2009). Once compacted, it also reduces moisture penetration into the subgrade, which further protects the road's foundation layer. Since it directly supports the asphalt surface layer, the base layer is the most structurally significant. It is typically made with crushed rock, which will exhibit some mechanical interlocking and thus improved bearing strength, especially after compaction (Lay 2009).

The surface layer of the road, whether it is flexible (e.g. asphalt pavement) or rigid (e.g. Portland cement concrete), has several roles other than carrying the vertical loads of the vehicles travelling on it. These roles include skid resistance, low light reflection, environmental resistance, and to act as a water barrier for the layers below (Rogers and Enright 2016).

As mentioned before, there are two general categories for road pavement types: flexible and rigid. Though there are neither perfectly flexible nor rigid pavement systems in practice. Every road structure can be placed somewhere on a flexible-rigid spectrum, between the two idealizations, based on the types of materials used for the surface and base courses of the road. The use of bitumen-based materials, such as asphalt, for the surface course will construct a flexible pavement road, while cementitious binders make for a road that behaves more rigidly (Rogers and Enright

2016). The differences in behaviour can be seen in Figure 3, where the flexible pavement will undergo more deformation compared to the rigid pavement, which distributes traffic-induced stresses more uniformly across the sub-layers. Identifying the layer structure and material properties for paved roads helps to uncover the causes of their defects/failures.

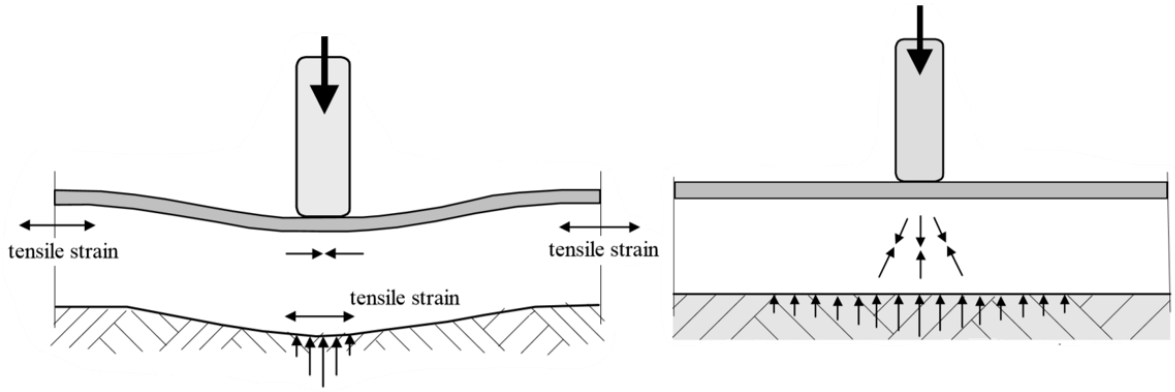

Figure 3. The idealization of flexible (left) and rigid (right) pavement structures (Lay 2009)

2.1.3 Pavement Distress

The poor performance of a paved road is often attributed to construction errors, material deficiencies, or inadequate maintenance strategies, as opposed to mistakes in the actual pavement design (Fwa 2005), and often there are more than a single contributing factor. Inconsistencies in pavement placement together with the traffic loading, which further interacts with changes in environmental conditions could cause defects in the pavement (Addis 2002). Poor drainage conditions and frost heave are recognized as two primary causes of pavement degradation (Yoder and Witczak 1991).

Pavement distress manifests itself in a variety of visual forms. This can make classifying road defects in the field difficult; however, many types of pavement distress are defined by their appearance (e.g. crocodile cracking, which is similar in appearance to the squared, scaly back of a crocodile). A report from the University of Wisconsin, Madison categorized all pavement distress types into four categories: surface defects, surface deformation, cracks, and repairs (Walker et al. 2002).

Each pavement distress category can be observed visually on the road surface, but not all of their causes are observable from the surface. One example of this is secondary rutting which is classified in the surface deformation category, and though it is induced by vehicle traffic, its root cause is an

insufficiently compacted subgrade layer (Hjort et al. 2008). Other types of pavement distress are named and described in Table 1.

Table 1: Types of pavement distress (Fwa 2005; Hjort et al. 2008; Walker et al. 2002)

Category	Name	Description
Surface Defects	Raveling	Loss of bituminous material in asphalt, mainly due to moisture-related stripping. The loss leads to a reduction in fine aggregates and exposure of coarse aggregates, some of which may also be lost. See Figure 4.
	Flushing	Excessive amounts of bituminous binder on the surface produces inadequate friction for tires, making the road surface dangerously slippery.
	Polishing	Smoothening of asphalt aggregates due to repeated wear from vehicular traffic.
	Delamination	Due to water entrapment or poor bonding during construction, the uppermost asphalt wearing course peels away.
Surface Deformation	Rutting	Vertical displacement channels in the longitudinal direction of the road under the common path of vehicle wheels. Depending on the road's age, the major factor contributing to rutting may be moisture and lack of compaction (new roads), or overloading from vehicles (older roads)*.
	Shoving	One or a few undulations that occur in the transverse direction at the road surface due to horizontal stresses that build up from vehicle acceleration and deceleration.
	Settling	Gentle depressions in the road surface caused by moisture build-up and/or compaction in base layers or subgrade.

	Frost Heave	Moisture build-up beneath the asphalt layer that, due to cold weather, expands when frozen and create bulges along the pavement.
Cracks	Longitudinal	Long cracks formed parallel to the direction of traffic. Usually caused by aged-asphalt hardening or differential settlement transverse to the road.
	Transverse	Cracks, which are perpendicular to the flow of traffic, usually caused by shrinkage and hardening as asphalt ages.
	Crocodile	Common in high traffic loading areas, these interconnected cracks are most often the result of fatigue in the asphalt layers and insufficient compaction in lower layers.
	Block	Like crocodile cracks, they are interconnected but differ by forming much larger blocks between cracks, usually close to right angles. They are caused by age-related fatigue, less so by traffic.
Repair	Pothole	Deep pockets in the road surface where asphalt layers have been removed, often due to freeze-thaw cycles, and the granular base layers begin to erode as well.
	Patches	Patches are used to smooth out the road where major defects have occurred; often placed over potholes, they can bulge up or sink below the top of road grade, which indicates distress.
	Edge Break	Irregularities in the road's edge of pavement, which make the sides of the road appear rough, are often caused by inadequate support at the road shoulder or insufficient asphalt placement.

*Primary rutting is due to deformation of upper asphalt layers, while secondary rutting is due to consolidation (compaction) of lower layers, particularly the subgrade.

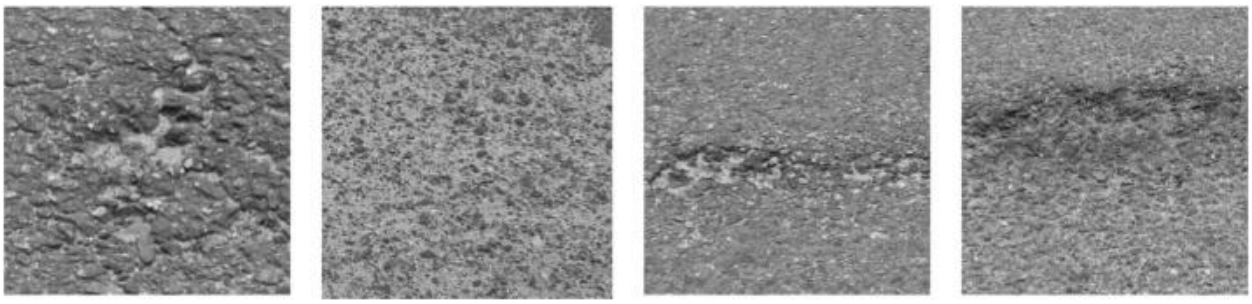

Figure 4. Examples of raveling on an asphalt road (Hoang 2019b)

Figure 5. Examples of potholes on paved roads

Though there are many types of pavement distresses, as shown in Table 1, there are only two main groups of conditions that lead to their formation. First, there are the environmental conditions that lead to pavement distress, such as moisture, freeze/thaw cycles, temperature, and time (Fwa 2005), and there are structural conditions that contribute the pavement distresses, including fatigue and overloading (Walker et al. 2002). These can also be seen in Table 1, where, for example, pavement rutting is caused by both a lack of drainage (environmental) and poor compaction (structural). In addition to the problems that pavement distresses indicate, such as poor drainage, they cause challenges for the road users as well.

2.2 Road Roughness

Road roughness affects more than passenger ride comfort. Excessive roughness also contributes to increased road noise, fuel consumption, tire wear, and other maintenance costs (Abulizi et al. 2016). The contents of this section will identify methods for conceptualizing, quantifying, and measuring roughness, which will enable the consideration of data collection and analysis procedures of Section 2.3.

2.2.1 Road Surface Quality

When considering road roughness and surface quality, describing a road as categorically rough is ambiguous. It could mean that the surface is regular but not smooth, such as with raveling (see Figure 6 left and Table 1). But calling a road "rough" could also mean that the surface is an irregular mix of smooth sections, potholes, ruts, and patches. Figure 6 shows the difference in what these two different types of roughness can mean, and how differentiating between them is valuable.

Figure 6. Road surface texture at different scales. Left: A rough surface with small-scale regularity; Right: An irregular road with a mix of roughness features at different scales

To address the ambiguity, Perttunen et al. (2011) break the problem of assessing road surface quality into two components based on the scale of the texture. There are microtextures, which refer to the small-scale roughness of the surface, and they primarily contribute to friction. Next, there are the macrotextures, which refer to the larger defects in the road, such as potholes, and how they affect road conditions.

The distinction between micro- and macrotextures enhances the definition of road surface texture, which refers to the vertical deviations from an ideal horizontal plane on a surface (Alauddin and Tighe 2008). Road texture is a function of the texture wavelengths and vertical amplitudes, as shown in Figure 7 (Cackler et al. 2006). The four classifications of surface texture are micro-, macro-, mega-textures, and roughness.

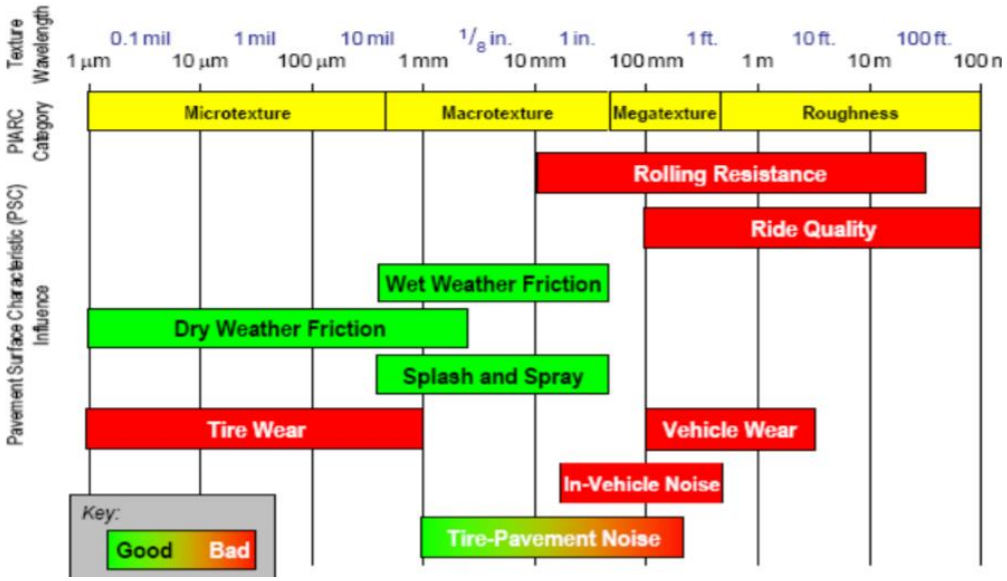

Figure 7. Basic elements of road texture and their impact on driving (Cackler et al. 2006)

The surface texture of the road interacts with vehicles in several different ways. Small scale textures in the road surface increase the friction in wet and dry conditions that, compared to a smooth surface, improve a vehicle's ability to brake, and thus increase road safety. This comes at the expense of tire wear, but it does not drastically increase fuel consumption at this smaller scale, unlike the larger textures that develop extra rolling resistance.

2.2.1.1 Road Failure

Roads are still able to function after they have failed; their failure refers to an inability to meet a sustained level of performance. Two types of road failure exist: structural failure and functional failure (Kyungwon et al. 2007). Structural failure occurs when the road cannot sustain the loads imposed by traffic, while functional failure is based on poor serviceability for the road users (Kyungwon et al. 2007). Excessive road roughness, which is uncomfortable for vehicle passengers and may accelerate vehicle wear, is a form of functional failure.

2.2.2 Quantifying Road Roughness

By quantifying the condition of the pavement, asset managers can better appropriate resources for road rehabilitation projects. The quantification of road roughness is typically done using a condition index. Since there are different road condition indexes and each use different rating methodologies, they make cross-index comparisons difficult. But the comparison of different road

sections across a single index is still useful within a pavement management system for assessing overall road network quality and allocating resources. The next subsections discuss some of the most common condition indexes used for pavement roughness assessment.

2.2.2.1 International Roughness Index

Based on experiments organized by the World Bank in 1982, the international roughness index (IRI) offered a way for different countries to resolve discrepancies between their roughness-measuring tools and methods (Sayers et al. 1986). The international roughness index is a slope value (e.g. m/km), derived from a series of observations, that conveys the irregularity of a road surface over some distance interval. More specifically, the IRI is a measure of the motion accumulated between the wheel and body of a vehicle, divided by the longitudinal road interval length, hence the slope units (Sayers et al. 1986). In Figure 8, this would be represented by the distance between the two masses, as described by Z_s and Z_u, compared to their original distance when at equilibrium.

Initially, the data used to generate the international roughness index (IRI) was from a longitudinal road profile (Sayers 1998). The elevation data points, which make up the road profile, are interpolated between each other and the accumulated motion of the vehicle is simulated using the quarter car model, a 2 degree of freedom (DOF) model for car suspension (Sayers et al. 1986).

The quarter car model, as shown in Figure 8, describes a single wheel's response as it drives over a surface. K_s and C_s represent the spring stiffness and damping coefficient of the car's suspension, respectively, while K_t represents the spring stiffness of the inflated tire; tire damping is neglected due to it being non-proportional (Popov and Geng 2005).

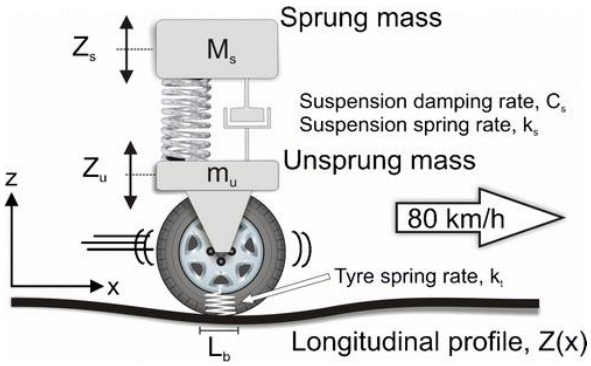

Figure 8. The quarter car model used in the International Roughness Index (Loizos and Plati 2008)

An IRI value of 0 m/km means that the road surface is perfectly flat. Technically, there is no upper-bound on IRI values, though the roughness expressed by an IRI value of 8 m/km would require that a vehicle significantly reduce its speed for safe passage over the road (Kyungwon et al. 2007).

2.2.2.2 Pavement Condition Index

The pavement condition index was first developed by the US Army Corps of Engineers primarily as a way to quantify the condition of the military's airfield runways (Campillo 2018). Unlike the IRI, the PCI is based on two measurable conditions: road smoothness and pavement defects. It has been shown that there is a limited correlation between IRI and PCI (Kyungwon et al. 2007).

The PCI is a numerical descriptor of the pavement condition on a scale of 0 – 100, where a low score represents poor road conditions, and a high score means conditions are good. An extension of the PCI is the pavement condition rating (PCR), which, depending on the PCI value, assigns a qualitative score to the road segment. The quality descriptors include failed, serious, poor, fair, satisfactory, and good (Koch et al. 2015).

According to the Ministry of Transportation Ontario, the PCI is made up of a roughness component and a distress component (Chong et al. 2016). The roughness component, based on a ride comfort rating, measures the functional performance of the road surface (i.e. how smoothly and comfortably does the road perform when used by transportation vehicles). The distress component, based on the distress manifestation index, refers to the structural performance of the pavement (i.e. have there been major defects, such as potholes, and where are they; how are they distributed along a road segment).

There are other indexes used to assess road roughness, such as the roughness defect score (RDS), but it is only suitable for assessing unpaved roads (Hugo et al. 2008).

2.2.3 Indexing Drawbacks

The primary concern with interpreting pavement index or ratings values is that they combine many factors present in the road surface into a single number. This can lead to misunderstandings of the road condition since it is not necessarily apparent what the cause of a poor score is. Similarly, the indexing method itself could misrepresent the true condition of the road, especially when the road condition data is collected automatically through indirect means. Road roughness data collection methods are described in Section 2.2.4.1.

2.2.4 Measuring Road Roughness

"Approximately 8 man-days of labour are required for each kilometer of roadway lane measured in both wheel tracks" (Sayers 1984). That labour quantity is based on the oldest approach to road roughness measurement: the use of a rod and level to survey the road profile, which is part of a broader category for roughness measurement, known as direct measurement (Sayers 1984). Direct measurement, in contrast to response-type measurement, is done with a profilometer that discretely measures the surface profile. Though this type of measurement has traditionally been done with rod and level equipment, modern profilometers use laser-based systems to measure the surface profile (Abulizi et al. 2016).

Wambold et al. (1981) classify roughness measurement devices into one of two categories: response-type and profilometers. The distinction is based on the indicator they are measuring. Devices that measure a dynamic mechanical response to the road roughness are referred to as response-type equipment. Devices that measure the road profile are categorized as profilometers. The smartphone accelerometer is a response-type device because the acceleration response does not come from direct contact with the road. Additional information on roughness measurement devices follows.

2.2.4.1 Types of Measuring Devices

There are four types of roughness measurement devices: direct profile, indirect profile, response-type, subjective rating panels (Abulizi et al. 2016). Device descriptions and examples of them can be found in Table 2. Additionally, in terms of cost, because their purpose is to measure the road

surface as accurately as possible, direct profiling devices can be the most expensive. The response-type road roughness measuring system (RTRRMS) devices, which measure the response of the vehicle or mounted equipment as it travels over the roads, represent a more affordable and accessible option for measuring road roughness. RTRRMS devices are typically calibrated with another device that approximates how the RTRRMS would score the pavement surface on an index, such as the IRI (Sayers 1984).

Table 2. Types of road roughness measuring devices

Device Type	Example	Description
Direct profile	Profilometer	Collects a set of elevation points to directly measure the road surface profile, though some interpolation is necessary to provide a detailed picture. Types of profilometers include rod and level, and laser profiler equipment.
Indirect profile	Accelerometer	When traveling at a constant speed and bearing, the vehicle should experience little acceleration beyond the vertical acceleration induced by road roughness and defects. The acceleration signal can be correlated to road roughness.
Response-type	Roadmeter	Collects data based on suspension stroking in a vehicle or specialized trailer. This is proportional to the accumulated deflection of the vehicle suspension, thus the data can be used to infer road roughness. The response-type road roughness measurement system (RTRRMS) device could be a passenger car or towed trailer with a roadmeter installed to capture the accumulate suspension deflection.
Subjective rating	Expert panel	Trained inspectors visually inspect the road for excessive roughness and surface defects which translates into a score, based on guidelines and experience, that they assign to the segment of road.

Many road asset managers use highspeed profilers to collect IRI data, but there is a trade-off between the accuracy of the data collected and the cost of owning and operating the machinery (Abulizi et al. 2016). The use of smartphone acceleration sensors can address the problem of expensiveness, and with proper methods of analysis, the acceleration data can provide information that is accurate enough to make decisions within the pavement management system (Wahlström et al. 2017).

The activity of road surface monitoring requires mobility, because the deployment of static sensors cannot produce the same data as a mobile sensor array (Mednis et al. 2011). Modern mobile road profiling equipment, as shown below in Figure 9, have been developed to incorporate an array of sensitive and expensive sensors. They are expensive systems with starting prices of around £500,000 (Radopoulou and Brilakis 2017).

Figure 9. A road surface profiler (Shamsabadi 2019)

2.3 Smartphone-based Vehicle Telematics

Due to the availability of low-cost sensors, such as 3-axis accelerometers in smartphones, the ability to collect data correlated to road roughness has expanded (Douangphachanh and Oneyama 2013b; Du et al. 2014; Hanson et al. 2014). The process of collecting sensor data, filtering the data, and using numerical or machine learning methods to classify road roughness is referred to as indirect profile classification (Gorges et al. 2019). They can be used to contrast or complement

another type of roughness assessment, namely vision-based methods, which use algorithms to assess road conditions or detect defects based on image and video data (Koch et al. 2015). Vision-based methods for road condition assessment are discussed in Section 2.4.1.

The nature of the modern smartphones makes them a suitable device for telematics: most smartphones house 10 or more sensors and can transmit data over cellular networks (Souza et al. 2018). The use of smartphone sensors for collecting data relevant to driving conditions is a relatively new area of research, which approximately began after the launch of the original iPhone in 2007 (Mohan et al. 2008a). Up until the smartphone era, when cellphones became ubiquitous and full of sensors, the main use of a smartphone in research was to complement data collection, and purpose-built external sensors were used for primary data collection (Eriksson et al. 2008). Pothole Patrol, an early research project that used a smartphone for data collection, only used the phone for collecting GPS information, while the acceleration data was collected by external sensors fixed to the vehicle (Eriksson et al. 2008). Other research at the time made use of the smartphone's ability to collect triaxial acceleration data, and even used the phone's microphone for a novel honk detection feature (Mohan et al. 2008a).

The main reason to use a smartphone as the primary device for collecting data is the ease of deployment (Wahlström et al. 2017). A smartphone user can conduct data collection, analysis, transmission, and others all with a single device. This reduces the need for data acquisition equipment or personal computers while conducting experiments. Additionally, the development cycle for smartphones (though, not necessarily smartphone sensors) is much shorter than that of vehicles, which can mean that smartphones more quickly adopt new technology (Wahlström et al. 2017).

2.3.1 Types of Sensors

Broadly speaking, of all the different types of sensors housed inside a smartphone, they can be categorized into one of three groups based on their usage (Souza et al. 2018). They include:

1. *Motion*—Sensors that relate the local relative position and movement of the device, such as accelerometers and gyroscopes.

2. *Environmental*—Sensors that collect information about the devices surrounding ambient conditions, such as thermometers, barometers, and ambient light sensors (e.g. the iPhone uses a photodiode for its ambient light sensor).
3. *Position*—Sensors that receive information about the device's position globally, such as GNSS receivers and magnetometers.

Another way to characterize the types of sensors used by smartphones is based on where the data being collected is in relation to the smartphone. From Wahlström et al. (2017):

1. *Exteroceptive*—Acquiring external information. An example is a GNSS receiver that combines cellular and WiFi signals for accurate device location (usually 10m with 95% confidence, recorded at 1Hz (Zandbergen 2009)). Other examples include a magnetometer for headings or cameras and microphones for video and audio recordings.
2. *Proprioceptive*—Acquiring internal information. Examples include internal thermometers or the in-device accelerometers and gyroscopes that describe the device's state of motion.

2.3.1.1 Global Positioning

Smartphones are able to position themselves on a map by combining several types of technology. The use of a global navigation satellite system (GNSS, colloquially referred to as GPS) receiver is the most common, but the positioning data is enhanced with both cellular-localization and a server-side check on the device (Eriksson et al. 2008). Cellular-localization pings nearby cell towers to approximate location. A server-side check will approximate location based on nearby available WiFi networks, which can be important for positioning in dense urban areas where tall buildings obstruct satellite signals and can cause multipath errors. Eriksson et al. (2008) ran multiple trials to test the accuracy of their GPS device and found it to be 3.3 meters.

2.3.1.2 Triaxial Accelerometer

Smartphone acceleration sensors capture acceleration from motion and gravity, known as dynamic and static acceleration (Harikrishnan and Gopi 2017). The acceleration information is captured on a silicon chip that detects slight movements in a mass attached to the housing. This excites a capacitor whose current flowing through it can be correlated to the acceleration. Figure 10 shows the output of a smartphone's accelerometer. Based on the axes assigned to the device, the phone

would have been in a landscape orientation during the data collection for the blue series to be averaging negative 1 g.

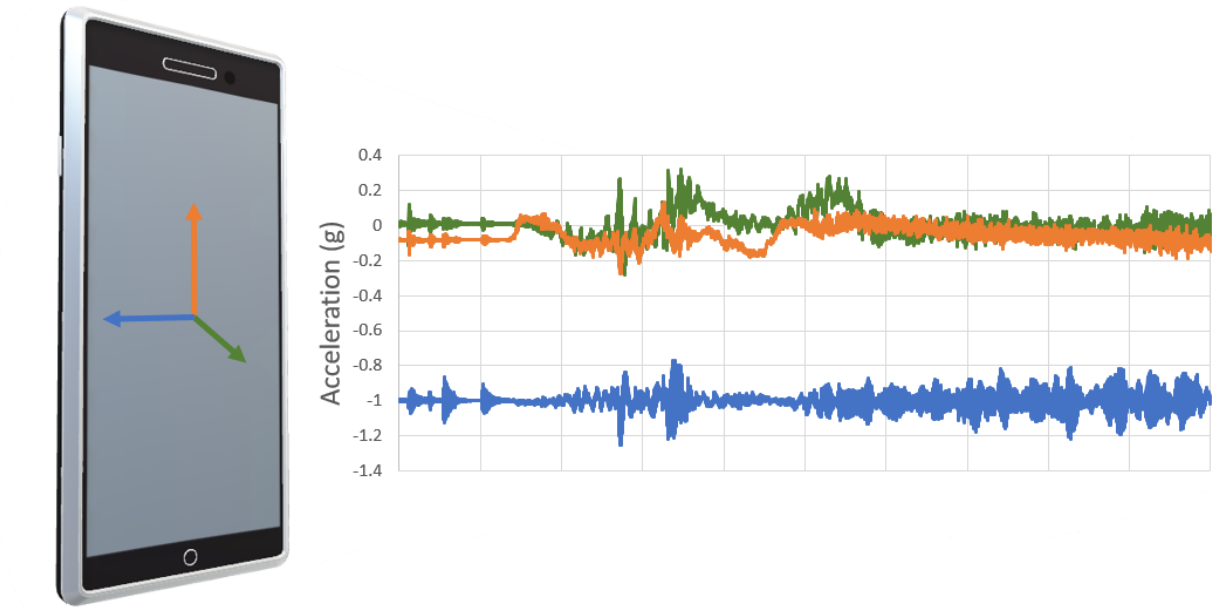

Figure 10. Triaxial acceleration measurements from a smartphone

Perttunen et al. (2011) and Harikrishnan and Gopi (2017) reported that the tri-axial accelerometer signal is useful when the car is affected unevenly, such as when only half of the vehicle travels over a pothole, compared to the entire vehicle traveling over a speed bump (Harikrishnan and Gopi 2017; Perttunen et al. 2011).

2.3.2 Drawbacks of Smartphone-based Data Collection

There are some drawbacks to using smartphone sensors, as opposed to purpose-built sensing equipment. Because the smartphone sensors are considered features of the device, rather than the core of the device, the quality of the sensors may be reduced for economic reasons. A reduction in sensor quality can lead to misreading and a lower sampling resolution; most smartphone accelerometers have a maximum sampling frequency of 100 Hz, but purpose-built accelerometers can capture data at a temporal resolution several times greater, such as with the Pothole Patrol, where acceleration data was collected at 380 Hz (Eriksson et al. 2008).

Other issues include the potential for battery drain and challenges associated with collecting data from a device that is not fixed in place, but the inclusion of gyroscopic sensors makes the disorientation problem addressable by, for example, using Euler angles (Mohan et al. 2008a). Additionally, smartphones will never supplant in-vehicle sensors because the in-vehicle sensors relay a significant amount of information related to vehicle subsystems, which smartphone sensors could not feasibly pick up on.

2.3.3 Data Utilization

A review of the literature exposes two approaches to utilizing acceleration data: (1) Anomaly detection and (2) roughness assessment. Though their objectives differ, the general method for achieving them is similar: data is acquired, then cleaned and processed before it is used for a predictive model, which can be used to enhance road maintenance planning. This process is shown in Figure 11 and is the subject of the remainder of Section 2.3.

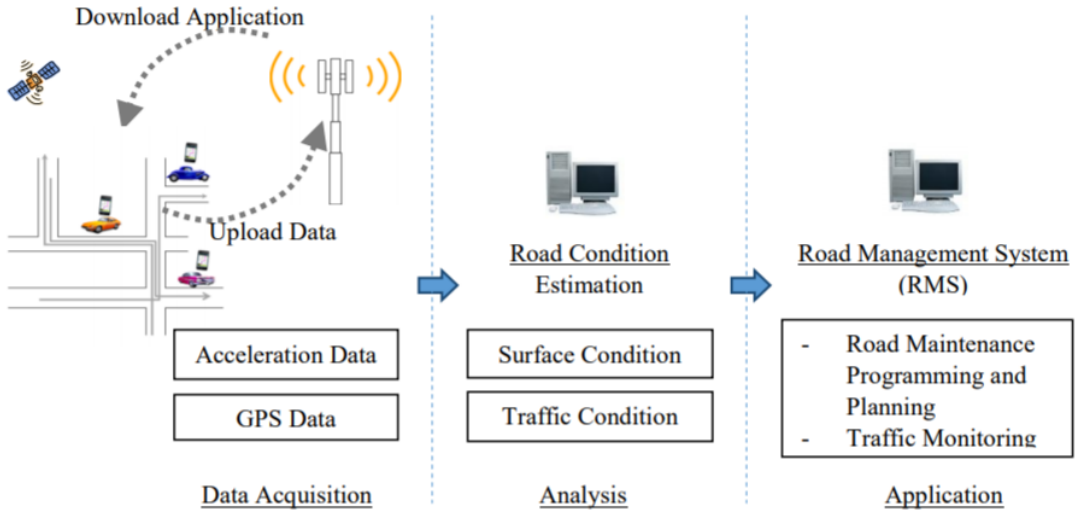

Figure 11. Acquire, Analyze, Apply—the approach to using smartphone data for a PMS (Douangphachanh and Oneyama 2013b)

2.3.3.1 Set-Up/Data Collection

Properly setting up the equipment used to collect data relevant to road roughness is important because it will improve measurement consistency during collection, increasing the probability that trends can be found in the data through multiple trial runs (Hanson et al. 2014).

An equipment setup adequate for collecting roughness data would include an accelerometer and GPS logger, plus a motor vehicle. That combination of equipment will be able to collect the device-experienced acceleration, its location, and a timestamp. The main consideration for device placement is where there will be the least interference in the acceleration signal. For example, to reduce the influence of engine vibration, Du et al. (2016) placed their accelerometers at the rear of the vehicle. Convenience of placement is also an important factor. The most common locations for mounting accelerometers were either the vehicle dashboard or the windshield, as presented in Figure 12 (Douangphachanh and Oneyama 2013b; Hanson et al. 2014; Perttunen et al. 2011; Souza et al. 2018; Yeganeh et al. 2019).

Figure 12. Left: windshield-mounted smartphone (Souza et al. 2018); Right: dashboard- and windshield-mounted phones (Yeganeh et al. 2019)

The configuration of equipment is also important. The primary means of configuring equipment for acceleration response is the data collection frequency, typically measured in Hertz (Hz), which in this case refers to how many samples are collected per second. (e.g. 1 Hz would mean that 1 acceleration reading is taken each second.) Internal smartphone GPS reads at 1 Hz.

The upper limit of sampling frequency on most smartphone accelerometers is 100 Hz (Mednis et al. 2011). With a higher sampling rate, more bandwidth is required for transmission and there is a greater storage requirement. This leads researchers to configure the accelerometers used in their experiments based on the objective of the experiment. For roughness indexing on a highway, the sampling frequency can be as low as 20 Hz and still produce meaningful results (Hanson et al.

2014). For anomaly detection on a city street, a lower sampling frequency could miss reading responses to road anomalies and the sampling frequency should be higher (Mohan et al. 2008a).

2.3.3.2 Preprocessing

The preprocessing phase includes all activities that are required to augment, modify, or clean the raw data before it is analyzed for results. Three common activities for preprocessing are filtering, windowing, and feature extraction.

2.3.3.2.1 Filtering

In signal processing, a digital filter is any procedure that removes part of the signal (Han 2010). For road roughness assessment with a smartphone device, only acceleration data collected while the vehicle is in motion is desired, so a filter to remove instances collected when the speed is zero is desirable (Tai et al. 2010).

But there are also other considerations. Engine vibration can be detected by the smartphone device, depending on its placement, and a low-pass filter could be used to attenuate the high frequencies contributed by engine vibration. However, the mechanical vibration of the vehicle also occurs in higher frequencies, and thus the use of a low-pass filter could be counterproductive. High pass filters (typically Butterworth), those that pass by high frequencies and reduce the influence of lower ones, have been used to reduce the influence of static acceleration (namely, gravity) since it is expressed in the lower frequencies (Cabral et al. 2018; Tai et al. 2010).

Other filters can be used for denoising, such as the Kalman filter for outlier removal of GPS data (Perttunen et al. 2011). A filter that attenuates small accelerations in the signal based on their magnitude relative to the peak accelerations in the signal has been used, as shown in Figure 13 (Harikrishnan and Gopi 2017).

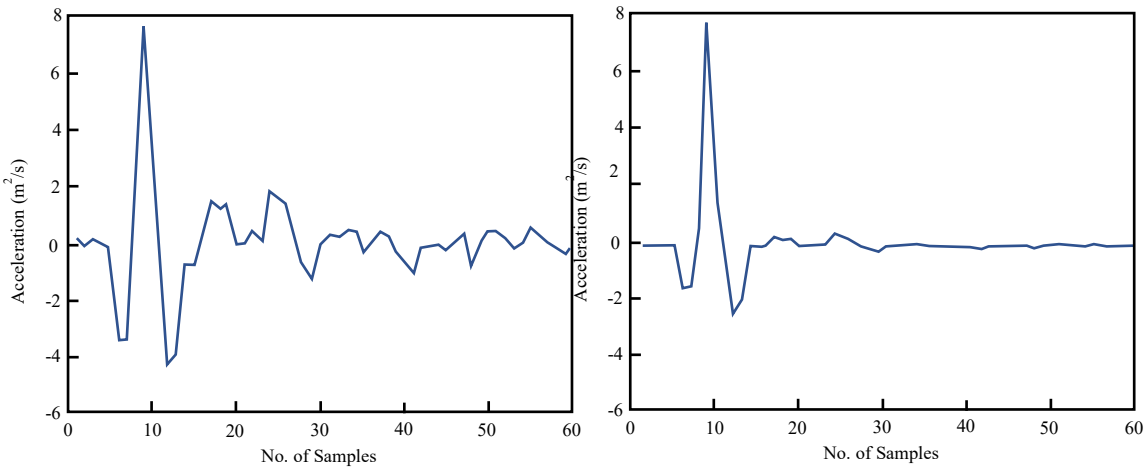

Figure 13. Acceleration before (left) and after (right) the signal has passed through a maximum absolute value filter (Harikrishnan and Gopi 2017)

2.3.3.2.2 Segmentation & Windowing

Compared to the length of a stretch of road that a driver may travel over, the size of a pothole or other defect is quite small, and the scale (texture) of what makes the driver experience an unpleasant ride is even smaller. So, in order to more easily identify road anomalies, or to comprehensively evaluate the roughness of a road, the acceleration-response signal is often discretized into much smaller pieces.

There are two approaches to the signal discretization: segmentation and windowing. Segmentation cuts the data into *n* segments of a length determined by the researchers. Each segment is analyzed independently. Windowing cuts the data into segments, but with a slide that must be less than the window length. A common window-slide split is a 2-second window with a 0.5-second slide (Cabral et al. 2018; Perttunen et al. 2011). For a 10-second signal, there would be 5 sections when using the 2-second segmentation approach. But there would be over three times as many windows of analysis as the segmentation approach while using the 2-second window with a 0.5-second slide. This means that many more segments may be analyzed for the same signal when using a windowing approach as opposed to segmentation.

Due to the reduced number of frames of analysis, one advantage of segmentation over windowing is the reduced computation burden. An advantage of windowing over segmentation is the more

thorough analysis potential since there is a little chance of an anomaly that straddles a segment boundary being missed during the analysis phase.

Segmentation is acceptable for roughness indexing, while windowing would be a better approach for anomaly detection. This can be observed in the literature: only the research efforts on anomaly detection used a windowing approach, such as Cabral et al. (2018) and Perttunen et al. (2011), while the segmentation was the most common approach for roughness indexing (Chang et al. 2009).

2.3.3.2.3 Feature Extraction

It is uncommon for raw data to be input into a machine learning (ML) algorithm. Typically, new variables, known as features, are derived from the raw dataset and are then fed into an ML algorithm. This process is called feature extraction; it has been said that it is more important than choosing the right predictive algorithm (Feffer 2017).

Though feature extraction is a large field of study, with important applications in image processing, the process is not necessarily complicated. Many researchers extract basic statistical features (e.g. mean, standard deviation, etc.), which are distinctive enough to train and validate a model (Cabral et al. 2018; Perttunen et al. 2011). Threshold-based features can also be collected, such as the number of instances that the acceleration spikes above/below a certain value, but the choice of what the threshold value should be is challenging and can create a less flexible model.

Sensor data is collected in the time domain, yielding temporal statistics and features, but the signal collected from the sensor can be converted into the frequency domain, usually using the Fast Fourier Transform (FFT) algorithm (Cabral et al. 2018; Douangphachanh and Oneyama 2013b; Perttunen et al. 2011; Souza et al. 2017).

Deep learning, with multiple convolutional layers, can sidestep the feature extraction step, but the deep learning process comes with a substantial computing overhead (Strubell et al. 2019). The most efficient process would involve detailed feature analysis to find what features best serve a predictive algorithm.

2.3.3.3 Analysis

Within the literature, there are two main objectives for analyzing the data collected by accelerometers mounted to vehicles. The first objective is roughness assessment, which is an

approximate measure of how far a stretch of road is from being a smooth planar surface. The second objective is anomaly detection, which seeks to identify potholes, patches, and other road surface defects, and then geotag their location.

2.3.3.3.1 Roughness Classification & Indexing

The work of roughness assessment comes in two forms. There is roughness classification, where the road roughness data is used to score the road surface into one of a set of established roughness categories. The simplest example of this would be binary roughness classification: the road surface is considered either rough or smooth (Souza et al. 2017). Alternatively, roughness indexing is an approach that will attempt to correlate the roughness data with an index, usually the International Roughness Index. Table 3 chronologically summarizes roughness classification and indexing attempts that have been published.

Table 3. Roughness classification and indexing research

Year	Researchers	Contributions
2013a	Douangphachanh and Oneyama	Correlated frequency domain magnitudes against IRI values from an external source to show the correlation between the two. Correlation is highly dependent on speed.
2013b	Douangphachanh and Oneyama	Similar to previous work (2013a), but with smartphone placement in more realistic locations, such as a shirt pocket.
2014	Douangphachanh and Oneyama	Collect more acceleration data from smartphones fixed in place on vehicle, apply a filter to reduce the influence of turning and braking, create a linear model base on their 2013a reference data.
2014	Hanson et al.	Using 3rd party software, compare smartphone acceleration data to data collected from a highspeed profiler on a 1 km highway. For 100-meter sections, the two methods provide similar results.
2015	Jang et al.	Program the phones to only collect data when a threshold is reached. Data is transmitted to a server where a model classifies it as either rough, smooth, or as an impulse event.

2016	Du et al.	Using two dedicated acceleration sensors, their system collects data and, using the half-car model, and a power spectral density function to calculate the IRI.
2017	Souza et al.	Classify road segments as either smooth or irregular after testing a variety of supervised learning algorithms on four different feature vectors.
2018	Souza	Classification approach to the road being: regular or deteriorated; cobblestone or dirt; pothole or speedhump. Evaluated various distance measures in time series, such as dynamic time warping.
2018	Souza et al.	Based on previous work, showcase the capability of a road condition evaluation system, Asfault, that rates the smoothness quality of a road surface on a five-point scale.
2019	Yeganeh et al.	Compare the driver-reported experience of road surface quality to the IRI value approximated from smartphone acceleration data via a linear equation with the RMS of the acceleration data.

Across the research, each study used a different approach to the features extracted for classification. In general, there are three types of parameters used to describe roughness as it is observed in the acceleration-response. (1) Amplitude parameters relate to the strength of the signal's acceleration value, seen by the vertical amplitude in a graphical representation, and are often measured with reference to the temporal average of the signal (Gadelmawla et al. 2002). (2) Spacing parameters relate the horizontal characteristics of the signal, often using thresholds, such as the number of peaks above a point. (3) Finally, there are hybrid parameters that are calculated by the combination of amplitude and spacing parameters, such as the steepness factor (Yeganeh et al. 2019).

2.3.3.3.2 Anomaly Detection

Anomaly detection using an accelerometer in a single direction (vertical) of motion alone is challenging. A major obstacle to correctly identify an anomaly is to establish a "ground truth" (Mohan et al. 2008b). This refers to the set of indicators that can be used to consistently and

correctly identify an anomaly on the road surface, such as a pothole. Perttunen et al. (2011) utilized video to look back and label their acceleration signals. The features associated with anomalies in those segments of the data could be used to train a machine-learning model, which would then be able to classify future anomalies. The most widely used classification tool is the support vector machine method (Eriksson et al. 2008; Perttunen et al. 2011; Tai et al. 2010). Other approaches and models used for anomaly detection are presented in Table 4.

Table 4. Acceleration-based road anomaly detection research

Year	Researchers	Contributions
2007	De Zoysa et al.	A system proposal to fit public buses with accelerometers and other sensors to measure road conditions. By including GPS in their system, they suggest that a bus having to slow down at locations that are not bus stops may indicate poor road conditions. Buses as the condition monitoring vehicle are problematic because the effect that weight distribution can have for acceleration response.
2008	Eriksson et al.	The Pothole Patrol's detection algorithm is based on speed, z- and x-axis acceleration thresholds which are compared to a labeled set of known pothole acceleration instances. They use several filters to strip other event classes (e.g. Speed bumps and railway crossings). Detection is primarily based on a z-axis threshold and an acceleration-speed ratio.
2008a	Mohan et al.	Using smartphone sensors for a combination of brake detection, traffic assessment, honk detection, and brake detection. The system assumes the smartphone device is held in the user's pocket, not mounted to the vehicle, thus making the reorientation of the device necessary.
2008b	Mohan et al.	The chosen anomaly detection is a bump in the road. Both convex (speed hump) and concave (pothole) bumps are considered, but they are not distinguished. Two bump detecting approaches, based on the

		vehicle speed, are used to classify the acceleration signal. Both are threshold-based, but the low-speed approach requires a sustained dip in acceleration (20ms; 0.8g), while the higher speed approach is triggered by short spikes in the signal.
2010	Tai et al.	Acceleration data is collected from a motorcycle-mounted device. Anomalies are labeled by the driver at the time of impact, and segments of the data series are later displayed as histograms to display the distribution of accelerations in the time surrounding the vehicle driving over a pothole. In the histogram approach, as window size increases so does accuracy. However, the localization of the anomalies suffers.
2011	Mednis et al.	Real-time pothole detection using a difference- and threshold-based detection algorithms. Z-DIFF, the difference-based algorithm identifies large differences in successive acceleration values, indicating that there is a sudden increase. Post-processing on the acceleration data is also done with a standard deviation algorithm to reduce false positives in pothole detection.
2011	Perttunen et al.	Focusing on man-made anomalies (eg. speed bumps), the researchers use a combination of temporal features (eg. acceleration variance) and spectral features based on the assumption that anomalies would produce lower frequency components compared to vibration from the motor and road surface. These features were used to train an SVM for anomaly classification.
2019	Nguyen et al.	“Lightweight” algorithms are proposed to detect anomalies, mostly potholes. The system also includes and fault exclusion component to reduce false positive from user actions. Detection is done with previously discussed algorithms, including Z-DIFF. It is enhanced using the Grubb method, a test for outliers is a univariate dataset.

2019	Kyriakou et al.	With a smartphone mounted to a car, the researchers collect data and look in the data for the most significant variables based on regression analysis. Of the 31 variables collected, the most significant were the forward acceleration, lateral acceleration, roll, and pitch. Since they do not directly indicate anomalies, the variables are used in an artificial neural network (ANN) which outputs a binary classification: pothole or no defect.

2.3.4 Challenges

There are several challenges with the acceleration-based approach, such as how to consistently detect anomalies at precise locations and how to label acceleration anomalies in the time domain (Tai et al. 2010). Perttunen et al. (2011) utilized video to label their acceleration signals. They plotted the acceleration signal along with speed, superimposed over the spectrogram, and identified spikes and dips in acceleration that matched with events identified in the recorded videos. This is a time consuming and imperfect approach since it requires a lot of manual review and judgement.

Earlier work focused on detecting anomalies, such as speed bumps or potholes, using threshold-based methods (Harikrishnan and Gopi 2017). Basic threshold methods are easily implemented, but they can lack the flexibility to distinguish acceleration responses collected under different conditions. Vehicle speed and mounting position of the smartphone have been shown to enhance or diminish the signal (Douangphachanh and Oneyama 2013b).

2.3.4.1 Effects of Speed

Douangphachanh and Oneyama (2013a) study showed a linear correlation between IRI values and summed frequency magnitudes from an acceleration signal over the same stretch of road. But the correlation is not consistent, due to the effects of speed. Low speed produced better results in some research (Douang and Oneyama 2013a), but it was too low in some others (Du et al. 2016). The effect of speed can make a large difference in the acceleration response value while traveling over a pothole or a speed bump (Mohan et al. 2008b).

A speed-adjusted threshold can be effective in reducing the variability in acceleration responses over the same anomaly but at different speeds (Harikrishnan and Gopi 2017). Others have attempted to remove the speed dependence of features altogether (Perttunen et al. 2011).

2.4 Computer Vision

Humans are "very good at visually detecting patterns", but to program a computer to recognize the same patterns is a very difficult problem (Berndt and Clifford 1994). Despite the challenges, the field of computer vision has expanded in recent years, and the capabilities of computer vision-based techniques have advanced a great deal since Berndt and Clifford described the problem of computer vision.

Computer vision is a subfield of artificial intelligence. The typical pipeline of a computer vision-based system takes in raw image signals, then processes and analyzes them in order to provide computational insight to the user. Whereas the previously discussed sensor-based (acceleration) assessment techniques infer the road condition by measuring surface qualities, vision-based techniques make surface observations of the road to identify defects and anomalies (Koch et al. 2015). This section describes what research studies have been done to create vision-based systems that could identify pavement defects and distress.

2.4.1 Visual Surface Analysis

In general, visual assessment of pavement condition can be described as automatic, semi-automatic, or manual, based on the amount of human interaction involved with the system (Koch et al. 2015). Surface analysis of pavement happens broadly in three stages (Yousaf et al. 2018). They are (1) pavement data collection, where spatial information is collected for (2) distress identification and (3) distress assessment. By relying on equipment, such as cameras and laser scanners, data collection is the easiest stage of surface analysis to automate. Distress identification, where road defects are recognized, and distress assessment, where they are categorized based on the subjects of Table 1, are more challenging to automate. But if the data collection is thorough and consistent, the opportunity to use computer vision-based methods can supplement the traditional approach to defect assessment using trained experts (Ahmed et al. 2011).

2.4.1.1 Challenges to Automating Surface Analysis

Despite the large opportunities that computer vision provides, there are challenges in both the capability and accuracy of such systems. Cheng and Myojim (1998) identified three problems to overcome with computer vision-based distress detection on paved surfaces:

1. Special devices (cameras, lasers, etc.) are required
2. Systems have low processing speeds, leading to low accuracy
3. The systems can only identify a few kinds of distress

Advances in technology, both in hardware and software, have mostly addressed the first and second challenges. Challenges still remain for creating a system that is flexible enough to function in different environments without losing capability.

Computer vision-based approaches to pavement condition assessment may also be constrained by the computing burden, storage costs, and privacy concerns (Wahlström et al. 2017). It is also possible that a computer vision framework—in finding correlations in large datasets—will plateau in the progress being made for activities such as classification (Hartnett 2019). In addition, there are four types of variability problems specific to surface analysis while using cameras for data collection:

1. *Surface*—differences in the appearance of paved surfaces, which may be the result of different asphalt admixtures or rates of degradation.
2. *Lighting*—unlike an indoor environment, the road surface lighting comes from the sun and affected by the surrounding elements (e.g., shadows), and cannot be controlled. Changing shadows lead to apparent differences in road defects depending on the time of day.
3. *Environmental*—image and video capture of the road surface can be affected by environmental conditions, such as debris from trees or precipitation.
4. *Set up*—it is important to keep cameras or laser scanners positioned in the same orientation and at the same distance from the ground when mounted to vehicles for data collection. Inconsistencies can produce uneven lighting and band patterns that could add an extra step for noise removal in preprocessing.

2.4.2 Image Preprocessing

Preprocessing activities modify raw data that has been collected to be analyzed later on. Previously, raw data referred to the acceleration signal, but here it is the digital image frames captured by a camera. There are several popular preprocessing techniques, including:

- *Resizing*—if images come from a variety of sources, they may also be of different sizes, as measured by the number of height and width pixels. Resizing of all images to fit one size simplifies later operations performed on the images.
- *Colour transforming*—digital images are represented numerically as a set of planes that ascribe a red, green, and blue (RGB) value to each pixel. Transforming an image to greyscale would have the image represented by only one plane, which will speed up processing times.
- *Denoising*—image noise comes in various forms, but the objective of denoising for any image is to create an estimation of the true, undisturbed image by suppressing contamination in the image.

Two other processing techniques, segmentation and morphology operations, are discussed in more detail below.

2.4.2.1 Image Segmentation

Early work on the topic of automatic image-based defect detection (as opposed to manual review) was done with a thresholding approach to image preprocessing (Cheng and Miyojim 1998; Koutsopoulos and Downey 1993). Considering the capabilities of computer hardware at the time, this approach makes sense: the threshold approach converts an image into a black and white form based on identifying each pixel as above or below a pixel intensity threshold.

In image processing, segmentation activities refer to any steps taken to partition an image into multiple sections. These sections can be labeled and grouped such that all the sections that share a label will also share other relevant characteristics. This makes segmentation methods useful for detecting objects and boundaries in a frame. If there is no prior knowledge about the objects contained in the image, it is considered to be bottom-up segmentation.

Types of segmentation include region growing and clustering (e.g. k-means), or semantic segmentation using a convolutional neural network (CNN). Thresholding is the simplest form of

image segmentation. It can produce a binary image, as shown in Figure 14 (right), based on each pixel intensity value being above or below a defined threshold. Binary images contain unwanted noise and texture, as can be observed in the speckled region surrounding the pothole on the right side of Figure 14.

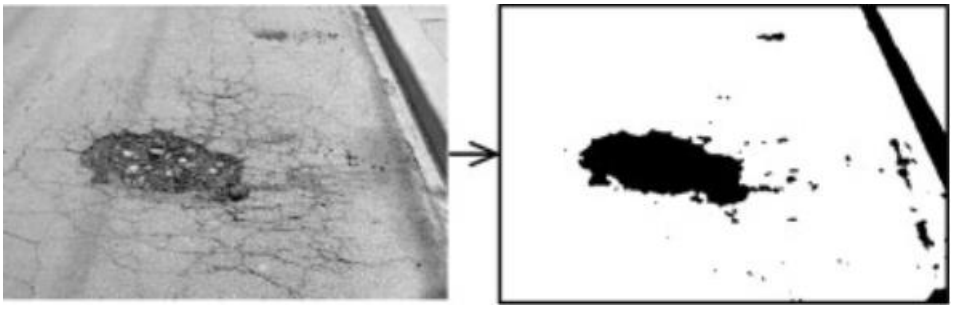

Figure 14. Threshold-based image segmentation for a pothole (Koch et al. 2011)

2.4.2.2 Morphological Operations

Morphology is the study of form. In computer vision, morphological operations process images based on the contained shapes. The operations, which are normally performed on binary images, can modify individual pixels based on their similarity or difference to nearby neighbouring pixels. Thus, the operations are more sensitive to the relative ordering of pixel values, rather than their magnitude.

The combined advantage of using both thresholding and morphology processing on an image is shown in Figure 15. The objective is to discern between the background (the sky, in purple) and the foreground (cars, buildings, street, in yellow). The threshold does a good job of categorizing most information, based on the sky being lighter than the foreground. But it is imperfect, also including the bright traffic light. The morphological operations are able to remove the small purple areas from the second image because most of their surrounding pixels are yellow. This leaves behind the two main sky regions as purple.

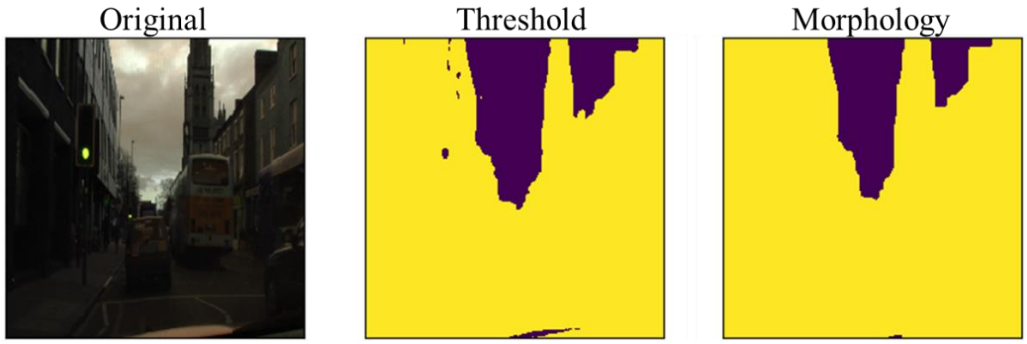

Figure 15. Identifying the sky in an image using two image processing operations (Canuma 2018)

The two main morphological operations are erosion and dilation. Erosion adjusts pixel values based on the minimums of its neighbors, while dilation works in the opposite manner. The two operations may be performed sequentially (erosion, then dilation) to denoise and then enhance the image. These operations have also been used preceding an image filter (e.g. Gaussian filter) to remove shadows cast on images of pavement (Zou et al. 2012).

2.4.3 Image Features

Feature descriptors reduce the amount of information input for training and using models. There are several kinds, such as histogram of oriented gradients (HOG), scale invariant feature transform (SIFT), and speed-up robust feature (SURF).

The HOG feature descriptor, commonly used for object detection, calculates where edges are located in an image by measuring the difference between neighbouring pixels, and then determine the overall direction of the change, thus estimating the gradient. HOG is related to SIFT, another commonly employed feature descriptor. The algorithm considers the information contained within the image with respect to a Cartesian coordinate system and at different scales. Once the features are identified, as shown in Figure 16, the features could be compared to a training set of images using a Euclidean distance match for nearest neighbours (Treiber 2010). Further steps can be taken in the SIFT technique to minimize the effect of outliers and mitigate false positives.

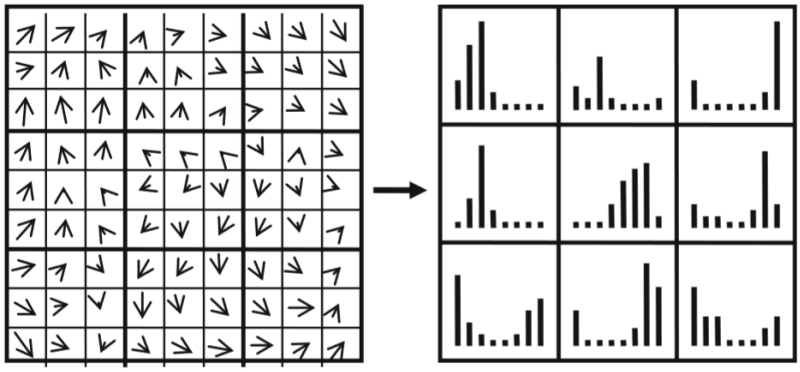

Figure 16: Gradient orientations (left) and their distributions (right) for the SIFT feature descriptor (Treiber 2010)

2.4.3.1 Features of Cracks

According to Zhang et al. (2017), there are five features to base a pavement crack detection model:

1. *Intensity*—Crack pixels typically have lower intensity than regular road surface pixels surrounding them. This was also described by Cheng and Miyojim (1998), who showed that under their threshold-based approach, distresses pixels had lower pixel light intensities in video frames.
2. *Spatial Distribution*—pavement cracks are represented in images as thin regions of lower intensity pixels.
3. *Statistics*—the histogram of pixel intensities for a cracked road surface may have a slight second mode at the lower end (left side) due to the relatively small collection of similarly lower intensity distressed pixels.
4. *Geometric*—the high aspect ratio of crack shapes is one of their defining features.
5. *Orientation*—the orientation of several connected cracks can be identified as distinct from other features.

Gopalakrishnan et al. (2017) used a deep convolutional neural network (DCNN) as a feature generator for pavement images. The network itself is trained on semantically labeled training images created in a manual process.

2.4.3.2 Pothole Features

The key feature used for identifying potholes is their relative darkness compared to their surrounding area. Part of that darkness comes from the shadows cast inside the pothole, which can

produce inconsistencies in image collection, but can also be identified quickly for real-time identification (Nienaber et al. 2015). In addition, they will also have a different texture: either coarser in dry conditions or smoother in wet conditions if water collects in the depression (Yousaf et al. 2018). In addition, they often have a roughly elliptical shape (see Figure 14), which can be exploited with morphological operations.

2.4.4 Defect Detection

Once an image is processed and features are extracted, it can be analyzed to detect pavement distress and defects, which can then be classified. Classification is the process of examining a feature vector and identifying it with a known entity (Koutsopoulos and Downey 1993). Though not all research studies followed a straightforward detect-then-classify approach to identifying pavement distresses. Due to the wide variety of pavement distress types and forms, it is more common for work to be done in addressing a single type of distress. Distress types and methods used to identify them are listed in Table 5 for some of the reviewed literature. It should be noted that due to the emergence of convolutional neural networks in the last few years, most studies have used these methods for pavement defect recognition in the last three years.

Table 5. Detection methods used for various pavement defects

Year	Researchers	Defect	Detection Method
2020	Cao et al.	Multiple	Convolutional NN with multi-stage object detectors
2020	Mei and Gül	Cracks	Convolutional NN with Connectivity Maps
2020	Liu et al.	Cracks	YOLO algorithm with convolutional NN
2019a	Hoang	Patches	Support Vector Machine and 1st order statistics
2019b	Hoang	Raveling	Convolutional NN and co-occurrence matrix
2019	Soloviev et al.	Cracks	Convolutional neural network
2018	Yousaf et al.	Potholes	Least squares support vector machine
2018	Hadjidemetriou et al.	Patches	Support vector machine with co-occurrence matrix
2017	Gopalakrishnan et al.	Cracks	NN, SVM, logistic regression; NN performed best

2017	Zhang et al.	Cracks	Series of thresholds based on crack properties
2017	Ouma and Hahn	Potholes	Fuzzy c-means clustering to recognize ROI
2017	Radopou and Brilakis	Multiple	Semantic texton forest decision tree
2016	Hadjidemetriou et al.	Patches	Support vector machine
2015	Nienaber et al.	Potholes	Segmentation and edge detection algorithm
2012	Zou et al.	Cracks	Segmentation-based probability map
2011	Koch and Brilakis	Potholes	Recognition filters by threshold, shape, and texture
2009	Nguyen et al.	Cracks	Backpropagation NN based on statistical features

The main challenge in detection of pavement distress comes from the trade-off between the quantity and quality of image data acquired for training and validating a model. Some researchers took the approach of collecting thousands of images, but sacrificed some control over their composition (Nienaber et al. 2015; Zhang et al. 2017). Others took more specifically framed but less practical photographs (Zou et al. 2012).

One solution proposed to improve the quality of images was that a powerful illumination system should be incorporated to provide consistent lighting (Zakeri et al. 2017). Though the more recently adopted convolutional neural networks, which are more flexible than the past methods, perform well on image-based tasks and reduce the need for consistent lighting (Hoang 2019a; Soloviev et al. 2019). But despite their potential depth, the convolutional neural networks do not eliminate the challenges of significant variability in the appearances of pavement distresses.

2.5 Pavement Management Systems

At its most basic level, the reason for implementing a pavement management system (PMS) is to improve asset management of road infrastructure. Asset management techniques are employed by the PMS, which aims at maintaining road surface infrastructure. Pavement management systems integrate data from different aspects of asset management, such as engineering, financial, and regulatory information, so that accurate condition assessments lead to optimal prioritization of road maintenance and rehabilitation (Bektas et al. 2014). In other words, a pavement management

system houses and analyzes information about infrastructure networks to enable asset managers to make decisions for resource allocation.

The PMS originated in the latter half of the 20th century when priorities shifted from construction of new roads to maintenance of existing roads (Terzi 2007). Early pavement management systems lacked any condition forecasting or economic analysis between preventative and deferred maintenance strategies. These systems were used to prioritize road projects based on factors including traffic loads and current pavement conditions (Kulkarni and Miller 2003).

2.5.1 Components of a PMS

The essential components of a pavement management system are: (1) a pavement information database, (2) a prioritization model, and (3) a decision model (Bandara and Gunaratne 2001). More specific subcomponents of a PMS are shown in Figure 17; they enable the PMS to comprehensively manage the road assets. Economic assessment, for example, is a function of historical costs, the current governmental financial outlook, and the state of the existing road network (Kulkarni and Miller 2003). Combining this information allows planners to decide what resources will be allocated to different segments of the road network.

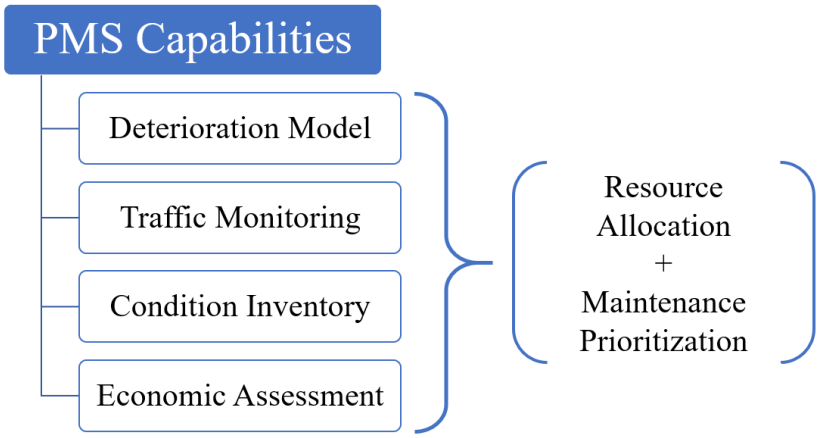

Figure 17. Functions of a pavement management system

2.5.2 Implementing a PMS

A pavement management system is required to systematically schedule maintenance in a timely manner to improve the overall pavement network condition (Campillo 2018). For many municipalities, its implementation could represent a significant technological enhancement to its

asset management program, but it has costs, too, that should be weighed against the benefits (Amekudzi and Attoh-Okine 1996).

2.5.2.1 Benefits of Implementing a PMS

Coordination—Additional assets can be managed in conjunction with roads and highways (e.g., A road reconstruction can occur at the same time as its underlying pipe or cable infrastructure is upgraded).

Optimization—With detailed monitoring, roads can be maintained or reconstructed closer to the point where their serviceability falls below the acceptable level.

Cost Reduction—Combining data collection, scheduling, historical information, and other elements, the PMS can help decision makers identify cost savings where they otherwise would have missed them due to data siloing. Also, time is saved, which improves maintenance costs.

2.5.2.2 Costs of Implementing a PMS

Change—whether it is a bespoke system (expensive) for a large municipality or a generic system (cheaper, but less catered), the upfront burden of implementing a PMS is a drag on wider spread adoption of the systems.

Users—the PMS is meant to facilitate and enhance the work of decision-makers, but they must be trained to use the software effectively. Additionally, users who may be collecting data or incorporating GIS features may also need training.

2.5.3 PMS Software

There are many available software implementations for a PMS. Most involve graphical user interface (GUI) that layers information collected in the field (e.g. pavement distress images) on top of a map of the city, which is based on GIS data (Wolters et al. 2011). This can be seen in Figure 18, which includes a satellite base map, overlaid with a colour-coded road network, and images taken by road inspectors at key points. In this case, the PMS software is enhancing the asset manager's ability to assess the condition of the road network digitally.

Figure 18. Example of a PMS user interface (Shamsabadi 2019)

Chapter 3: Research Methodology

The objective of this research is to create tools for understanding the condition inventory of roads in a pavement management system. This chapter explains how to capture and analyze the data required to create that condition inventory. Figure 19 shows an overview of the data collection and analysis process discussed in this section. With a smartphone mounted to the windshield of a vehicle, the vehicle collects video footage and motion data as it drives throughout the city. The data is split based on two uses: for assessing the roughness of the roads (see Section 3.3.1) and for identifying defects at various locations (see Section 3.3.2). Once processed and analyzed, the data is displayed in a condition inventory map that would be an added feature to a pavement management system.

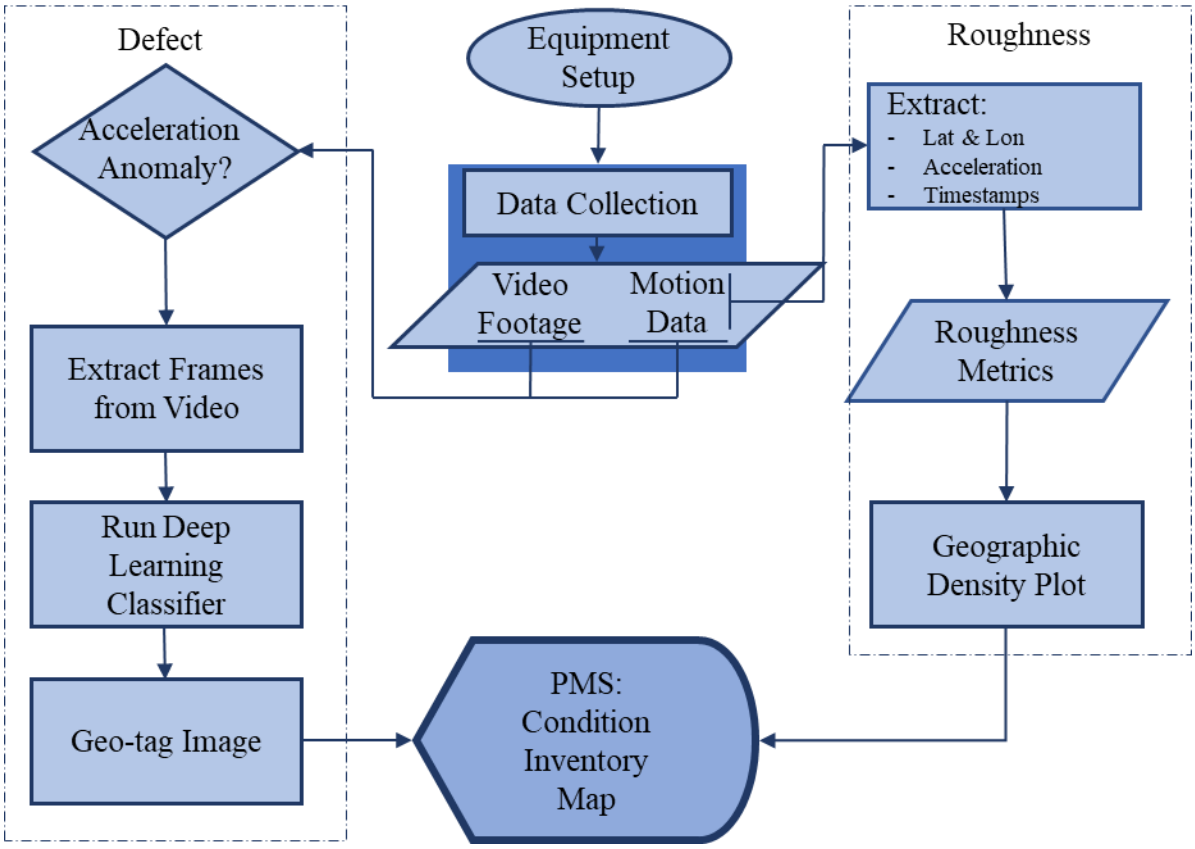

Figure 19. Overview of the pavement condition assessment system, from data collection to display

3.1 Data Collection

There can be a trade-off between the quality of data and the ease of collection in the data collection process. Decisions for how to collect data for this research tended toward equipment and

procedures that prioritize flexibility and ease of use. In part, that is due to the quality and low-cost of sensing equipment available today. But it is also due to the nature of this research's objective: to create a reliable inventory of the paved road network's condition. For these purposes, it is more important to know where the road is rough and where there are defects, rather than attempting to precisely quantify the size and severity of defects.

3.1.1 Equipment & Setup

One of the primary advantages of the system developed for this research, as opposed to more expensive road-profilers, is the low cost of its components (not including a vehicle). The equipment used in this research for data collection include:

1. 2006 Lexus RX400h
2. 2016 iPhone 7 Plus
3. Windshield phone mount

A key aspect of the data collection process is that a single device—the iPhone—captures all of the data. Because all of the data is collected by a single device, and because the quality of the data is, in part, dependant on the position of the phone inside the vehicle, ensuring the device is mounted consistently for every trial run is important. To that end, we mounted the iPhone on the windshield in a centered position, as shown below in Figure 20. The positioning of the windshield mount and phone is based on maximizing the view that the phone's camera would have over the single lane that the vehicle would be traveling in. The orientation of the phone, in a landscape, allows for the recorded video to be captured in a wide-frame, i.e., landscape format, which is more conducive for capturing the relevant video data.

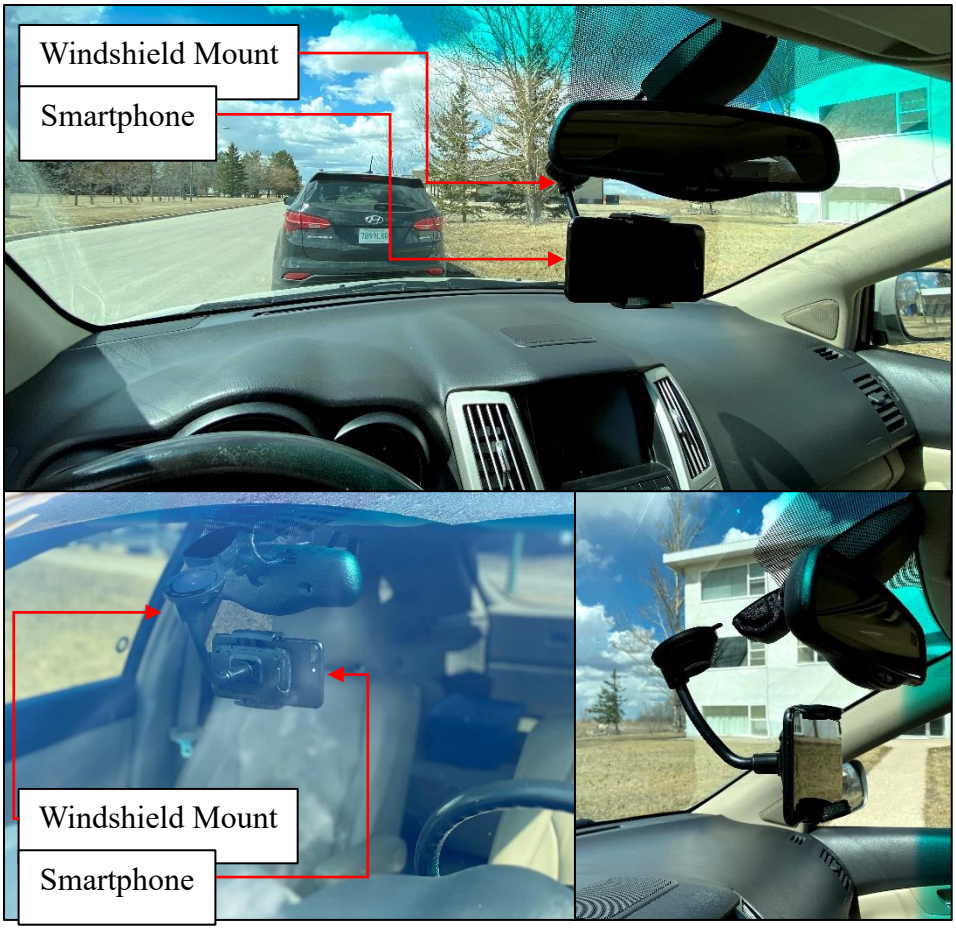

Figure 20. Equipment setup used for data collection

3.1.1.1 Sensors

The data collected by the phone will go toward describing the condition of the road in two ways. The first is in terms of perceived motion (i.e., how would a road user feel as the drive over the road). This data is best captured from two of the phone's onboard sensors: its triaxial accelerometer and its gyroscope. The accelerometer's data is read as a g-force in three orthogonal directions. It measures the sum of acceleration from gravity and the phone's own acceleration. Thus, when the phone is lying flat, the accelerometer will output -1g in the Z-direction. Similarly, the gyroscope expresses rotational movement about the three orthogonal axes. With this combination of data from the two sensors, the phone's motion can be described with 6 degrees of freedom, as shown in Figure 21. This allows for the device to differentiate between gravity and user acceleration. Section 3.3 will outline how this data can be used to estimate the road roughness and identify road anomalies.

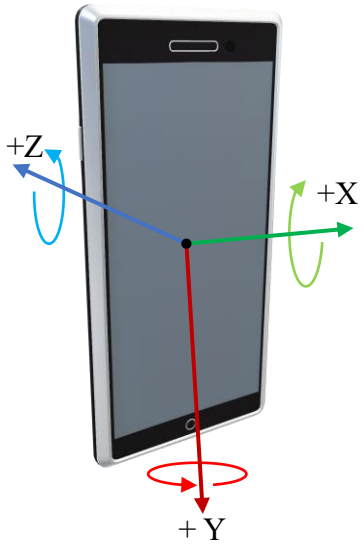

Figure 21. The six degrees of freedom captured by the phone's motion sensors

While the accelerometer and gyroscope contribute to the phone's acute, local sense of motion, the phone's GPS antenna provides its sense of motion globally. This provides the latitude and longitude coordinates of the phone, accurate to within five meters, which can be overlaid on a base map to display the route driven by the vehicle. This allows for the estimated roughness to be localized to a city's road network.

In addition to a sense of motion, the second way that the road's condition can be described is in a visual sense (i.e., how does the condition of the road appear). Visual data is collected by the phone's camera, which collects footage as the vehicle drives throughout the road network. The video footage can then be processed to identify defects in the road, and these defects can be tagged with the coordinates collected by the GPS antenna.

Understanding the performance of each sensor is helpful for interpreting the data each of them produce. Here, the key performance metrics for the sensors can be thought of as their sample rate and precision. The sample rate, a frequency measure, refers to the number of data readings the sensor can output per unit time, and precision being the level of detail that can be relied on in the data.

Table 6. Sensors plus their frequency and precision

Sensor	Frequency	Precision
Accelerometer	100 Hz	0.02 g
Gyroscope	100 Hz	0.01 rad/s
GPS	1 Hz	± 5m
Camera	30 fps	1920×1080p

3.1.1.2 Software

Data is collected using two iOS apps, SensorLog and Timestamp Camera Basic. The SensorLog app produces a comma-separated-values (CSV) files with the output of all sensors on the phone and the corresponding time stamp. SensorLog accesses the raw data output by the various sensors on the iPhone through Apple's Core Motion framework. The data reported by Core Motion are related to the device's motion and environmental conditions, including from the accelerometer and gyroscopic sensors listed in Section 3.1.1.1. Other values come from the onboard magnetometer and barometer. In total, SensorLog collects 79 values about the device and writes them to a CSV file. Table 7 provides a summary of the data types recorded by the SensorLog app. The cells are headed and organized based on an activity and a sensor. Each list item is an output value, and "X,Y,Z" refers to three outputs in the three orthogonal directions related to the phone (see Figure 21).

Table 7. Summary of outputs of the SensorLog iOS app

Location – GPS	Activity – Accelerometer
- Latitude/Longitude - Altitude (m) - Speed (m/s) - H/V Acc. (m)	- Pedometer steps (No.) - Pace (m/s) - Distance (m) - Floors ascended/descended (No.)
Motion – Accelerometer / Gyroscope	Pressure – Barometer
- X,Y,Z accelerations (g) - Gyroscope X,Y,Z (rad/s)	- Altitude (m) - Pressure (kPa)
Magnetic – Magnetometer	System – Internal Specifications

- Induction X,Y,Z (μT) - Yaw (rad) - Roll (rad) - Pitch (rad)	- Logging time (hh:mm:ss.SSS) - Device ID - Battery level (%) - Audio Power peak/avg (dB)

Video data is collected using Timestamp Camera Basic, as opposed to the iPhone's default camera app, because Timestamp Camera is capable of overlaying several metrics associated with the video at the time each frame is captured. The most important data overlay is the timestamp for each captured frame, because that can be referenced against the acceleration timestamp output by SensorLog, which allows for the video footage and the motion data to be synchronized with one another. A frame of all possible data overlays for a video taken with the Timestamp Camera is provided in Figure 22. For the videos captured for this research, only the time, in hh:mm:ss.SS format is overlaid on the video, as shown in the top-left corner of Figure 24.

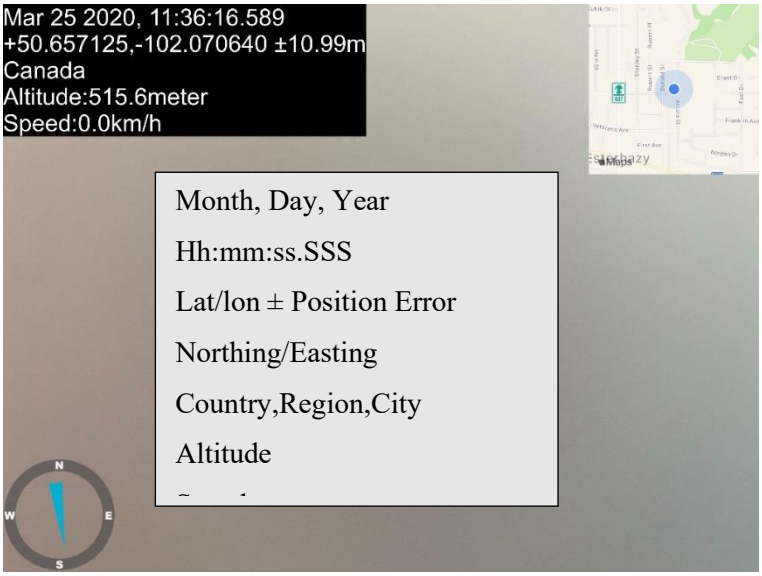

Figure 22. Example of viewfinder and data provided by Timestamp Camera app

3.1.2 Driving Routes

The types of roads covered during the data collection phase were a mix of highway and urban. Figure 23 shows all of the driving routes taken throughout the City of Thunder Bay. It shows the 105 kilometers traversed for the data collection. The split between road types is 84.8 kilometers of urban travel and 20.2 kilometers on a highway.

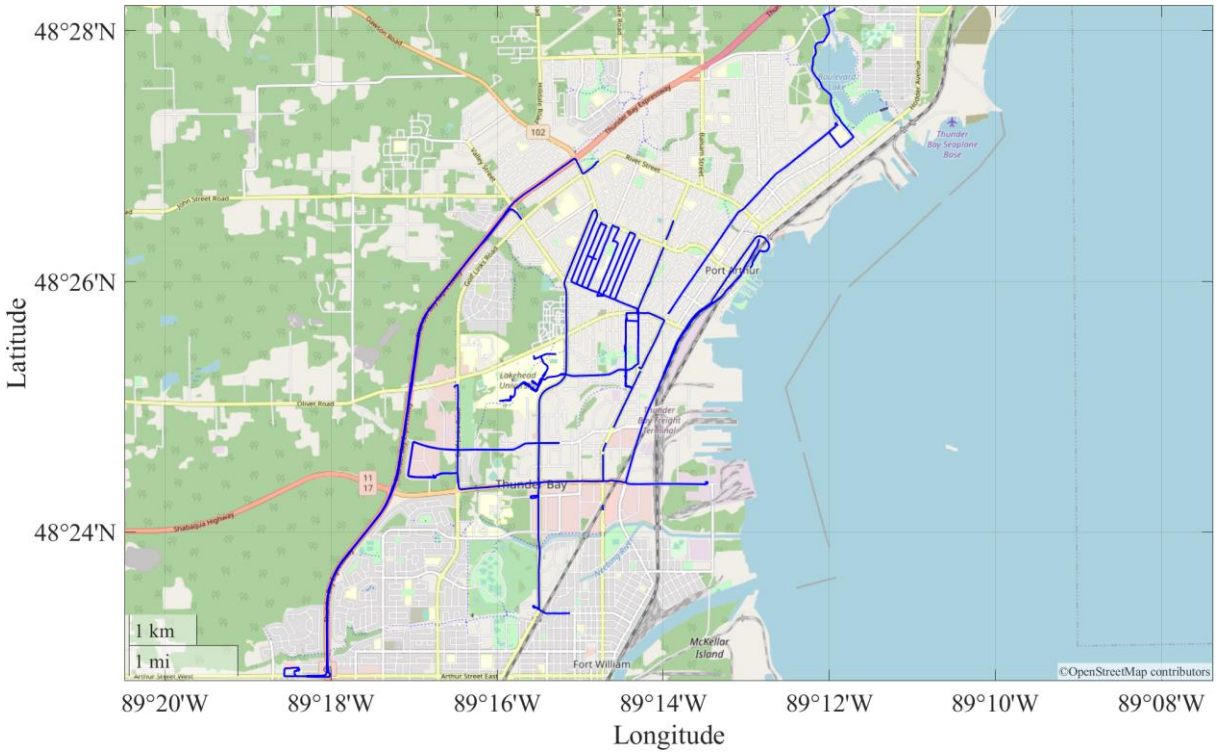

Figure 23. Data collection driving routes in Thunder Bay

3.2 Preprocessing

With a large set of data collected, including over 20GB of video footage and 1.5GB of CSV files, the preprocessing phase can begin. The two objectives of preprocessing this data are to (1) reduce the file sizes of the data and (2) reduce the dimensionality of the data; often these objectives are met in conjunction. Reducing the dimension of the feature space will yield smaller files that can be processed more quickly and consume less storage. Different preprocessing activities are conducted based on the type of the data (e.g., sensor logs or video footage).

3.2.1 Motion Data Preprocessing

As was shown in Table 7, there are 79 values output by the SensorLog app and they are collected at a frequency between 1 and 100 times per second. Only a few are necessary for interpreting the motion of the vehicle and relating to the roughness of the road, so the first step in preprocessing the motion data is feature elimination. This is done by removing variables that have little or no impact on extracting useful information from the data. For example, knowing the battery level of the phone does not materially improve the ability to assess the condition of the road surface.

This process lowers the number of variables from 79 down to 14, as shown in Table 8. These remaining variables are deemed most relevant for describing the overall roughness of the road and identifying possible defects. The loggingTime is critical for synchronizing the data with the video footage, since both the loggingTime and the timestamp on the video footage (see Figure 22) come from the smartphones internal system time. The other times, gpsTime, accTime, and gyrTime, which are associated with the GPS, accelerometer, and gyroscope, respectively, were kept in case of any discrepancies between the loggingTime and the GPS, or acceleration, or gyroscopic sensor readings.

Table 8. Variables remaining after feature elimination plus their description

Variable	Example Value	Description
loggingTime	03:08:25.6	Time the values are written to the CSV file
gpsTime	1.57E+09	Times since 1970, seconds
lat	48.40768	WGS84
lon	-89.2795	WGS84
alt	214.4297	Meters
speed	7.08965	Meters/second
accTime	325793.9	Times since last device reboot, seconds
accX	0.265625	G
accY	-0.65627	G
accZ	-0.70103	G
gyrTime	325793.9	Times since last device reboot, seconds
gyrX	0.071587	Rad/s
gyrY	-0.33898	Rad/s
gyrZ	-0.00059	Rad/s

3.2.2 Video Preprocessing

For this research, the video files themselves are not preprocessed in any way. Manipulating video files is extremely computationally intensive and, here, would not add any significant value to the data or procedures. Data synchronization, as discussed in Section 3.5.1, is an easier process for matching the motion data to the video, and it removes any requirement for editing the video file.

This means that the videos are scanned for frames that may contain road defects, and these extracted frames are then subject to some preprocessing.

3.2.3 Image Preprocessing

The image preprocessing done here was minimal and simple. Cropping was the only preprocessing activity used, and the preprocessed images were the ones used for training the classifiers discussed in Section 3.4.2. Otherwise, to save on storage and processing requirements, the frames collected from the videos should be cropped as small as possible, but still maintain enough information that can be seen by a reviewer at a later time. Images are cropped to a size of 780 pixels wide by 230 high. Originally, they were cropped to a size of 800 wide by 250 high, but these values were reduced because it allowed for two images to be held in the video memory of the NVIDIA GPU instead of one, thus improving the speed at which deep learning can occur.

The area identified in Figure 24 is the region of interest (ROI). It contains all of the information relevant to the road condition while discarding the surroundings. The minimum horizontal ROI bound, L_{min}, can be approximated as a function of the vehicle's width and the camera's field of view (FOV) in the horizontal plane. The field of view refers to the area that is captured by the camera's image sensor based on the horizontal and vertical angle projected out from the center of the camera to observe that area.

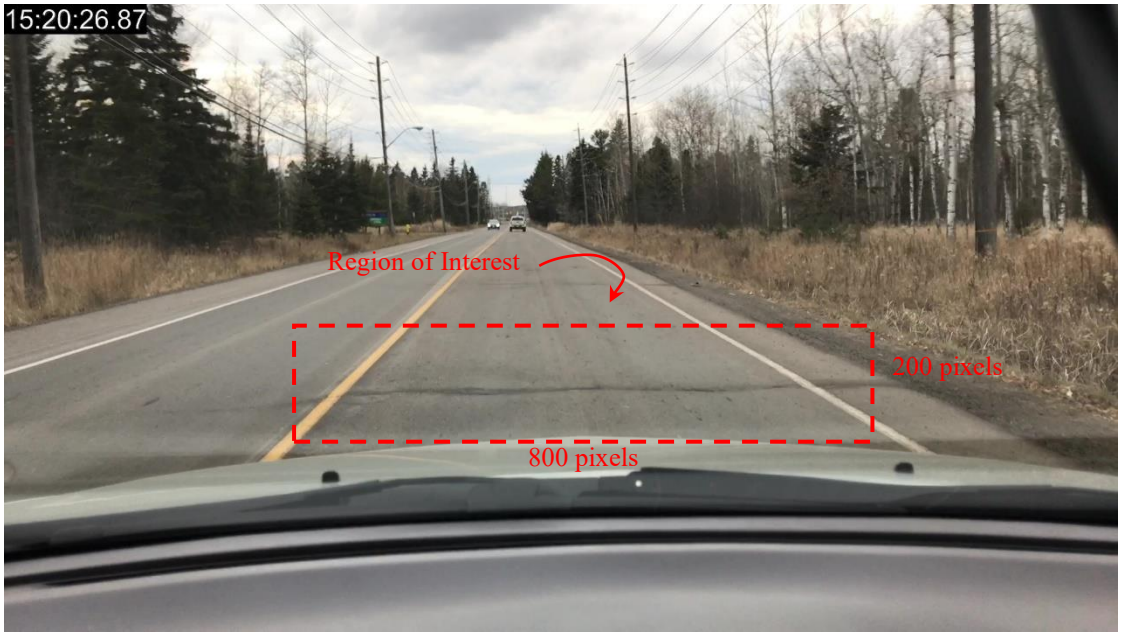

Figure 24: Region of interest within a video frame captured during testing

3.3 Motion Analysis Process

There are two objectives for analyzing the captured motion data. The first is to describe the roughness of the road over its entire length of travel. This has been done using statistics from the acceleration time series and presenting it on a map of the route that was travelled. The second objective is to identify key points in the road that likely include significant defects. Identifying road defects is especially important because it ties into the vision-based component of this system, which is discussed in Section 3.4.

3.3.1 Roughness Metrics

Quantifying road roughness with the equipment setup used in this work is challenging, because the data collector (the smartphone) is physically distant from the vehicle wheels. After the wheels, the forces from driving over the road surface are transmitted through the shock absorbers, the vehicle body, and then windshield mount (shown in Figure 20) before affecting the smartphone itself. The main challenge is that the collected motion data (vertical acceleration, in this research) is influenced by the parts between the smartphone and the wheels. For example, shock absorbers are designed to dampen the effects of bumps in the road so the driver is more comfortable. Because of this, trying to produce official roughness values, such as the IRI, requires a lot of extra processing steps, despite there being clear patterns in the unprocessed acceleration data. This led

to the decision to avoid trying to produce official roughness values, and instead focus on describing the acceleration data as it is.

The different sampling frequencies for the smartphone's onboard sensors are listed in Table 6. For roughness metrics, the two relevant readings are the GPS and accelerometer outputs. Because the number of acceleration data points outnumber the GPS data by a factor of 100, some work must be done to either condense the number of acceleration data points or expand the number of GPS latitude-longitude pairs. The approach taken, shown in Figure 25, is to condense the number of acceleration data points. The main reason for taking this approach is that, ultimately, there should be a visual display (see Figure 26) to interpret the road roughness, so the GPS data should be preserved in its original form.

What is needed to condense the acceleration data down to match the GPS data is a single metric that can be computed on windows of the acceleration data that are 100 data points long. There are several basic descriptive statistics to try, such as the mean, median, or variance. The mean and median do not give a good indication of roughness because the acceleration has a consistent average of about -1g. Even in segments of the time series where there are large peaks, indicating excess acceleration from a bumpy road, there are roughly as many high peaks above and below the average value, which cancels out the effect of the large spikes in acceleration readings. The same is true for the median. The variance metric, i.e., the sum of squared differences between the mean and each of the 100 acceleration values, fairs better (see Section 4.2). But the metric that showed the best representation of the condensed acceleration time series was the absolute value of the difference between the minimum and maximum acceleration values in the window.

The top plot in Figure 25 shows the window in red, plus the maximum and minimum values. Directly below, in the lower plot, is a single data point, the MMR (min max range). This shows how the original acceleration time series, which in Figure 25 is about 9200 values long, is reduced to the lower plot, made up of only 92 data points. The only exceptions are at the very start and end of the acceleration data, where there may be fewer than 100 values before the next GPS reading would trigger a new window. In these cases, the window is made smaller to accommodate this difference.

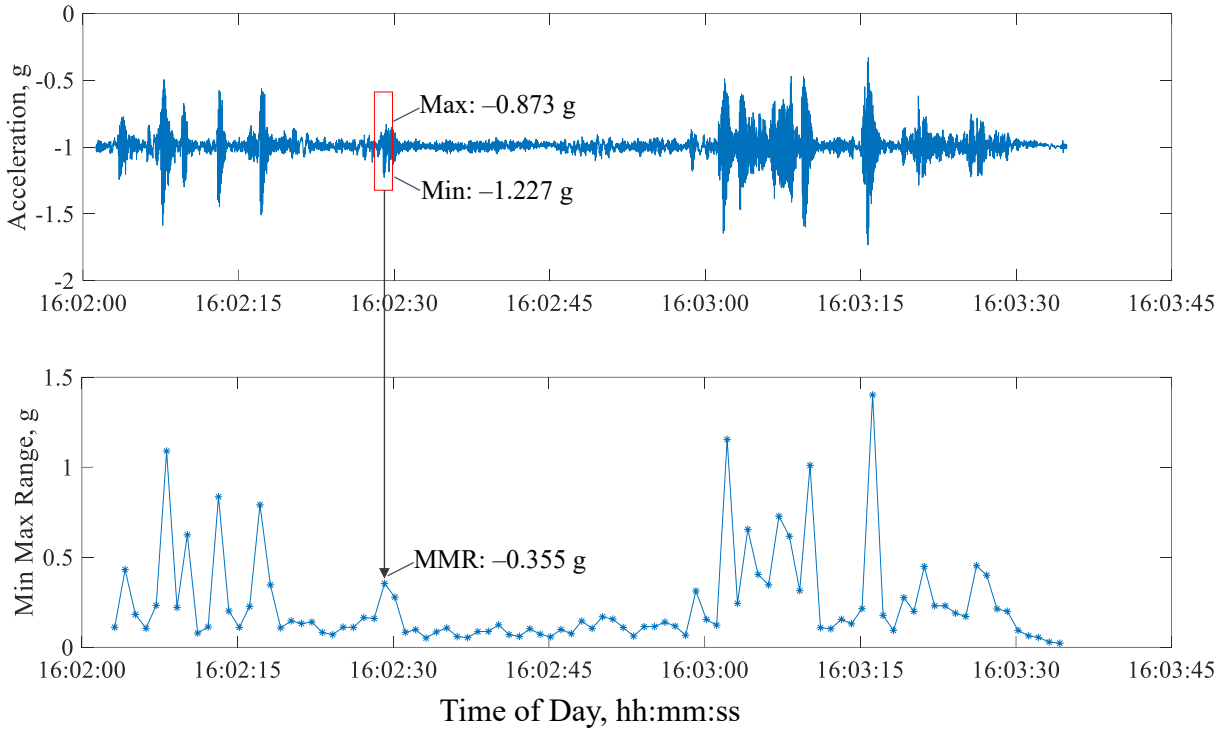

Figure 25. Original acceleration data (top) condensed to a series of MMR values (bottom).

Now that each pair of latitude and longitude has a third value, the MMR, that characterizes the driving conditions within ±0.5 seconds of that point, and the normal line plot of the driving route can be visually enhanced to show what parts of the driving route were smoother or rougher. This can be accomplished by using a geographic density plot, which is a feature of MATLAB's Mapping Toolbox. The density plot takes input arguments of latitude-longitude pairs, plus a weighting vector of the same length as the pairs, and plots each pair as a point on the map. The colour and transparency of each point is affected by the magnitude of the corresponding value in the weight vector. An example of this is shown in Figure 26.

The top frame in Figure 26 shows a 600-meter driving route from the Lakehead University parking lot to the Balmoral street entrance to campus. The normal display of the route is shown as the blue line. The geographic density plot is the overlain shading that ranges from transparent green to a nearly opaque red. Based on the weights from the MMR vector, the regions covered in a faint green hue are the smoothest and the dark red ones are the roughest. Below the route on the satellite image map is the plot of the MMR from Figure 25. It shows how the intensities on the satellite

base map are created. The MMR plot represents the entire 600-meter driving route shown in the satellite image of Figure 26.

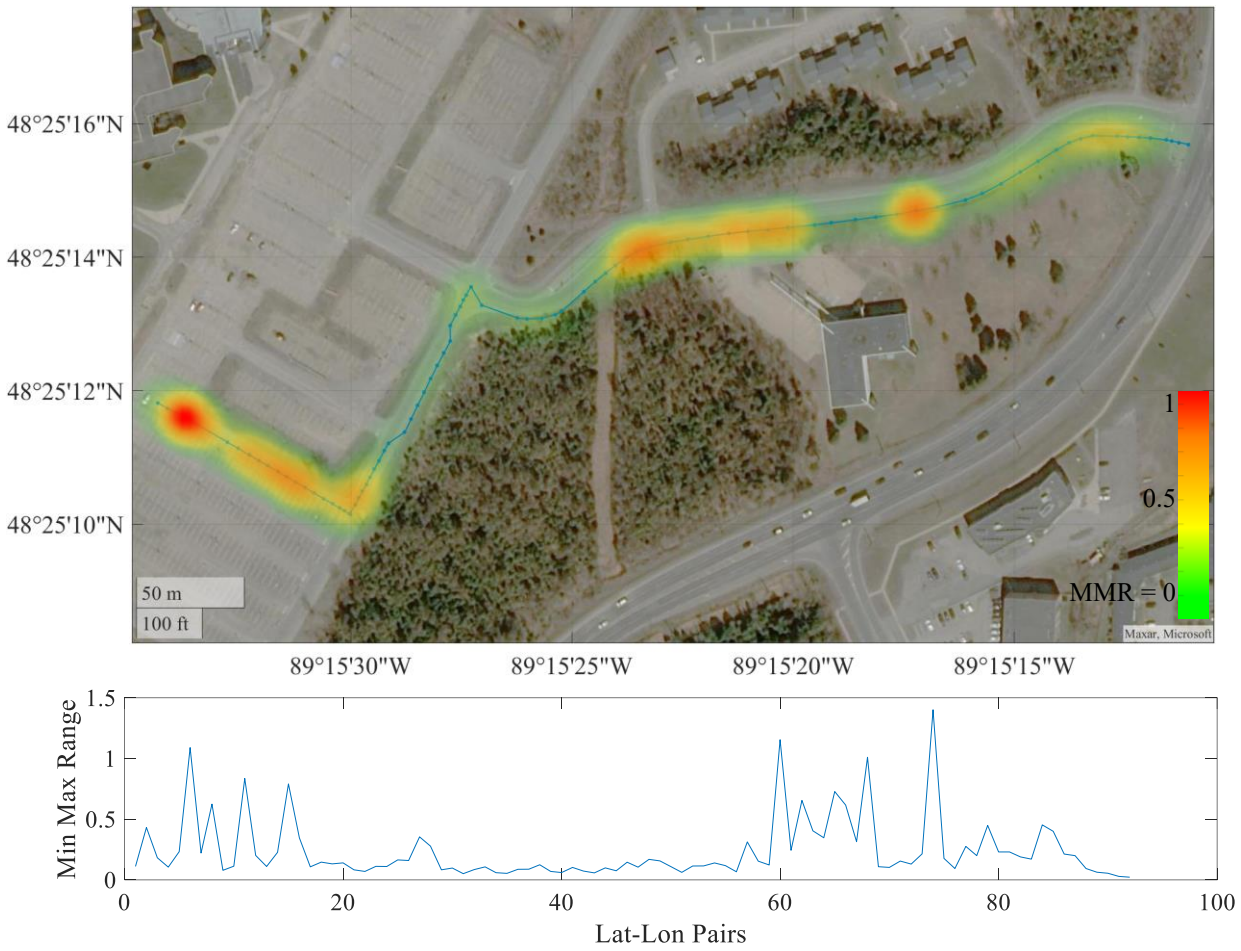

Figure 26. A geographic density plot using the MMR roughness metric

The ability to convey a sense of how the road roughness is distributed across the driving route, and where the most intense locations are, is the full extent of how road roughness was incorporated into the work of this research. Maps like Figure 26 show a good view of this, but it is difficult to incorporate other data, such as video. The next section moves away from describing overall roughness, and instead looking for specific instances—anomalies—in the road surface that can be tied to the data from the collected video footage.

3.3.2 Anomaly Detection

The purpose of the anomaly detection is to use the acceleration data as an index to find frames in the video footage that show the actual defects in the road (further explained in Section 3.5.2). In

this research, anomaly detection means identifying instances in the acceleration time series data where the vehicle has driven over a defective patch of road (e.g., crack, pothole, etc.). Referring back to Figure 25, the acceleration data show a clear spike near the 16:03:15 mark. This indicates that the vehicle drove over a sudden change in the road surface, like pothole. The 15-second interval prior to this shows several instances of acceleration spikes but also more high points overall, meaning the road is more continuously rough over that section of road. This highlights the main challenge of detecting anomalies in the acceleration data: to avoid identifying too few or too many points in the time series where the acceleration magnitude could indicate a defect.

The first approach considered was a threshold-based anomaly detector. The idea would be to find some magnitude of acceleration such that any acceleration value in the time series that exceeds the threshold is automatically considered anomalous. Figure 27 can be interpreted to justify this approach. It shows a plot of the acceleration value for each percentile taken for every acceleration value collected, over one million of them. The four red circles show were the 1st, 5th, 95th, and 99th percentiles fall. The dotted blue lines cross the y-axis at those percentile values. What they are meant to show is that the top and bottom 5% of acceleration values combined span a range wider than the inner 90% of values. Setting a threshold somewhere in those ranges could yield good anomaly detection.

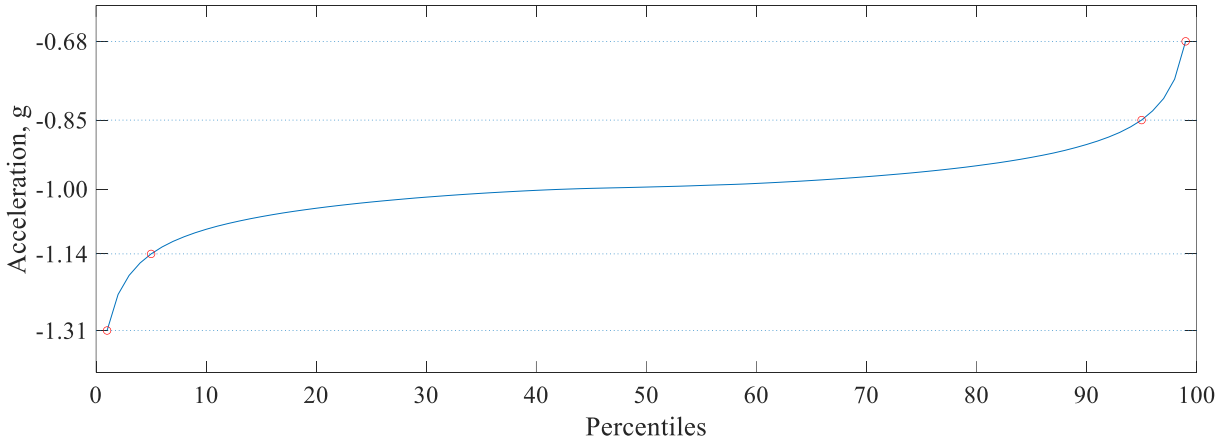

Figure 27. Percentiles for the entire set of acceleration values collected.

Figure 28 shows how problems arise with the threshold-only approach to identifying anomalies. It shows a plot of the same acceleration data from Figure 25, but zoomed in on just the region where the large spike of acceleration occurred starting near the 16:03:15 mark. From reviewing the video

footage, it can be confirmed that the cause of the spike in acceleration is from driving over a single pothole. When viewing the time series at the scale required to see it entirely, it appears that there is a singular spike in acceleration. Once the region is inspected more closely, it can be seen that the acceleration's deviation from the mean occurs over a nearly two second period. If the 95th percentile is taken as the threshold, as shown by the dotted blue line in Figure 28, there are 40 values circled in red that exceed the threshold. Since the ultimate objective is to extract a single frame from the video that shows the pothole that incited the jump in acceleration, the simple threshold produces too many. If this were the case, too many frames would be extracted from the video, many of them redundant.

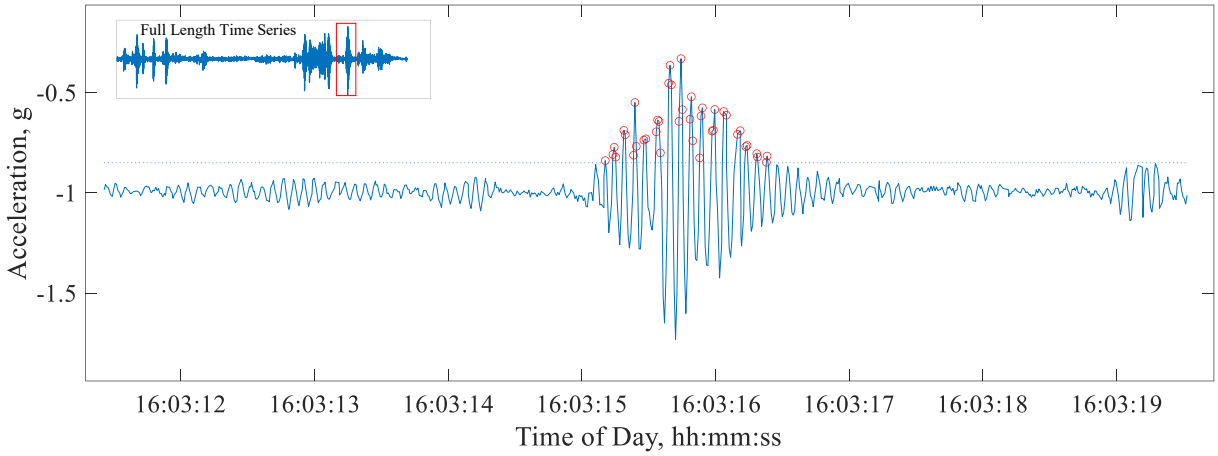

Figure 28. Closer view of a pothole-induced spike in acceleration.

The threshold could be made higher, say at the 99th percentile found in Figure 27, but then it would skip over smaller spikes in acceleration caused by less severe defects, such as cracks. This problem rules out a simple threshold-based approach to identifying anomalies, but it does not rule out using a threshold all together. Instead, we can introduce an extra parameter that will allow the threshold value to be kept low enough without over-identifying the anomalies. The minimum peak distance is the parameter that accomplishes this. Its purpose is to act as a filter against all values that pass the threshold.

We see the effects of the minimum peak distance in Figure 29. It shows the 9200 instances of acceleration values collected, with peaks identified by the red circles. Here, the threshold is set at -0.85, the 95th percentile value that was shown in Figure 27. It is identified by the blue dotted line. Without the minimum peak distance in this case, there would be 120 points identified as

anomalous, far too many considering this data was collected while driving for only 600 meters at relatively slow speed. By including the minimum peak distance of 300, there are only 10 points of interest.

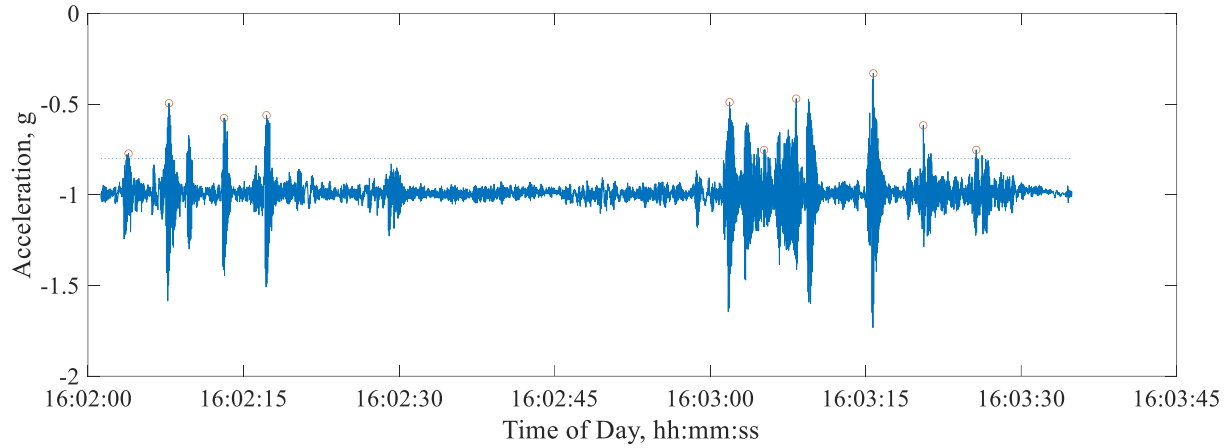

Figure 29. Acceleration time series plus detected anomalies circled in red.

Being able to successfully identify points in the road where there is likely a defect or anomaly concludes this section. Section 3.5.2 describes how this method is related to other functions that allow for video frames to be extracted. Before that, though, the process for analyzing the potential video frames is discussed in the following section.

3.4 Image Analysis Process

The purpose of analyzing the image frames, which were extracted from the video footage, is to identify defects on the road surface. The availability of these frames is based on an anomaly in the acceleration data, which was explained in Section 3.3.2. The frames are cropped, based on the specifications discussed in Section 3.2.3, and then are ready for analysis.

This section presents the three approaches that were taken to best meet the objective of taking an image frame and identifying the presence of defects and their type. First, a simple segmentation method—thresholding—is used to determine if there are defects present in the image. Second, the thresholding method is augmented with morphological operations—erosion and dilation—to remove some of the noises faced by a strictly threshold-based defect detection approach. Finally, a series of convolutional neural networks (CNNs) are trained to perform classification of the pixels of each image, thus identifying and classifying pixel regions that are defects and not.

Image segmentation, the process of atomizing an image and labeling its pixels, was the first set of methods chosen for discerning defects on the surface of the paved roads captured on video. One reason a segmentation approach to labeling defects on the road surface seemed promising, theoretically, at least, is that the visual manifestation of many defects is noticeable through their contrast with the normal road surface. Thresholding is the simplest method to exploit the contrast difference among the available image segmentation methods.

In a general sense, image segmentation is about simplifying an image, whose constituent parts are pixels, into a set of grouped pixels—the segments. There are many ways that this process of partitioning an image into sub-image groups of pixel clusters can be done.

3.4.1 Thresholding

Thresholding is the simplest method of image segmentation. Given a grayscale image, which can also be seen as a 2-dimensional intensity map, a thresholding method will mask all intensities beyond or below a predetermined value. With an 8-bit image, which contains values ranging from 0 – 255, the threshold must be in the range of 0 – 255, and whatever that value is, all pixel intensities above it will be masked as solid black and all intensities below it will be solid white, or vice versa. In other words, thresholding an 8-bit image takes an intensity map of 256 values and transforms it, based on the threshold value, to an image of only two values, black and white (i.e., binary image).

A benefit of using a simple threshold to try to identify defects is its simplicity. A threshold approach could be taken based on the assumption that, at some threshold value, a distinction can be made between all pixels that constitute defects (e.g., cracks and potholes)

Thresholding requires that an image be distilled down to a single-color channel. A typical RGB image has three colour channels: red, green, and blue. Any one of these colour channels can be isolated if the other two are filtered out of the image, or a weighted average of these three channels can be calculated.

The basic idea behind thresholding is simple: if an image is boiled down to a single colour channel, say greyscale, then maybe there is some pixel intensity value above or below which the visual features of interest can be segmented out. For example, in Figure 30 we can see how the original image of crack sealant present on the paved road surface is converted to a grayscale image from

which a threshold value can be set in order to create a mask over the pixels that exceed the chosen threshold value. Any pixel in Figure 30 with an intensity value less than 140 will be set to zero, and any pixel value above 140 will be set to one, thus creating a binary image, one where there are only two possible intensity values.

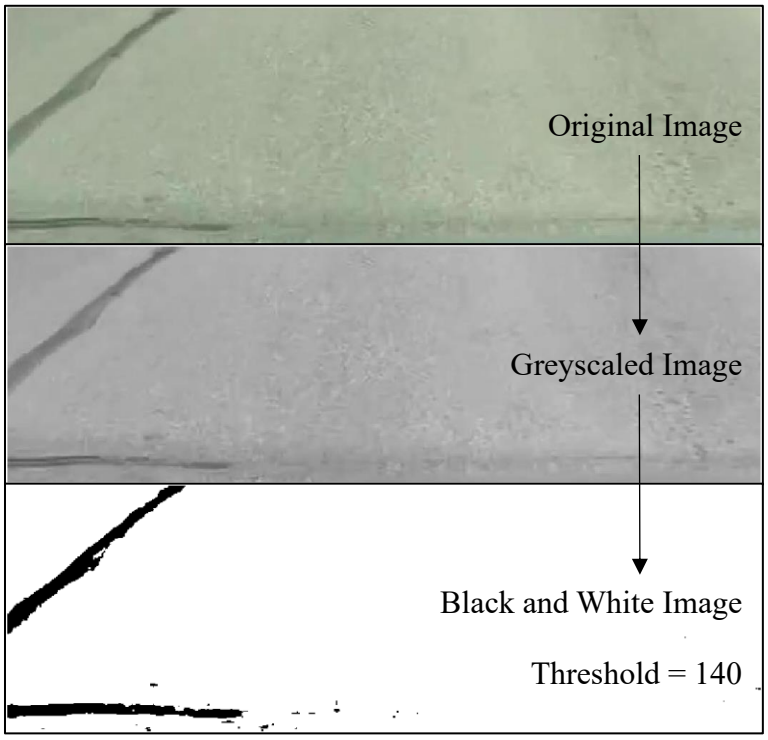

Figure 30. An example of image segmentation with thresholding.

Cracks, patches, and potholes are all road defects that can be, to varying degrees of success, identified with a threshold-based segmentation approach (Bello-Salau et al. 2014). Because this research includes a large set of training images whose pixels have all been labelled into one of seven classes (see Section 3.4.2.2), the threshold segmentation results can be quantitatively evaluated (see Section 4.3.1). Otherwise, the results would have to be manually reviewed.

3.4.1.1 Basic Thresholding

The first employed thresholding was with unedited grayscale images and a single threshold value, similar to what is shown in Figure 31. This was done with the BasicThreshold function, which outputs two arrays that correspond to the input image. The first, mask, is a logical array (contains only zeros and ones) where the ones correspond to values in the image above the threshold, and the zeros correspond to values at or below the threshold value. The other output, maskedImage, is

a copy of the original image with the mask values placed over the areas that exceeded the threshold. The developed MATLAB function is provided below:

```
function [mask,maskedImage] = BasicThreshold(threshold,image)
%BasicThreshold -- Given an 8-bit image and a threshold value in the   %
range [0,255], the function will identify all values above the        %
threshold and output the mask, a logical array of pixels that         %
are/aren't past the threshold, and a maskedImage that shows the       %
threshold mask overlain the original image

% check if image is rgb or gray
imageDimensions = size(image);
if length(imageDimensions) == 3
    % image is rgb, convert to grayscale
    grayImage = rgb2gray(image);
else
    grayImage = image;
end

mask = grayImage > threshold;
maskedImage = labeloverlay(image,~mask);

end
```

An example of this function's capability is shown in Figure 31, where the upper half is the original image, and the lower half shows the same image with a blue mask applied to the regions which were below the threshold. The threshold value was set to 140 in this case. Figure 31 also shows some of the limitations we can expect with a simple thresholding approach. In the lower left-hand side of the bottom image, the mask covers much more than the crack. The limitations of thresholding are further discussed in Section 4.3.1.

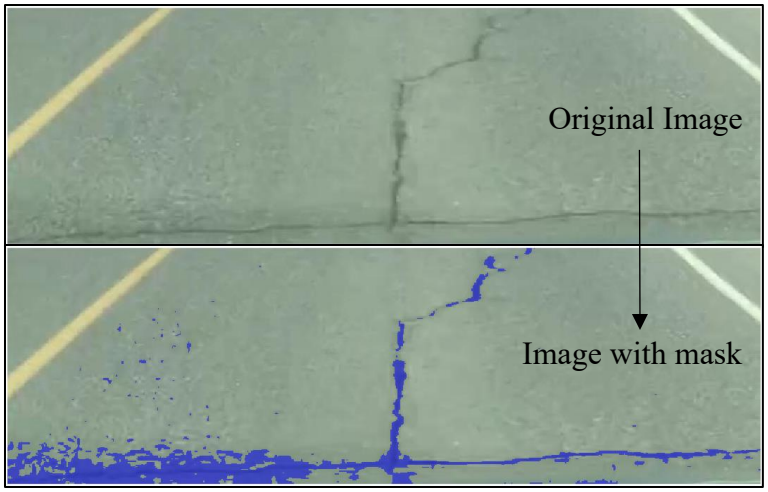

Figure 31. An image of a crack in the road and a threshold mask laid on top of it

One of the main limitations of this method is the difficulty of adapting to differing lighting conditions. This can be seen in the difference between the images shown in Figure 31 and Figure 32. The threshold value of 140, which is used in Figure 31 and the middle frame of Figure 32 works only in the former figure. A better characterization of the road's defects comes from a threshold value of 100, as shown in the bottommost frame in Figure 32.

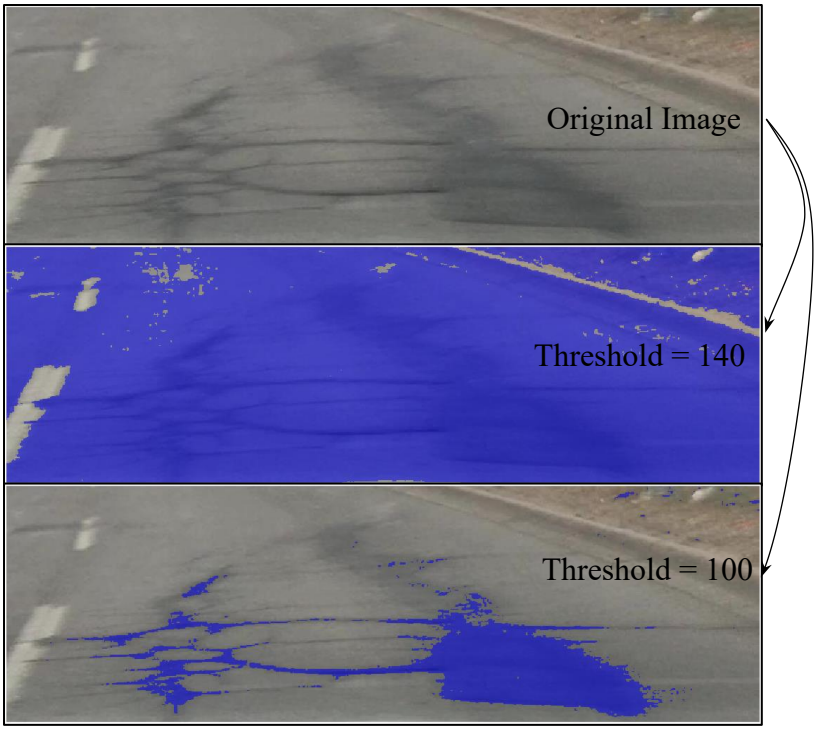

Figure 32. An example of how choosing the right threshold value can affect mask quality

3.4.1.2 Contrast Enhancement

Since the key attribute of the road defects in an image is that they are darker than the surrounding pixels, the exact threshold value used is not the most important. What matters more is that a value can be chosen that exploits the difference in pixel intensity between the darker road defects and their lighter surroundings. To do this, we can modify the grayscale image before a threshold is applied. This type of modification is called contrast enhancement.

Contrast enhancement techniques modify images in terms of their pixel intensities or the distribution of pixel intensities. One example of this is histogram equalization, where the intensities of the original image are modified to produce a histogram with a near-uniform distribution. The differences between an unmodified gray image and an enhanced image that used histogram equalization are shown in Figure 33. The frame in the upper left is the original image, and its histogram is shown directly to its right. Below, the contrast enhanced image sits beside its histogram. Both of the histograms share the same x- and y-axis limits, but the original image's distribution is mostly centered around its mean value of 130. The consequences of this distribution are shown in the original image, which is a flat shade of gray halfway between black and white.

This is much different from the enhanced image in the lower portion of the figure. It covers the entire range of possible values for an 8-bit grayscale image, meaning the image will have regions that are black, white, and ranging in between, thus having a greater visual range.

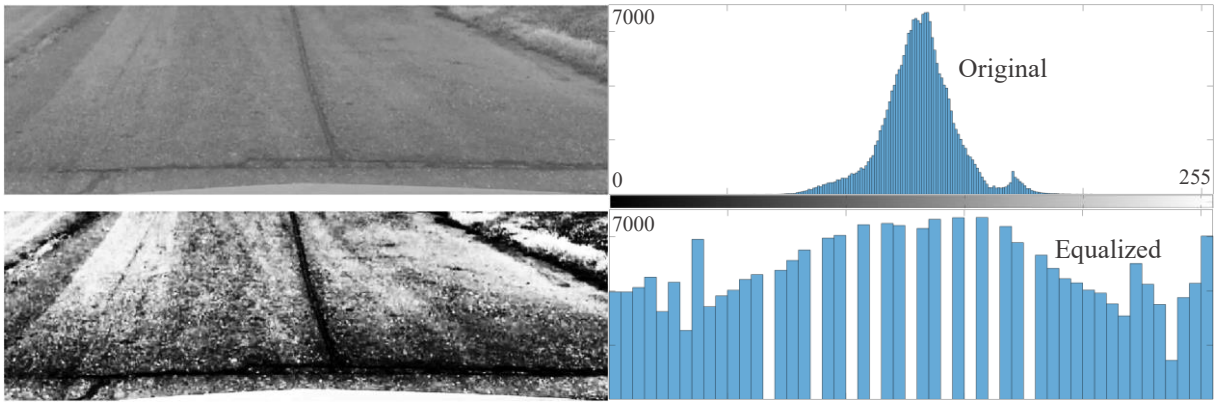

Figure 33. The original grayscale image, an equalized version, and their histograms

We don't want an equal distribution; we want a similar distribution with our desired features—the dark areas denoting road defects—to be slightly enhanced such that they are more identifiable for thresholding. So, we use the imadjust function (in MATLAB) to elevate the darker and brighter regions, but leave the rest of the image largely unchanged.

Figure 34 shows the outcome of another enhancement technique applied to the original image from Figure 33. Instead of stretching out the pixel intensities to create an even distribution, this technique takes the 1% of pixels closest to black (0) and the 1% closest to white (255) and fully saturates them to be that value (0 or 255). That is why the histogram on the right of Figure 34 shows its two outermost bins at a frequency of 2000. The image used for this demonstration is 800×250, so there are 200,000 total pixels, and 1% of that is 2000. Upon visual inspection, this method succeeds at enhancing the contrast while better preserving the appearance of the original image within it. Compared to the histogram equalization technique (Figure 33), there are far fewer bright areas, i.e., washed out regions in the image, which is also true of the original.

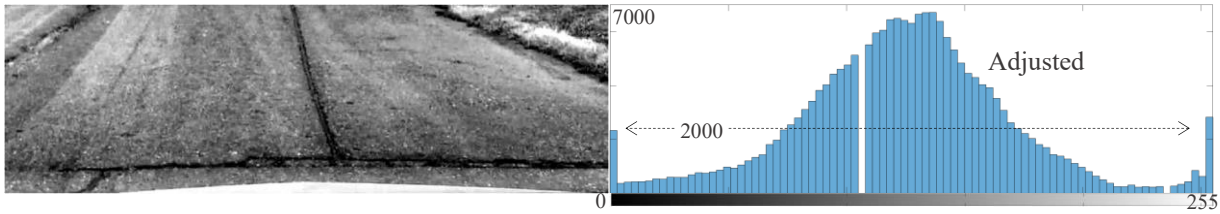

Figure 34. Contrast enhanced image using gamma correction, plus its histogram

The 1% contrast adjustment technique produces closer representation of the original image, so the next iteration of the thresholding function used images that were contrast-enhanced by this method as its inputs. And one change, specific to this use-case of the imadjust() function is that we are not concerned about saturating pixels at the high end of the histogram.

The AdaptiveThreshold function takes the 10% of pixels in the image that are darkest and stretches them across the full range of pixel intensities [0,255]. This has the effect of thresholding most of the brighter pixels, which are presumably not related to road defects. This produces the array called enhancedImage. It is then thresholded for any value less than 80% of the maximum intensity (0.8×255 = 204). In Figure 32, a threshold value of 204 would have been far too high to capture only the darkened region of the road defects, but because of the adaptive contrast enhancement used here, it is capable of producing the mask shown in Figure 33.

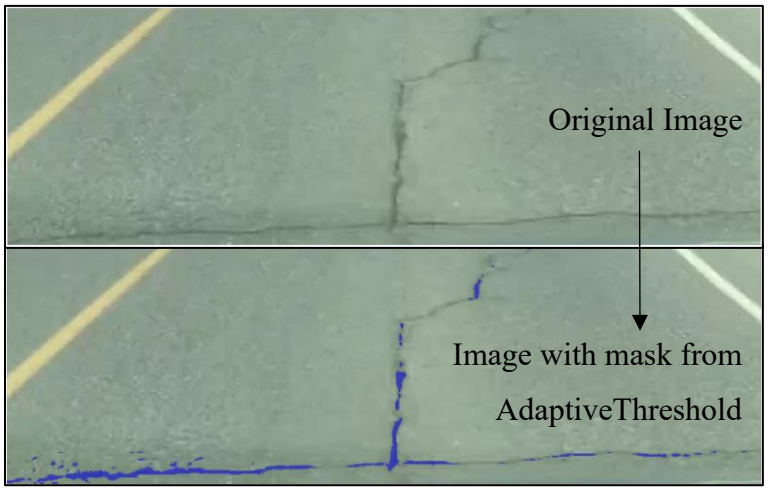

Figure 35. Mask derived from a contrast-enhance image using AdaptiveThreshold function

3.4.1.3 Morphological Operations

In image segmentation, morphological operations are those that apply to shape-based image features. Here, morphological operation—specifically, erosion and dilation—have been used as an added layer for excluding pixels that do not belong to a defect class.

Looking at the images of Figure 36, we can see one limitation of a thresholding approach to image segmentation: the road patch is darker that the surrounding road surface in many areas, but not all, which makes assigning a threshold pixel intensity value for segmenting the patch alone an unfeasible task. But patches do have image features beyond their relative. They tend to have specific solid shapes, such as a pothole-covering patch (circular), or a crack covering patch (long ellipse).

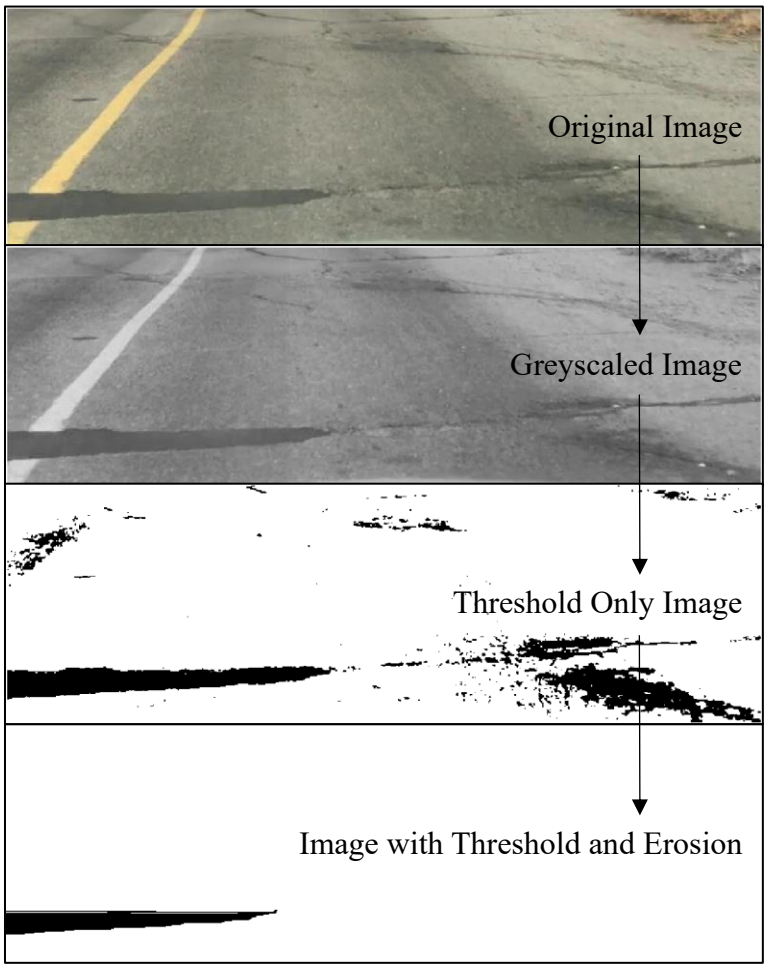

Figure 36. Image segmented first with a threshold followed by erosion.

Reasons for using more advanced methods, namely deep learning, as is discussed in the next section, are outlined in Section 4.3.1.2.

3.4.2 Deep Learning Models

Deep learning covers a variety of machine learning methods. What they have in common is that they use variations of neural network with multiple layers that have been trained using either supervised or unsupervised learning. The use of a convolutional neural network (CNN) is common for computer vision problems, such as dividing an image's pixels into groups of predefined classes. This is formally referred to as semantic segmentation, and it is the approach to classification that has been used in this research.

The goal with this approach is to create the best possible classifier. The classifier is a model that takes an image as an input and then outputs a label array, which has the same dimensions as the image such that each pixel from the input image has now been assigned a label by the classifier. Figure 37 shows a simplified version of this. For the purpose of explanation, assume the grassland input image is made up of 48 pixels, 6 high by 8 wide. The classifier takes the input image and assigns a label to each pixel based on the probability that the pixel represents that class. So, in Figure 37, we assume there are two classes: "S" for sky and "G" for grass the classifier would have labeled each of the pixels to create the output label array shown on the right side of the image.

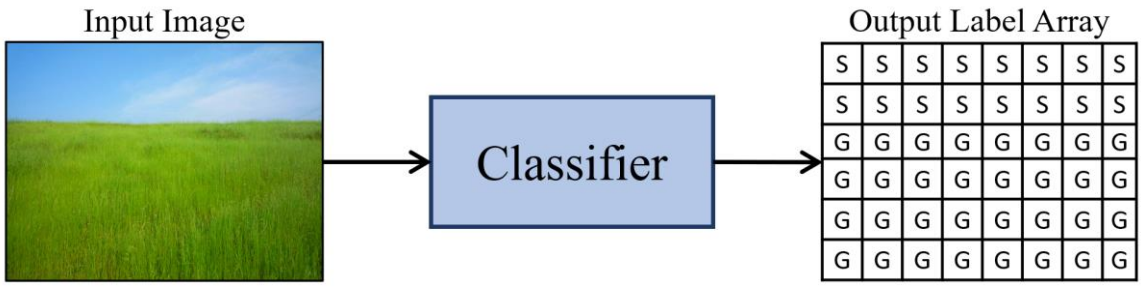

Figure 37. Basic process of semantic segmentation using a CNN as a classifier

Of course, this is an oversimplification of a complex process, but it helps raise some of the questions that this section will cover, including: how is a classifier made and how are they evaluated?

Making a CNN classifier follows a procedure as follows:

1. Collect a set of training images

2. Exhaustively label each pixel in the training images
3. Select a CNN architecture and train it

A CNN model works as a classifier by taking an input (in this case, an image) on a feedforward path through a series of intermediary layers before passing through the final output layer. The intermediary layers are known as hidden layers, because the user does not interact with them. There are typically three types of these layers: convolutional, subsampling, and fully connected. The convolutional layer is named for the operations it performs on the image, which will reduce the image size to a feature map dependent on the convolution's kernel size. The subsampling layer usually performs pooling, which further reduces the feature map's size and makes for easier computing. The fully connected layers are typically the last ones in the CNN. Their nodes are fully connected between the prior and proceeding nodes, and lead to the output layer.

The architecture of the layers can differ, meaning the number and arrangement of hidden layers vary. In general, there will be an input layer, followed by a convolutional layer, then a pooling layer. There may be several repeated pairs of convolutional then pooling layers, which are eventually followed by an output layer.

The formulation of these layers is done with back propagation, a type of algorithm commonly used for training neural networks. Once a CNN model has been trained, it is evaluated on a set of test images. Based on the evaluation of the test images, steps 1–3 may be refined or repeated in an iterative process to create the best classifier for the task being performed, which in this case, is labelling different types of defects on a road surface.

3.4.2.1 Training Images

The set of images used to train the CNN models were collected from the over three hours of driving footage captured for this research. The video footage for every instance of a noteworthy anomaly on the road surface was manually reviewed. The anomalies included cracks, potholes, patches, manholes, and sewer grates. Driving over any of them could cause a significant enough bump such that the acceleration would trigger the frame extractor function. Anytime a road defect or anomaly came into view in the frame, we took a screenshot. And once all of the video footage was thoroughly reviewed, the cropping function, mentioned in Section 3.2.3, trimmed away the outer 90% of the image, leaving the 10% portion of the frame that is relevant to the car's upcoming position.

The set of training images included a variety of lighting conditions. Collected videos included footage from sunny days to overcast days, some even when it was about to snow. This is important for having a variety of lighting conditions. It ensured that the training set included images where the road surface and any defects look significantly different due to the lighting. We collected as many images as could be found within the videos that showed good examples of road defects, including the ones that would not have been detected by the FindFrame function (the function that extracts corresponding video frames to a motion anomaly), since it is motion-activated.

3.4.2.2 Image Labelling

Each of the nearly 700 training images had to be labelled so that when the CNN trains using that image, the network can make probabilistic connections between the pixels of the original image and the semantic context provided by the labels. But first we had to decided how we would define the classes.

The challenge of defining the classes to use for labelling all of the training images is between having enough classes so that every feature of the image is logically defined, but having few enough classes so the CNN is exposed to a wide variety of examples of each class. For example, having a single class for all road cracks is simpler and should yield a more rigorous model than if there were several subcategories for the road cracks, such as longitudinal, transverse, and others. Having multiple subclasses for each road defect would make overfitting more likely during the training process, creating a classifier that performs much more poorly when exposed to images it has not encountered before. After reviewing the video footage and the extracted and cropped frames, it was decided that seven classes sufficiently capture the different types of features across the training image set without creating complexity in the labelling or training process. Each class is described in Table 9.

Table 9. Labels used to classify pixels on frames collected from video footage

Class Label	Description
Road	The main road surface, except for any of the defects contained within it. Also includes portions of cars if they appear in the image frame. Does not include anything beyond the pavement, so gravel shoulders are labelled as background.

Background	Any visual features caught in the cropped frame that are not part of the main road surface. Starting from the curb, including shoulders and other unpaved terrain. Also includes the hood of the vehicle, which is sometimes captured by the camera.
Crack	All types of cracks that are visually discernible in the cropped frame and contained within the road category label. Cracks in road-adjacent surfaces, mainly sidewalks, are not labelled since they would be contained within a background label, not a road label.
Pothole	All pothole features contained in the road surface. Most often they show some depth and appear as a rough elliptical shape. Sometimes a judgement has to be made whether the defect is a wide crack or a small and oblong pothole.
Patch	All patches on the road surface, large and small, characterized by the comparatively darker colour and more uniform texture compared to the normal road category.
Manhole	Also known as a maintenance hole cover. This category was considered worth creating a label category for because driving over one can cause a spike in vertical acceleration.
Grate	Similar to the manhole, but usually present at the road's curbside, and mainly differentiated from manholes by its square shape, as opposed to the manhole's circular shape.

All of the images were manually labelled using MATLAB's Image Labeler application. This app provides a graphical user interface that allows the user to load in a set of images, create scene labels, and then label the images. The graphical user interface is shown in Figure 38. The left pane is where the ROI (region of interest) class labels are created and selected. The image thumbnails in the bottom carrousel shown in Figure 38 come from the directory that is selected when the image labelling session begins. The top portion of the image, where the toolbar is positioned, contains the tools used for labelling the images.

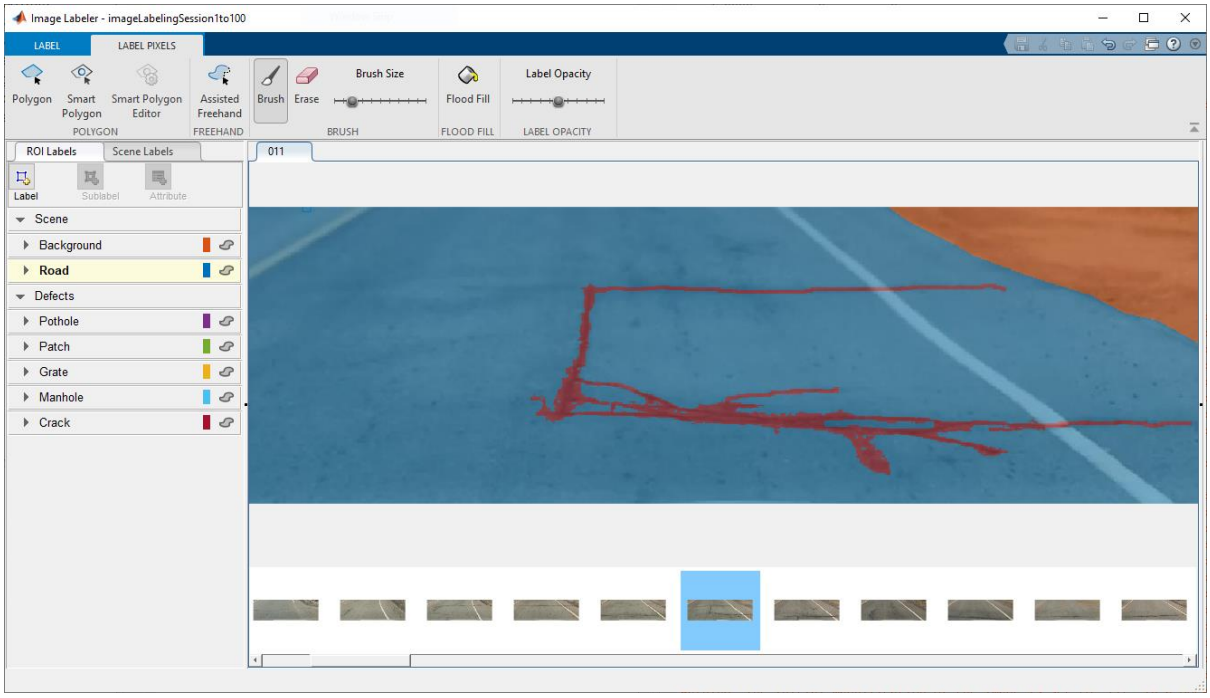

Figure 38. Example of working within the MATLAB Image Labeler application

There are three tools provided for labelling: brush, flood fill, and polygon. The brush tool uses the mouse pointer position to label the image as the mouse is clicked and dragged around the image. Because the brush size can be made smaller or larger, the brush tool was useful for labelling cracks. The flood fill tool had limited use for this application. It is applied on whatever point on the image that the user clicks, and then will spread out and label any adjacent pixels that share the same colour. Sometimes this worked for labelling cracks or patches, but usually there was enough variation in the colour of pixels that make up the defect which hindered the performance of the flood fill tool. In addition, the road surfaces had a rather similar colour to the defects which could mislead the flood fill to large amount of over labelling that would have to be undone. The polygon tool allows the user to click on the image, sequentially placing vertices of a polygon that, once closed, will fill itself in with the label type that was preselected. The polygon tool was useful for labelling patches and potholes because they usually had irregular perimeters.

Figure 39 shows an image labelled with five of the seven classes. The top half is the original and unlabelled frame, and the bottom half shows the labelling of each class present in a different colour. For this image, to ensure that every pixel is labelled, the process would go as follows. First, use the polygon tool to label the entire frame as Road. Then use the polygon tool to label the upper-

right corner as background, and then do the same at the bottom where the hood of the vehicle is shown in the frame. Use the polygon tool to label the manhole and patch, and then use the brush tool to manually trace all of the crack in the road.

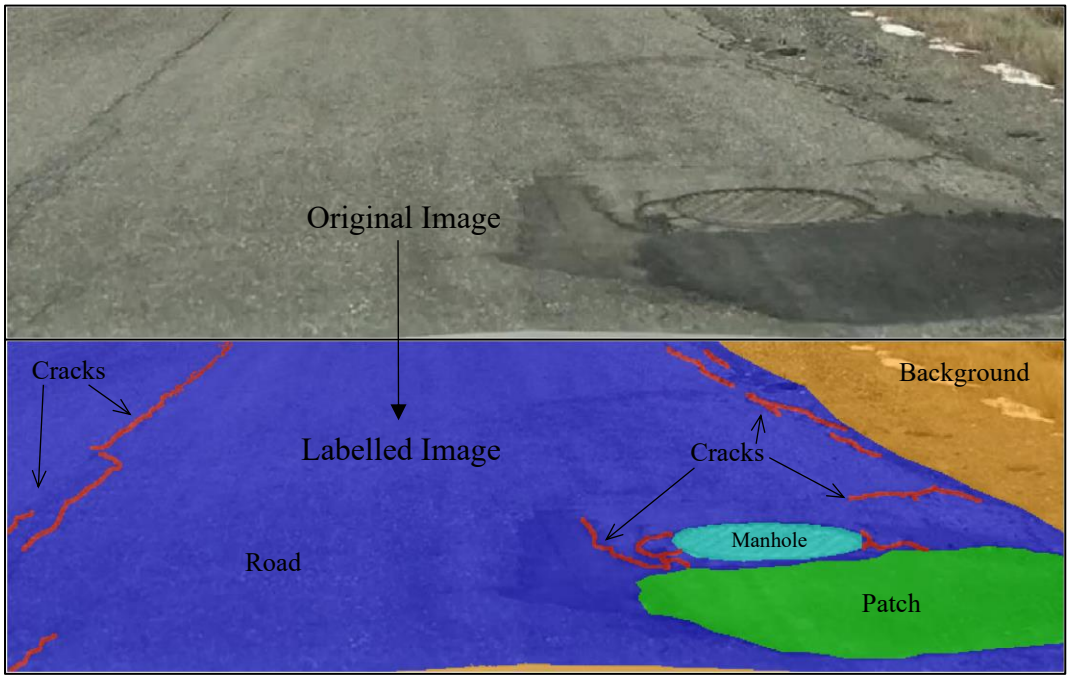

Figure 39. Top: Original image; Bottom: Fully labelled image with five classes

3.4.2.3 Model training

Once all of the training images have been labelled, it is time to export the data from the Image Labeler app. There are two components to export: the ground truth file and the pixel labels. The ground truth file, referred to as gTruth in all of the codes (see Appendix A), is a MATLAB object that is used to reconcile the different data sources necessary to perform deep learning. The ground truth object includes a directory to all of the images that have been labelled, the different classes and their numerical ID codes, and a directory to all of the labels. These labels are the second component of the Image Labeler app's export function. They are saved as Portable Network Graphics (PNG) image files inside a folder called PixelLabelData. The pixels that make up the label image are numbers that correspond to the classes defined in the Image Labeler application, which are shown in Table 10.

Table 10. Image feature classifications and their pixel labels

Class	Background	Road	Crack	Grate	Manhole	Patch	Pothole
Pixel ID	1	2	3	4	5	6	7

If an image was labelled and it only contained the road and background, then the pixels in the corresponding PixelLabelData image would only contain pixels having the value of either 1 or 2. Since, they are saved as 8-bit images, where black is 0 and white is 255, all of the images appear almost entirely as black. They serve no purpose visually; they are just a means of data storage and this way the amount of system memory required for the training process is greatly reduced. Instead of loading all of the label arrays into the memory, they are only accessed when needed.

The full code used to prepare, create and train the CNN models for semantic segmentation is included in Appendix A. This section will include snippets of the full code to describe the process for building and training the network.

The first step is data preparation. Section 3.4.2.1 describes the procedure that was followed to collect the images used for training, which is that as the video footage was played, the frames that showed road features and defects were extracted and saved. Because the frames were saved in the chronological order in which the video was played, there is a possibility of data overfitting. The solution is to randomize the set of images so there is no pattern to the order of the images used to train the classifier. Another important step for preparation is setting aside a percentage of the training images to be used for validation. In this case, 99% of the images went toward training, leaving only 1% for validation of the model. Again, there is a possibility of creating a model that overfits when too much of the data is used for training, but there is another set of labelled images that will be used for testing later on.

Now that the training images are fully prepared, we can proceed to create the CNN itself. In MATLAB, this could be done with the segnetLayers (for VGG16 and VGG19) and deeplabv3plusLayers (for ResNet and MobileNetV2) functions, which take image size, number of classes, and the network architecture as inputs, and then output the selected network architecture as a pretrained model.

```
lgraph = segnetLayers(imageSize, numClasses, 'vgg16');
```

```
lgraph = deeplabv3plusLayers(imageSize, numClasses, 'resnet18');
```

Based on the architecture chosen, the CNN will have pre-set layers and weights from the pretraining process on the ImageNet database with more than one million images. It has been shown that using pretrained CNN models on unrelated large training datasets can improve training process on a new set of training images and classes (Mei and Gül 2020). This approach of training is named transfer learning. The layers and weights should be adjusted to perform better with the new training images. This will give more weight to classes that are under-represented in the data (for the distribution of labelled data, see Section 4.3.2.1). The method used here is median frequency weighting, where the weight given to each class is determined by the inverse of the overall median frequency. This has the effect of making it expensive within the CNN's loss function to mistake an under-represented class.

```
tbl = countEachLabel(pxds);
imageFreq = tbl.PixelCount ./ tbl.ImagePixelCount;
classWeights = median(imageFreq) ./ imageFreq
```

Once the class weights are set, they can be added into the CNN by replacing the existing, predefined classification layer.

```
pxLayer = pixelClassificationLayer('Name','labels',...
                                   'Classes',tbl.Name,...
                                   'ClassWeights',classWeights);
lgraph = replaceLayer(lgraph,"classification",pxLayer);
```

The only remaining work before training the CNN is to define the training options, which are listed in the trainingOptions function below. Of the options listed, there are a few worth describing. Stochastic gradient descent method ('sdgm') is the method used for iteratively optimizing the network's parameters. When the network has iterated on its parameters through all of the training data, this is considered one epoch, so the 'MaxEpochs' value refers to how many times the network will pass over all of the training data. The learning rate dictates how much or little the network parameters are updated with each step through the data. The MiniBatchSize determines the number of images processed at each step and the Shuffle option randomizes the order of training images.

```
options = trainingOptions('sgdm', ...
```

```
'Momentum', 0.9, ...
'InitialLearnRate', 1e-2, ...
'L2Regularization', 0.0005, ...
'MaxEpochs', 100, ...
'MiniBatchSize', 1, ...
'Shuffle','every-epoch', ...
'CheckpointPath', tempdir,...
'Verbose', false,...
'Plots','training-progress');
```

With the data and network adequately prepared, the training of the network can begin. Because the image data and the network are arrays that can be broken up for parallel work, the training process runs much more efficiently on a graphical processing unit (GPU). The CNNs for this research were trained on a desktop computer with 16 GB RAM, 4.7 GHz Intel Core i7 CPU, and an NVIDIA GeForce GTX 1660Ti GPU. Training of a new model on the GPU took between one and two hours, but on a computer without a dedicated GPU, the process likely would have taken several days or more.

3.4.2.4 Model Evaluation and Refinement

Creating a model that can classify every pixel in an image is a complex task. Having good data and proper training is important, but so too is being able to evaluate the trained model and going back with changes or new data to refine it.

The best approach to evaluating the performance of the model is with a test dataset. From the images labelled for road defects; 50 images were held back from the model during its training so its performance could be evaluated on images it has never encountered before. The test images are processed by the classifier, which outputs a label array for each image. Each of these is then compared against the ground truth label array. There are several means of comparison, such as the accuracy, intersection over union, BF score, etc. Their meaning is discussed in Section 4.3.3.

The performance metrics are used to inform the decisions made around refining the model. The first method we used to improve model performance was to add images to the training set to expand the data available to the CNN. Originally the training set included only 445 images; another 213 were added later on to improve the performance. A benefit of improving the models this way is

that the model does not need to go through entire training process again. The model's ability to classify images, based on the weights it set from previous training, are the starting point for additional training with new images. Reapplying models like this is known as transfer learning. Another approach to improving segmentation performance is by using a different CNN architecture. The architectures used for this research are listed in Table 11. Using different architectures has the effect of providing a different starting point for the training process and, because their layering structure is different as well, they respond to image features in different ways. The performance of each of them is discussed in Section 4.3.3.

Table 11. Network architectures used for convolutional neural networks

Architecture	Description
VGG-16	Visual Geometry Group, the name of the research team who developed it at the University of Oxford. It is a 16 layer deep convolutional neural network trained on ImageNet database (Simonyan and Zisserman 2014).
VGG-19	Another CNN developed at the University of Oxford, but with 19 layers instead of 16. Like VGG-16, the image input size is 224×244, and the predefined network weights are from training on the ImageNet database.
ResNet-18	Short for Residual Net. ResNet-18 is 18 layers deep but uses residuals, references to other layers in the network, to improve training speed when compared to the unreferenced layers of VGG (He et al. 2016).
ResNet-50	A 50-layer deep Residual Net, similar to ResNet-18. It is trained on the ImageNet database and can classify over 1000 object categories.
MobileNetV2	Like all of the previously listed CNNs, MobileNetV2 is based off training done on the ImageNet database. It is 53 layers deep, developed by Google to specifically perform well on mobile devices (Sandler et al. 2018).

3.5 Data Integration

So far, all of the data discussed in this chapter has been separate from the other types. Motion and vision data have been analyzed separately, and now this section will cover the process followed to combine them into useful information. The process follows several steps for augmenting and creating new data. These processes are shown in Figure 40.

This flow diagram starts with the raw data, collected using a smartphone as discussed in Section 3.1. The two files are synchronized using the DataClipper function (see Appendix A), which reduces the size of the Sensor Data to match the duration of the video footage. Next, the clipped data and video are inputted to the FindFrame function, which identifies spikes in acceleration data

and the video frames that correspond to those spikes. This function locates those frames within the video and the GPS location at the time of capture. This is combined in the GeoTag function to create a set of images that have the location embedded in the image's metadata. The same images are also fed through a classifier that segments the image to identify any defects in the road surface. The list of defects, along with the corresponding images, are then packed in a Keyhole Markup Language (KML) file, which can be uploaded to various GIS programs, including the Google Maps Platform. An example of how this data is represented in Google Maps when uploaded as a KML file is shown in the bottom right corner of Figure 40. The steps outlined in the figure are further discussed in the remaining part of this section. Snippets of code are included; the full copies can be found in the Appendix A.

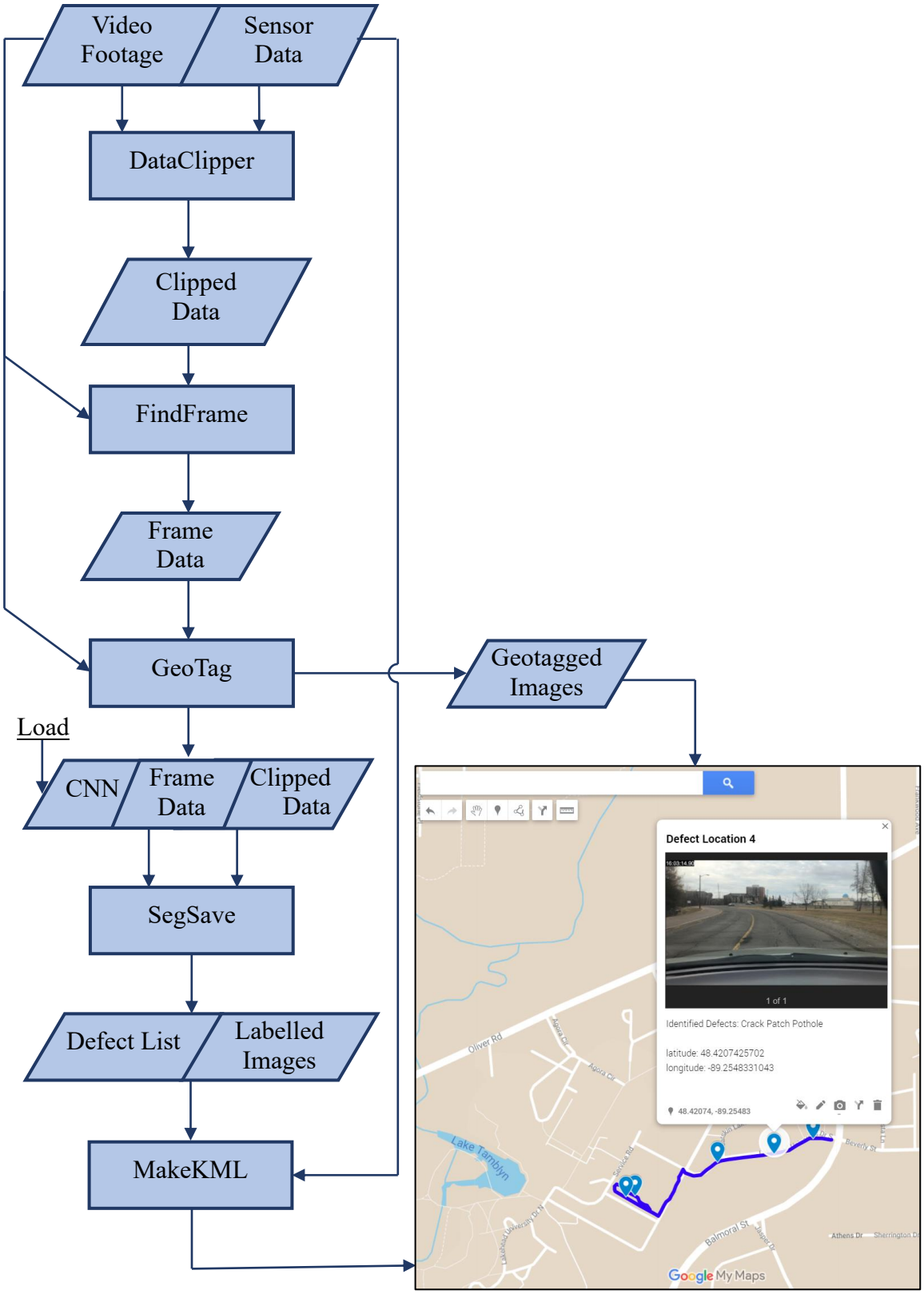

Figure 40. Steps to integrate the motion and video data

3.5.1 Data Synchronization

There are a few challenges associated with extracting image frames from the video footage based on motion data. The main challenge is synchronization. Because a single device is used to collect both the video data and the motion data, their start times are always slightly off from each other. This comes from the procedure used to collect the data: open the SensorLog app, begin recording, switch to the Timestamp Camera app, and begin recording video before starting to drive. When driving finished, repeat the steps in reverse: stop recording the video, then stop recording with SensorLog. The consequence of this is that the SensorLog data is always a bit longer than the video duration.

The differences in recording frequencies also make perfect synchronization between frames and motion data instances rare. Luckily, the data resolution for both can be high enough (30 frames per second for the video, 100 readings per second for the motion data), that nearly corresponding video frames and motion data are always within a fraction of a second of each other.

The DataClipper function addresses the synchronization problem by trimming the data from sensor log to have the closest start time as the video as possible. The key to this is the time overlay in the video footage, as shown in the upper left corner of Figure 24. Because the part of the frame that holds the timestamp is essentially black and white (i.e., it is a structured scene), MATLAB's built-in optical character recognition function can produce a string based on the characters present in the image. The character recognition accuracy is usually over 90%, but if it was to fail, then an error is presented to the user who may have to review the files and synchronize them manually.

Next, the time of day represented by the string is subtracted from all of the timestamps in the sensor data, and the minimum absolute value of these subtractions is taken as the first entry in the sensor data that corresponds to the video. All sensor data entries preceding this value are removed, and the same procedure is followed for the last frame of the video.

3.5.2 Frame Extraction

Now the data generated from SensorLog is synchronized with the video footage, and acceleration signal can be used to reliably extract frames using FindFrame function. It takes the synchronized data as its inputs and outputs a table that includes where each anomalous frame is located in the video and the latitude and longitude position at that time.

```
function frameData = FindFrame(clippedData,video)
```

It follows two main steps: find the acceleration anomalies, then find the corresponding frames in the video. The identification of high acceleration events, which usually represent road defects, has been discussed in Section 3.3.2, so the main challenge for developing FindFrame was making sure that the extracted frame contains the defect(s) that caused the acceleration event. The difficulty comes from the distinction between data and event synchronization. Data synchronization (explained in Section 3.5.1) has been done such that the first frame in the video is as close to the first recording of sensor data as possible. The problem for frame extraction is that the defect appears on the video before the vehicle drives over it and the smartphone registers the corresponding acceleration. In the FindFrame function, the approach taken to solving this problem is with a speed-based correction variable (see Section 3.5.2.1).

3.5.2.1 Correction Variable

The correction variable is the integer number of frames that must be subtracted from the data-synchronized video so that it is event-synchronized. Without the correction variable, the frame extracted from acceleration anomaly detection would be too far ahead in the driving route.

In other words, if acceleration is identified at point n in the acceleration series, it corresponds to frame m in the video footage, since it is data-synchronized. But frame m corresponds to a view of the road that is too far ahead, since the defect is shown in the frame before the acceleration series. Thus, we need to apply a correction, c, such that frame $m - c$ in the video shows whatever defect in the road that was detected in the acceleration series.

The correction variable is the output of a speed-dependent linear equation. The slope and intercept of the equation were derived from a linear regression. The points for the regression were collected manually by counting the number of frames back in the video before the defect was in good view, just ahead of the vehicle. The speed of travel at the time is the second piece of data used to produce Figure 41, which shows the points collected and the line of best fit with its coefficients used to produce the correction variable.

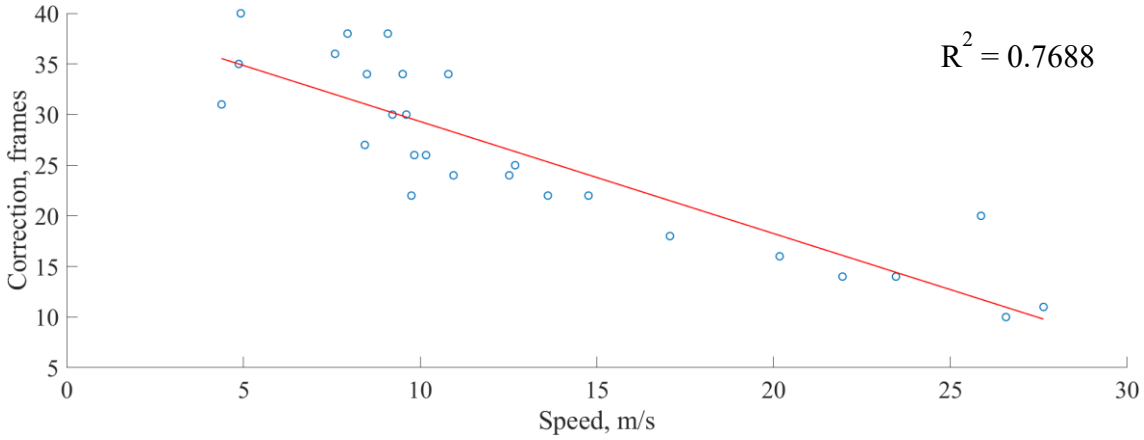

Figure 41. Values manually collected to determine speed-based correction factor.

3.5.3 Geotagging

The output of the FindFrame function is a table of values whose columns include: what frame number(s) in the video show the acceleration spike, and the latitude and longitude of the phone when the acceleration spike was detected. This table and the MATLAB video object are input into the GeoTag function for writing the image and adding the location (latitude/longitude) data to the images metadata. The location data is added to the metadata using ExifTool, a command-line application for manipulating metadata. The application is freely licensed, and as the name implies, it can modify metadata for images captured using the exchangeable image file format (Exif), though it can operate with other file standards than Exif.

An example of the two commands issued for ExifTool to write the location metadata to the image are shown below. Each directive is preceded by a hyphen. The -overwrite_original command is not necessary for geotagging, but it stops ExifTool from creating a new image with the added metadata. Next, -exif:gpslatitude and latitude ref will set the latitude, in decimal degrees, and which hemisphere it is in. The final part of the command is to include the file name that the metadata is being written to. The steps are repeated in the second line for the longitudinal information.

```
[1] >exiftool -overwrite_original -exif:gpslatitude=48.3  -exif:gpslatituderef=N
              -ImageFileName.jpg

[2] >exiftool -overwrite_original -exif:gpslongitude=89.1 -exif:gpslongituderef=W
              -ImageFileName.jpg
```

All of this is done automatically by GeoTag, using the MATLAB system() function that will initiate command-level instructions without having to leave MATLAB.

3.5.4 Image Defect Identification

The next step is to segment the geotagged images by the trained CNN classifiers. This is done with the function SegSave:

```
function defectList = SegSave(clippedData,frameData,video,CNN)
```

The clipped data from the DataClipper function is only used for its easily accessible datetime information, which is used for naming the image files when they are saved. The naming convention used for the labelled images is YYYYMMDD_###_Labelled.jpg, where ### is the frame number in the series of extracted frames (e.g., 20200105_003_Labelled.jpg would be the 3rd frame extracted from the video captured on January 5th, 2020).

The frame data contains the index of frames that must brought out of the video and cropped down to a size of 250×800 pixels. This cropped image is then processed by the convolutional neural network. The CNN will produce a label array, and all of the unique values in the array that are defects (i.e., not of the road or background class) are entered into a cell array called defectList, the output of SegSave. The function CreateDefectList, shown below, is executed in a loop after the semantic segmentation function to create a list of detected defects.

```
function defectList = CreateDefectList(labels)
 allDefects = {'Crack','Grate','Manhole','Patch','Pothole'};
 uniqueLabels = unique(labels);
 defects = removecats(uniqueLabels,["Road","Background"]);
 defects = uint8(defects);
 defects(defects==0)=[];
 defectList = strjoin(allDefects(defects));
end
```

The output, defectsList, is added as a row to a cell array that has as many rows as there are extracted frames from the FindFrame function.

```
defectsList =
  5×1 cell array
    {'Crack'              }
    {'Crack Patch Pothole'}
    {'Crack Patch'        }
```

```
{'Crack Patch Pothole'}
{'Crack'              }
```

The list of defects for each extracted frame, plus the saved frames are used in the final data integration function to show all of the information together in a useful format.

3.5.5 KML Data Packaging

The last step for integrating all of the data together is to package all together in a Keyhole Markup Language (KML) file. KML is a GIS file format, first used by Google Earth to show geographic information with the possibility of adding annotations. This is done with the MakeKML function, which is shown in full in Appendix A.

```
function MakeKML(clippedData,frameData,defectList)
```

The function has no outputs into the MATLAB workspace, but it does save two KML files into a folder. One file is for the total route driven, and the other is for the road defect information. The two files can be used with the Google Maps Platform API (application programming interface) to create custom map with the data. An example of these maps is shown in Figure 42. Each uploaded KML file creates a new layer in the map, indicated in the left pane of the window. There is a layer for the driving route, which is expressed by the blue line showing the travel path. Then there is a defect layer, stored as a set of points (in KML, referred to as Placemarks), which includes the location information, the identified defects from defectList, and the video frame from that moment during the drive. Much of the appearance can be customized, and the layers can be edited in case of a mistake or if a change needs to be made.

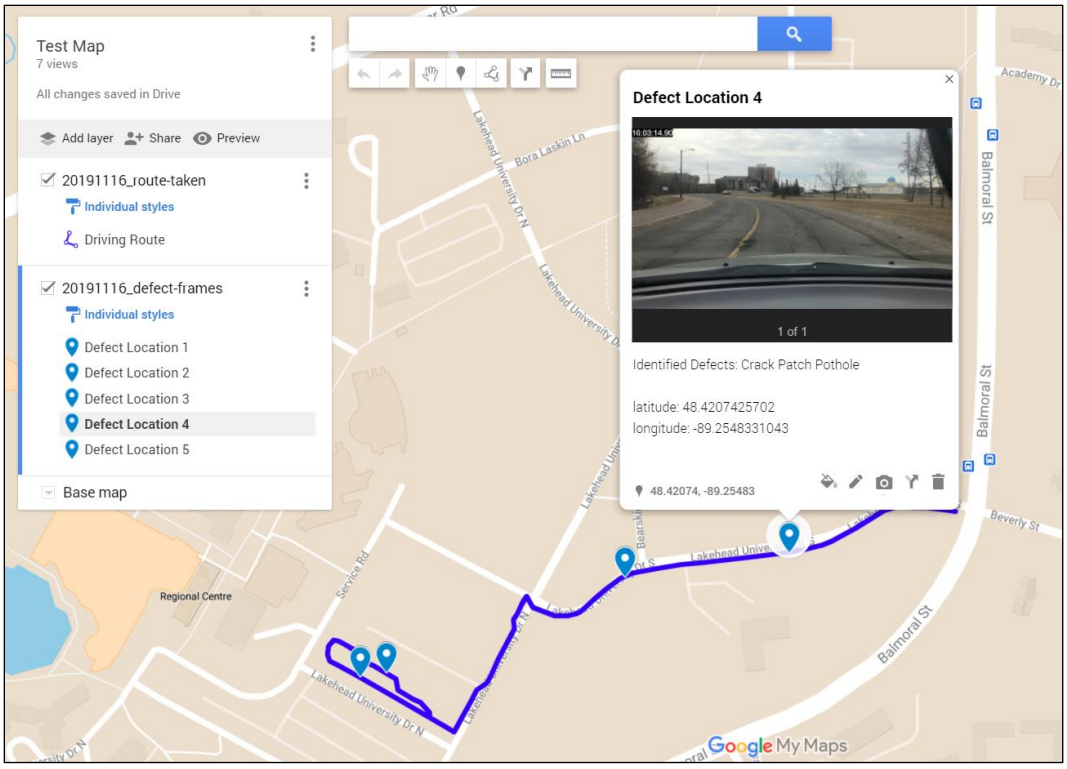

Figure 42. Example data display of KML files loaded into Google MyMaps

The data structure for KML is similar to XML, as shown below in the KML file used to describe the driving route that is shown in blue in Figure 42. The KML below describes a line object That has properties of a name, line width, line colour, and coordinates. Only one set of coordinates, in the form of latitude, longitude, altitude, are shown for brevity. The line shown in Figure 42 is made of 134 points captured during a 2.5-minute drive through Lakehead University's parking lot. Since the altitude has not been included with the data, it defaults to a value of 0.

```
<?xml version="1.0" encoding="utf-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
   <Document>
      <name>20191116_route-taken</name>
      <Placemark>
         <Snippet maxLines="0"> </Snippet>
         <description> </description>
         <name>Driving Route</name>
         <Style>
            <LineStyle>
               <width>5</width>
               <color>ffff0000</color>
            </LineStyle>
         </Style>
         <LineString>
```

```
            <coordinates> -89.2585944,48.41961071,0 ... </coordinates>
         </LineString>
      </Placemark>
   </Document>
</kml>
```

The KML file for the defect list is similar. Its placemark attribute contains point string coordinates instead of linestring coordinates, and it follows the same latitude-longitude-altitude format as the linestring. Below is a snippet of the KML file used to create the defect layer in Figure 42. The full file contains a new placemark for each defect locations, which would be five in the case of the data in Figure 42. For brevity, the four additional placemarks were removed on the third to last line.

The information that accompanies the point, such as the image and the identified defects, are included in the description section of the placemark. The "img src" refers to the URL that the image file can be found at. For this example, all of the image files are stored on the website GitHub, and are accessed from there. The height and width refer to the size of the image in pixels. And the latitude, longitude, and identified defects information is taken from the entry in frameData and defectsList variables that correspond to this particular defect. Finally, the
 entries are used to add a line break in the text, so that it can be read more easily.

```
<?xml version="1.0" encoding="utf-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
   <Document>
      <name>20191116_defect-frames</name>
      <Placemark>
         <Snippet maxLines="0"> </Snippet>
            <description>
                  img src="https://raw.githubusercontent.com/user/..."
                  height="300" width="auto"/> <br><br>
                  latitude: 48.4199002418 <br>
                  longitude: -89.2590580935 <br><br>
                  Identified Defects: Crack
            </description>
         <name>Defect Location 1</name>
         <Point>
            <coordinates>-89.2590580,48.4199002,0</coordinates>
         </Point>
      </Placemark>
      ...
   </Document>
</kml>
```

Chapter 4: Results and Discussion

This chapter presents the results produced by different modules of the developed system and a discussion of the data that has been analyzed.

4.1 Data Replicability

To ensure that the data collection setup will provide comparable results for different devices, vehicles, and mounting setups, a trial comparison was performed between the main testing vehicle, the Lexus RX400h, and another similar small SUV, a 2016 Hyundai Santa Fe Sport. Vehicle specifications have an effect on the response of the suspension system, and therefore, on the smartphone logging of the vibration. These specifications are listed in Table 12. These specifications correlate to how the vehicle's overall mass will distribute across its four wheels, similar to the quarter car model described in Section 2.2.2.1.

Table 12. Vehicle specifications for comparing vibration patterns between different vehicles

Specification	RX400h	Santa Fe Sport
Wheelbase	2715 mm	2700 mm
Axel track	1554 mm	1638 mm
Curb weight	1901 kg	1650 kg

The same smartphone and windshield mounting device was used in both vehicles. The trial was conducted over a straight 800-meter stretch of an urban road, and with vehicle speed kept between 40–45 km/hr. Each vehicle conducted two runs over the test strip of the road. The raw vertical acceleration data collected by the smartphone can be seen in Figure 43. The top two plots are from the RX400h and the bottom two are from the Santa Fe Sport. The plot of each time series was aligned with the others based on their maximum acceleration value, which can be seen at the 3500th position in the top subplot, and then in the three below it.

Each plot in Figure 43 is unique. Their series of acceleration values depends on the vehicle's speed, its position on the road, the positioning of the data-collecting smartphone, and some other factors. But there are visually discernable similarities in the overall form of each plot. In addition to the max acceleration event, which was used for synchronizing each plot, there are other acceleration

spikes around positions 500, 1400, 1700, 2200, and 3000. Of course, noting the visual similarities between the plots is only a start.

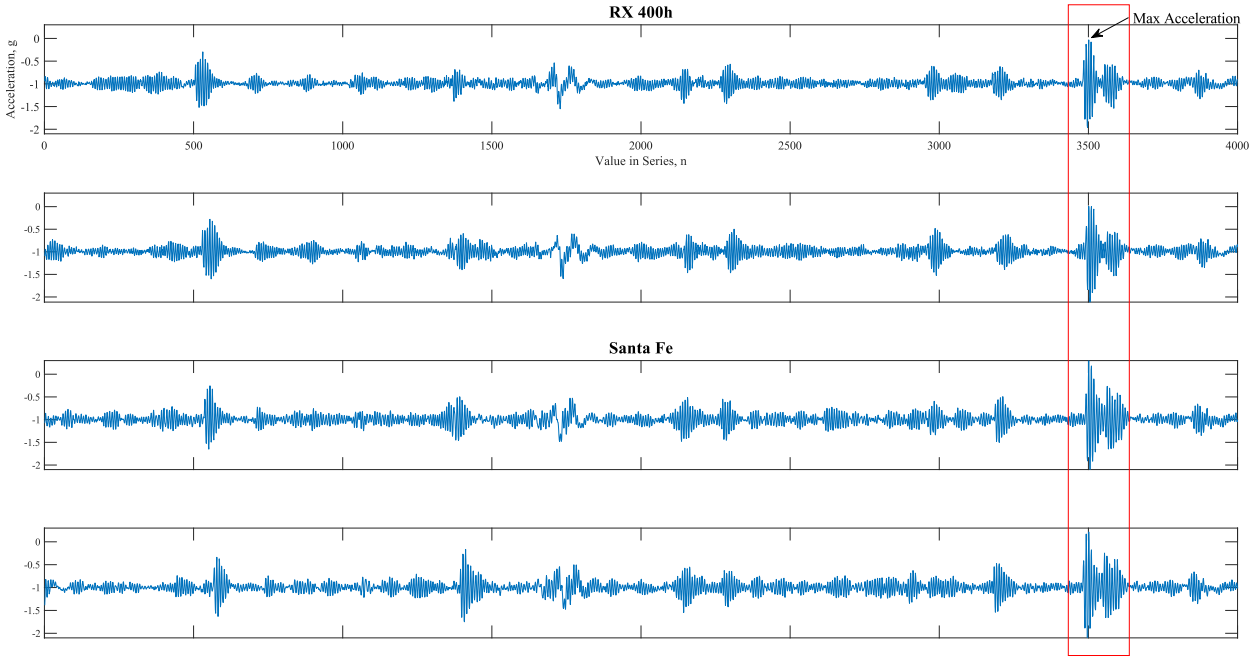

Figure 43. Comparison of acceleration signals for two different vehicles driving over the same stretch of a road twice

Taking an average of the four signals, we can see that most of the acceleration spikes that were present in the original plots are still present in Figure 44. They are still positioned near 500, 1400, 1700, 2200, and the maximum at 3500, but the low to medium amplitude noises between the higher amplitude peaks for each series in Figure 43 have, to some degree, been attenuated. The spikes in acceleration, corresponding to significant defects in the road, are present, but the amplitude of the vibrations between them is reduced for this acceleration average shown in Figure 44.

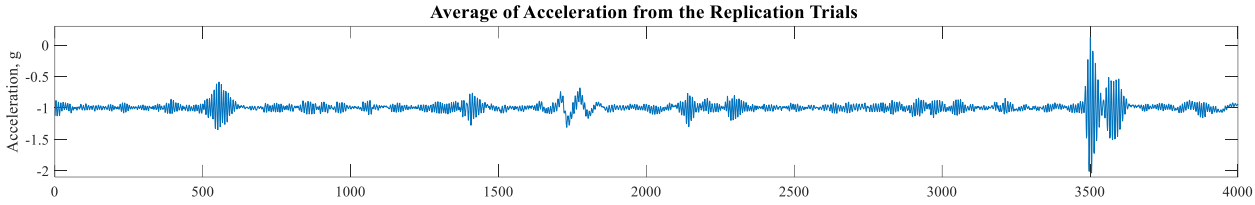

Figure 44. Average of the four plots in Figure 43

One way to show that the signal noise was reduced between the acceleration peaks is by comparing the variance between the averaged signal, shown in Figure 44, and the original acceleration signals,

shown in Figure 43. This comparison is provided in Figure 45, in which the variance of the acceleration signal, taken for segments of 100 values across the time series, yielded 40 variance values. Also, worth noting are the jumps in the variance that match the acceleration spikes from the previous two figures.

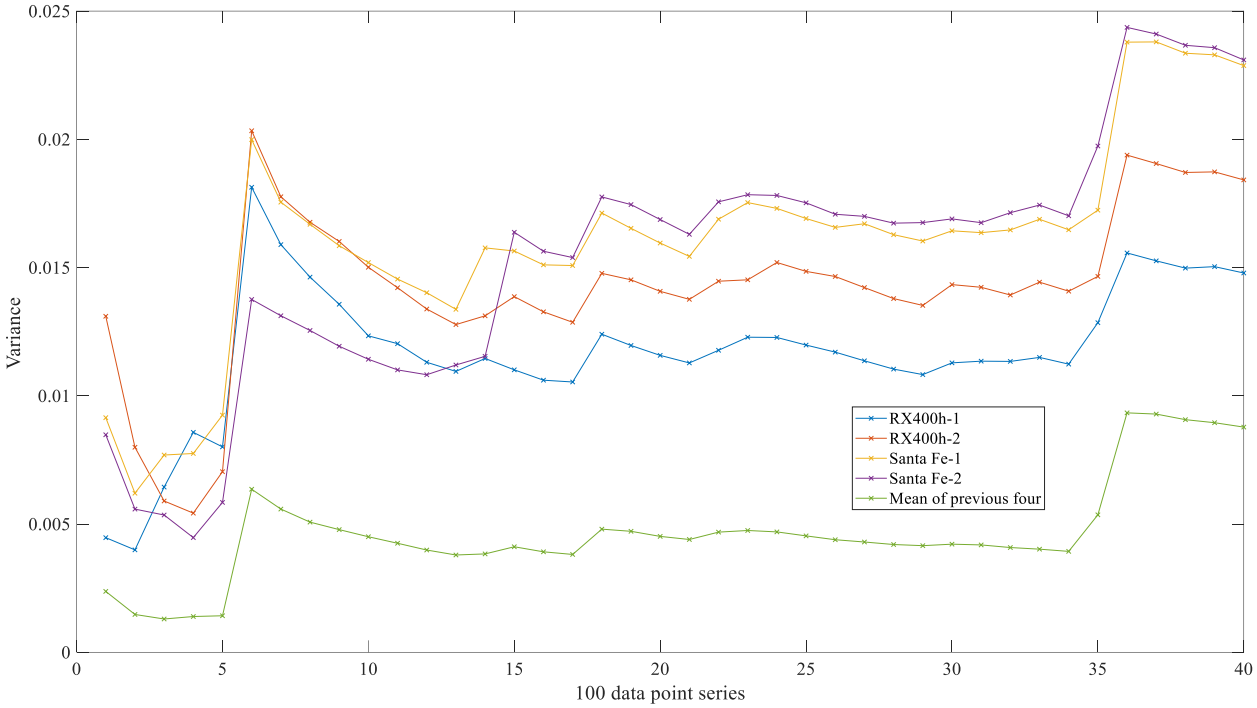

Figure 45. Variances of acceleration signals taken over 100-point range

Another method for quantifying the similarity between an ensemble of acceleration signals is with their correlation coefficient. Formally known as the Pearson correlation coefficient, it is a measure of their linear dependence between two values sets. For two sets of values, A and B, each containing N values, the correlation coefficient, ρ, for A relative to B is given by:

$$\rho(A,B) = \frac{1}{N-1}\sum_{i}^{N}\left(\frac{A_i - \mu_A}{\sigma_A}\right)\left(\frac{B_i - \mu_B}{\sigma_B}\right)$$

$$= \frac{cov(A,B)}{\sigma_A \sigma_B}$$

Where μ and σ, subscripted by A or B, are the mean and standard deviation of the datasets A and B. With those values, the correlation coefficient matrix, R, is constructed with all permutations of correlation coefficients relative to one another, producing:

$$R = \begin{bmatrix} \rho(A,A) & \rho(A,B) \\ \rho(B,A) & \rho(B,B) \end{bmatrix}$$

Since the correlation coefficient for a single set of variables, say $\rho(A,A)$ or $\rho(B,B)$, will always be equal to 1, we can simplify R to have only 1s on its diagonal (provided below). For perspective, it is helpful to know that the values in R will be contained in the range [-1,1], where ±1 indicates a perfect correlation between the value sets, and a correlation coefficient of 0 indicates that there is no agreement between the two sets.

$$R = \begin{bmatrix} 1 & \rho(A,B) \\ \rho(B,A) & 1 \end{bmatrix}$$

Table 13 shows the correlations coefficients for the four sets of acceleration values, with headings of Trial #1–4. As a comparison, a fifth acceleration time series has been included. This fifth series was captured while the Santa Fe Sport drove over the same stretch of road, but in the opposite lane because it was travelling in the opposite direction (denoted by "Opp. Lane" in column 7). The final column contains the correlation coefficients of the opposite lane time series, but all values in the series were flipped (i.e., the first value switches with the last value in the series, and so on) to see how well it would score with the main four trials.

The coefficients of trials #1–4 in Table 13 show a good positive correlation. This helps quantify the visual similarities that were observed in Figure 43. Though the values are closer to zero than to one, ranging from 0.186–0.388, they are orders of magnitude greater than the coefficients produced when comparing trials #1–4 to the set of acceleration values from the Opp. Lane column. Furthermore, there is additional but mostly weaker correlation agreement between trials #1–4 and the flip of the opposite lane values, which can be seen in the last column ofTable 13, ranging from 0.121–0.217.

Table 13. Correlation coefficients for different vehicles driving the same road section

		RX400h		Santa Fe Sport			
		Trial #1	Trial #2	Trial #3	Trial #4	Opp. Lane	Flip(Opp. Lane)
RX400h	Trial #1	1.000	0.276	0.388	0.231	-0.085	0.121
	Trial #2	0.276	1.000	0.234	0.268	-0.014	0.150
Santa Fe	Trial #3	0.388	0.234	1.000	0.186	0.027	0.217
	Trial #4	0.231	0.268	0.186	1.000	0.006	0.131
	Opp. Lane	-0.085	-0.014	0.027	0.006	1.000	-0.012
	Flip(Opp. Lane)	0.121	0.150	0.217	0.131	-0.012	1.000

Part of why the flip column shows good agreement with trials #1–4 is due to the nature of the road defects that caused the vehicle to experience the spikes in acceleration. They are mostly transverse cracks that span the entire width of the road, therefore, when the opposite lane pass is aligned with trials #1–4, the acceleration signals share most of the peaks, and the correlation coefficient between the five of them shows good agreement, especially when compared to the opposite lane values (column 7 of Table 13). The visual similarities can be seen in Figure 46, which shows the opposite lane acceleration time series plotted above itself, but mirrored, which is then plotted above the series, trial #3, which was captured by the same vehicle driving in the opposite lane. Similar spikes in acceleration are still present at ~1400, 2200, and 3500 for the bottom two plots, but not the top. This further supports the interpretation of the correlation coefficients present in Table 13 as being a sufficient proxy for similar acceleration signals.

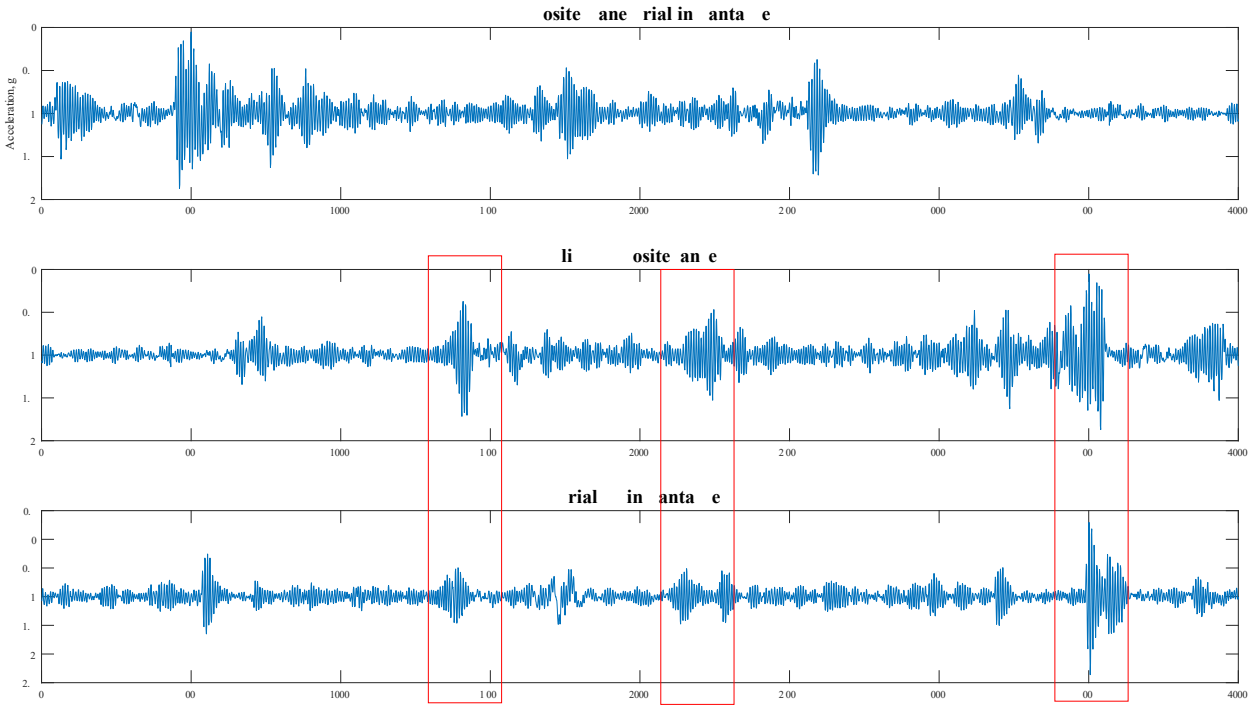

Figure 46. Opposite and modified acceleration series compared to a control trial.

Ultimately, the visual similarities in the plots of Figure 43 and the correlation coefficients shown Table 13 provide a satisfactory proposition that across different vehicles, the data collected is sufficiently comparable and replicable such that the data can be successfully compared and considered for detection of road defects and assessment of overall road roughness.

4.1.1 Replicability of Video Footage

Because the focal length of the main camera in most modern smartphones will permit a wider field of view than the width of the road being viewed, there were no major concerns that a different smartphone's camera would collect substantially different footage such that the image analysis results would be significantly different. There could be differences in the activity of cropping the frames extracted from the video footage, but that is a trivial consideration and a detail beyond the scope of this research.

Similarly, and also beyond this research, is the analysis of differences in colour representation between different smartphone camera sensors and their accompanying image processing software. For this research, the significance of these potential differences is especially reduced—the road surface is mostly grey (i.e., the RGB values are all similar in magnitude). Differences in RGB

values for each road pixel from different smartphone cameras would be minimal, and the exploration of the differences has not been considered for this research.

4.2 Motion Results

There were two goals with the acceleration data. The first was to show the quality of the road network's surface by mapping it in terms of a roughness-related variable. The other goal was to identify anomalies on the road surface.

4.2.1 Roughness Mapping

Figure 47 shows a map of all the driving routes taken in Thunder Bay with colouring based on the MMR value at each point of collected latitude and longitude. The colormap used ranges from green (smooth) to red (rough). The segments represent at least one second of driving time, and they show that most roads contain a mix of smooth and rough sections.

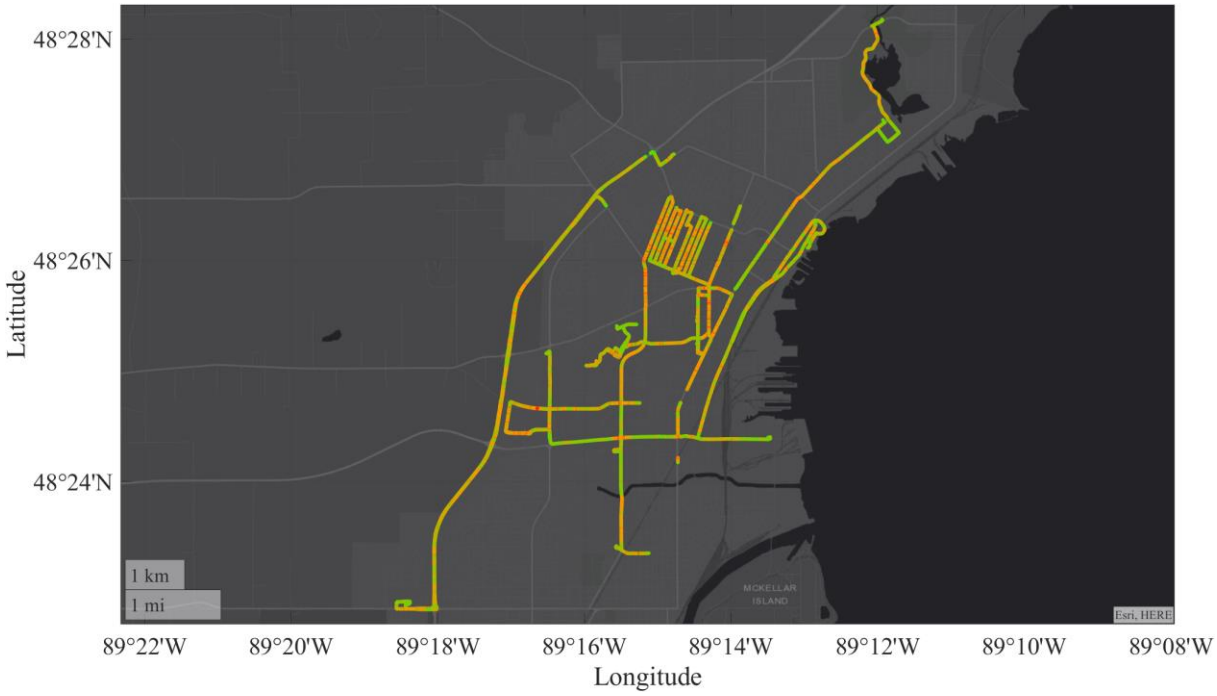

Figure 47. Roughness map of the selected Thunder Bay streets coloured using MMR intensity

One point of discussion is the use of MMR (recall: Minimum-Maximum Range) as the key roughness metric to visually enhance the map. The main reason is that the GPS and MMR values, that populate the map, were produced once per second. The amount of distance covered in that one second depends on the speed of vehicle, but that would mean about 8–12 meters of road at the test-

driving speed (about 30 to 45 km/hr). Another factor is that the acceleration response can change based on the speed, so the decision was made to make the plot based on the data points collected by time, as opposed to distance.

With that decision made, the challenge was to choose the actual roughness metric. The main comparison was between variance and MMR, shown in **Figure 48**. The plots show a lot of similarity, particularly in where the high-magnitude values are. The difference is that the variance, by its nature, will be close to zero in areas where there are small, brief spikes in acceleration. The MMR plot on the other hand, shows more peaks with prominence. This is notable in the MMR plot for values less than ~0.5 g.

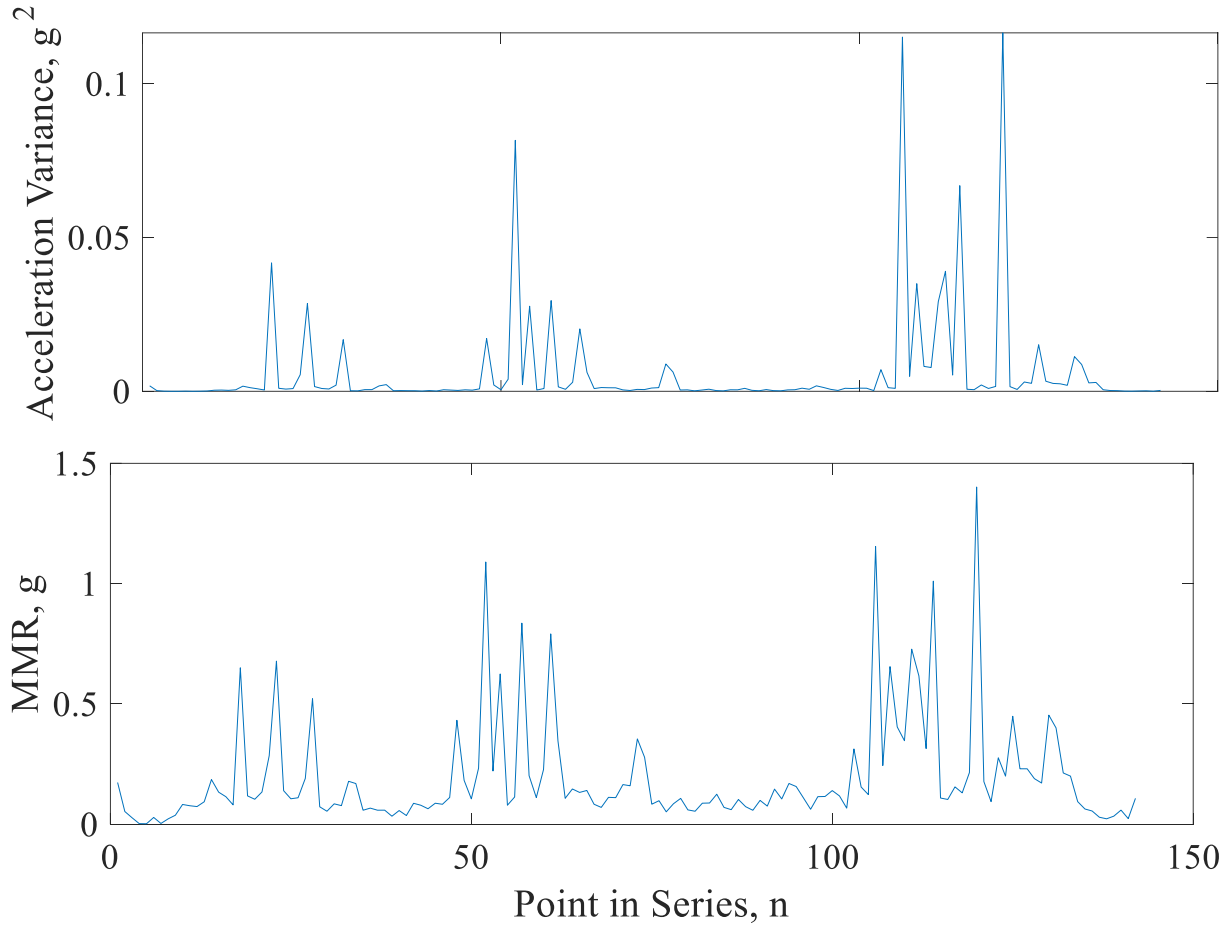

Figure 48. Comparison of variance and MMR as roughness metrics

4.2.2 IRI Challenges

The contributions of several researchers' work on roughness indexing are listed in Table 3. What they all have in common is some sort of ground truth for comparing their data to, and it is part of

what this research lacks. Some efforts have been made, in particular on a stretch of a road in the city of Thunder Bay that had been freshly paved during a data collection session. This is what the two plots in Figure 49 compare. In the upper plot, it is clear where the road transitions from old to new and back to old. There is much less variation within the range of 300–1400, and that same segment in the lower plot of the road before pavement shows much more vibration due to the road roughness.

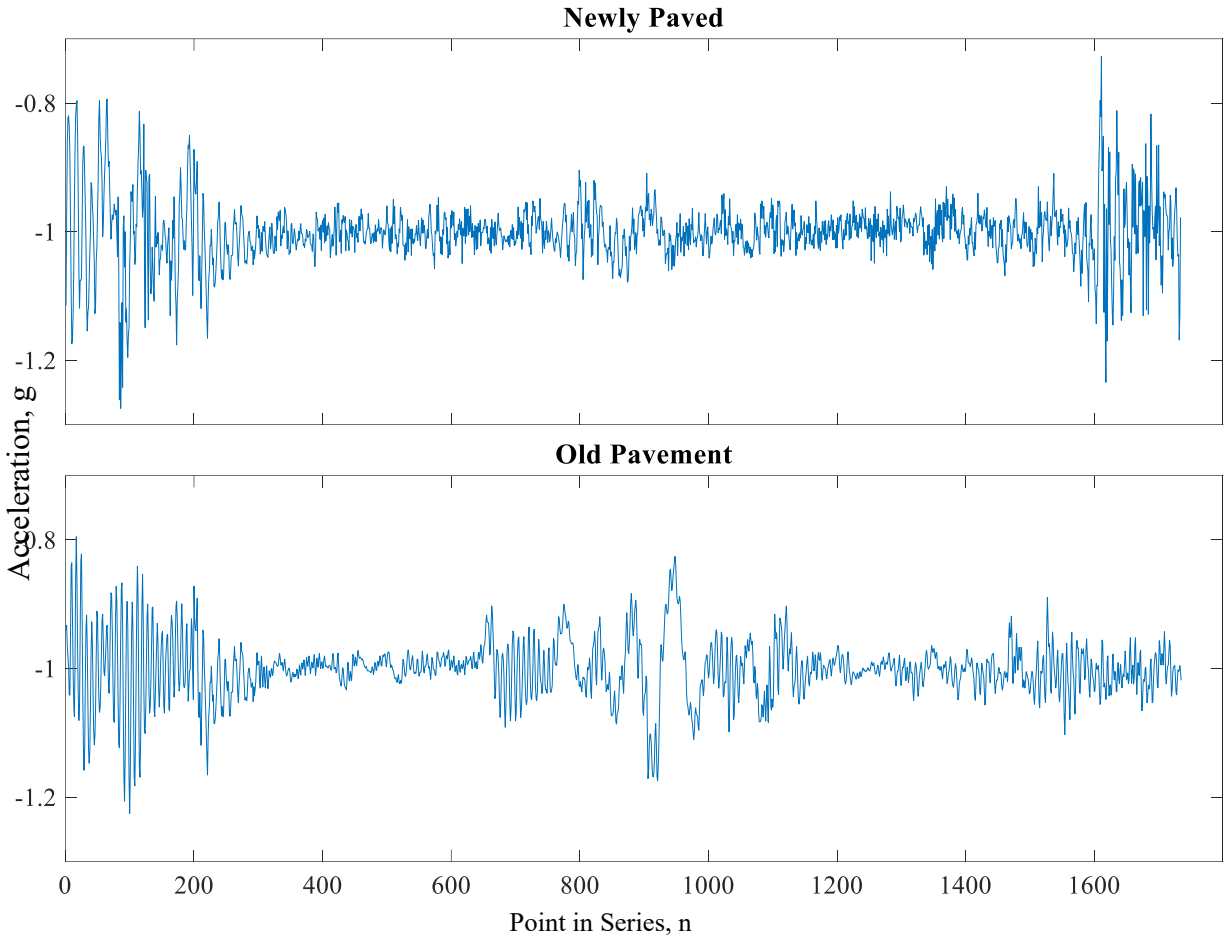

Figure 49. Plots comparing the same 20-meter stretch of a road before and after paving

It is difficult to make comparisons beyond qualitative remarks. Part of the reason for this is the setup of the smartphone itself when it collects data. Being mounted by an arm suctioned to the vehicle's windshield introduces motion behaviour beyond a simple response to the road surface. This is best exemplified in Figure 50, which shows the phones response to small taps while it is being held in position by the windshield mount. The initial spike in acceleration followed by a

decay back to baseline is the standard behaviour in damped free vibration. The problem with this is that the motion is influenced by the damping effects of the mount plus the minor instability of the mount's gripper, which add to the other mechanical damping influences from the vehicle itself.

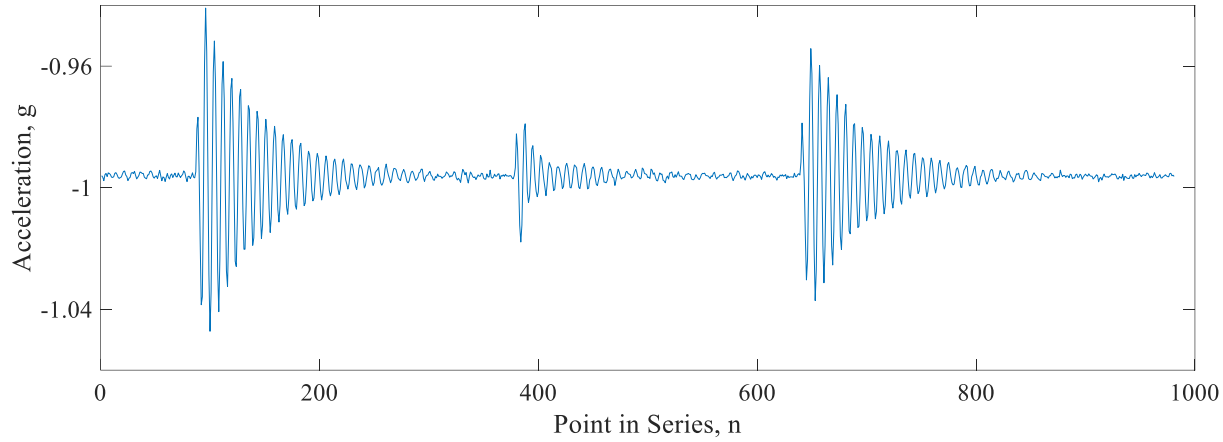

Figure 50. Damped free vibration behaviour noticed in the data collector

4.3 Vision Results

The objective of the analysis of the captured video footage is to identify defects that have been signalled as present by a spike in acceleration from the vibration data. This section covers the range of topics relevant to that objective. First it presents the results of the thresholding experiments, and it discusses why this method was not considered as part of the envisioned integrated system. Then the discussion moves to the training and refinement of the CNNs, including the data sets used and the results they produced.

4.3.1 Thresholding Results

In Section 3.4.2.2, we discussed the process of labelling images for training the CNNs. Besides training, the label arrays also act as a ground truth for evaluating the segmentation results of the trained CNN classifiers. Since the thresholding methods aim to segment the same defects labelled in the training images, they can also be used as ground truth data to evaluate the performance of the adaptive thresholding function that was described in Section 3.4.1.

Some adjustments had to be made to make the comparison between the label arrays, which were not intended to be used for evaluating the thresholding, and the masks produced by the adaptive threshold function. The main adjustment was to combine all defect-related classes in the label arrays into one class. This is necessary because the mask produced from thresholding is a binary

array: each value is either a one or a zero, indicating a defect present or not. The crack and pothole classes in the label arrays were combined for comparing to the masks; all the other classes were considered background. The threshold mask and label array were compared for each image in the test set (50 hold back images, discussed in Section 3.4.2.4). An example of this type of comparison is shown in Figure 51. It shows an image from the test set of images, which were also used to evaluate the CNNs, that demonstrates some agreement between the adaptive threshold in blue and ground truth in green.

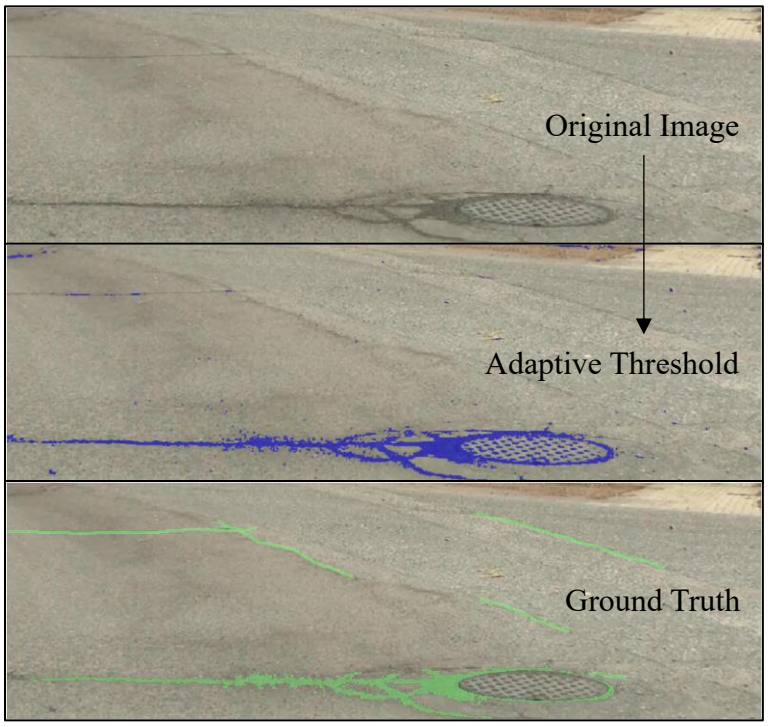

Figure 51. Comparing an image ground truth and adaptive threshold

The Jaccard index (sometimes referred to as intersection over union, IoU) can quantify how well the two image overlays match with each other. Mathematically, the Jaccard Index is the ratio of overlap to union between the segmented image labels and the ground truth. Given two sets of values, A and B,

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

In the above equation, A and B could represent a set of a single category, say ‘Crack’, for the ground truth and segmented image. In the case of the thresholding evaluation, set A would

represent the values that make up the blue mask in Figure 51, and set B would be the ground truth label in green. The Jaccard Index quantifies the overlap between the sets, thus a score of 1.0 would indicate a perfect segmentation by the adaptive thresholding function. A score of 0 would mean the segmented and labeled masks share no similarities, at least for that image. There is a drawback of relying too heavily on the Jaccard Index as a performance indicator: a high score is only possible if there is a near exact match to the ground truth labels.

The results from the evaluation of all 50 images in the test dataset are represented in Figure 52. It is a histogram of the Jaccard index values, showing their distribution in the range of zero to one. Only seven of the images scored a zero, meaning there was no intersection. Unfortunately, over two thirds of the samples had less that 20% overlap. On the other hand, even 20% overlap can be enough to show that a defect in the image has been detected.

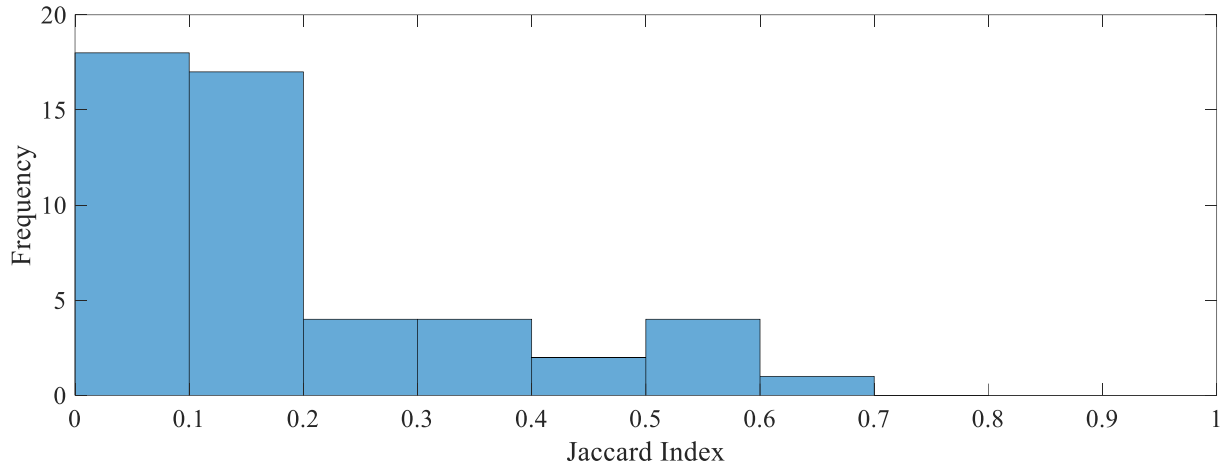

Figure 52. Histogram of results comparing thresholded image masks to the ground truth labels

4.3.1.1 Discussion on Thresholding

Of the available image segmentation methods, thresholding is the simplest method that exploits the difference in contrast often present at road defects. The basic idea behind using thresholding is that there is some pixel intensity value in the image beyond/below which the visual feature of interest can be segmented out. For example, Figure 53 illustrates how an original image of a paved road with cracks is enhanced as a grayscale image from which a threshold value can be set in order to create a mask over the pixels that exceed the chosen threshold value.

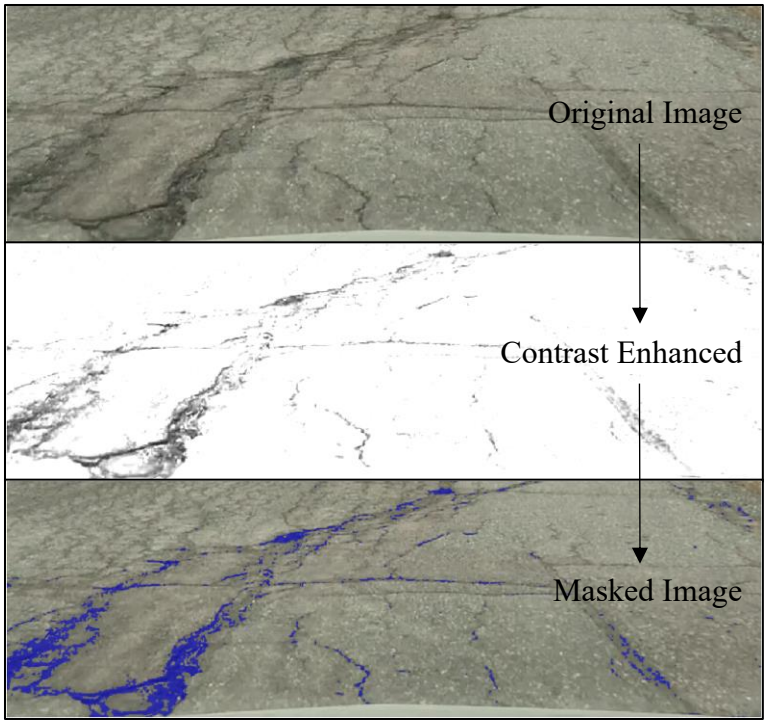

Figure 53. An example of image segmentation with thresholding

There are several advantages to using thresholding for image segmentation. It is an easy process to initialize and develop. The basic steps are to convert the image to greyscale, set a threshold value, and visually confirm that the segmentation is acceptable. Within those steps, the image manipulation and processing are fast and not computationally-intensive, namely when they are compared to other methods such as classification by a convolutional neural network. When compared to other high-level classification approaches, thresholding has two comparative advantages: easy model iterations and understandable results.

Because the image manipulation and processing are simple, it is easy to iterate on the existing model in an effort to marginally improve the segmentation results. The final result from thresholding, i.e., a mask placed over the image that indicates where the features of interest should be present, is easy to understand. If the mask is too large, lowering the pixel intensity threshold will reduce the size of the mask, and vice versa. This straightforward relationship between the intensity parameter and the image mask is lost with more advanced methods for segmentation and pixel classification.

4.3.1.2 Thresholding Drawbacks

There are, however, drawbacks to using such a simple approach to segmentation, including:

1. Information loss—the RGB data for the image is reduced from three color channels (RGB) down to one channel (grey).
2. Lack of fine tuning—there's only one parameter, intensity, that can be adjusted to produce different results, though preprocessing the images in different ways can expand possibilities.

Ultimately, and especially for the image data used in this research, which included variable environmental conditions, thresholding should not be considered the main way that image features (road defects) are segmented in an image, but rather thresholding is a versatile and easy to understand first step toward good segmentation results.

Reasons to move to more advanced methods

1 A practical system should be adaptable, but thresholding requires somewhat consistent images. Any background pixels, coming from grass or other non-road-surface features contain darkness that contaminates the thresholding method since it will take those pixels instead and use that to create the mask.

2 No way to distinguish features (not easily, at least). We can identify a defect and determine its size on the road. Distinguishing between cracks and potholes may be achievable, but between patches and potholes would be more challenging. Also, not much room for the so-called false positive bumps (e.g., grates and manholes).

3 Deep learning models take a lot of work upfront, but then can be improved with relative ease.

4 Added benefit of CNNs vs thresholding is extra colour channels to distinguish between regions of similar texture.

4.3.2 Evaluating CNN Performance

Before presenting the results produced by the different CNNs that were developed for this research, this section will cover the data that was used to train the CNN models and the metrices used to evaluate them. Hopefully, this will help further contextualize the results presented in Section 4.3.3.

4.3.2.1 Distribution of Labelled Data

In order to fully appreciate the results produced by the classifiers, it is helpful to understand the data they are based on. There were 658 labelled images in the training set. We created the final labeled training set by manually refining the labels on some of the images. Many of them were due to small labelling errors, such as having a few pixels in the image unlabelled and thus considered as an "Undefined" class in the classifier. The final iteration of label refinement was the main source of data for this discussion, though it was not too different from the original label set. Out of the 140 million pixels analyzed, only a few thousands were changed during this manual review process.

There are two ways to consider the pixel label distributions. The labels can be counted by the number of pixels themselves or by their instance (i.e., is there at least one instance of the class in the image). Counting by pixel gives a realistic sense of how the data is distributed, which is important for understanding CNN model training and results, but counting categories by instance is more useful for understanding the prevalence of defects in the training data, and more broadly on the road network itself.

Table 14 shows how the classes are distributed across the total set of training and test images. These samples included sufficient amount and variety of road damages that it seemed worthwhile to label them and use them to train CNN classifiers. Though they were not representative of what all frames from a video would follow as a distribution. Likely the labelled pixel distributions would be even more heavily weighted toward the Road and Background classes. The distribution of pixels in the test dataset is fairly close to that of the training set, showing that it should produce representative results.

Table 14. Distribution of labels across training and test dataset images (all values in %)

Image Set		Road	Background	Crack	Patch	Pothole	Manhole	Grate
Train	Pixels	87.0446	7.0049	3.5127	1.8899	0.1779	0.3473	0.0227
	Instances	100.000	81.0030	89.2097	33.7386	26.5957	20.2128	3.4954
Test	Pixels	86.1750	7.1958	2.4758	3.2874	0.3001	0.4758	0.0904
	Instances	100.000	86.000	86.000	30.00	32.000	28.000	10.000

4.3.2.2 Evaluation Metrics

Before moving on to discuss the results produced by the different CNN classifiers, there are a few more metrics for evaluating the performance of the classifiers that should be discussed. The Jaccard index (also known as IoU) has already been discussed in Section 4.3.1.

The fundamental parts of understanding the performance of these classifiers, or any other, are the four possible classification outcomes: True positive (TP), true negative (TN), false positive (FP), and false negative (FN). Together, they exhaustively account for the possible outcomes of a classification problem in supervised learning.

Consider the simplest classification problem, binary classification, where each piece of data is labeled as one class or another, say class A and class B. In this scenario, when data is given to the classifier, there are two possible input labels and the classifier assigns the data one of two possible labels, and thus there are four possible outcomes. These outcomes are visualized in Figure 54. Together, the classification conditions can be arranged to construct a confusion matrix, which shows how the input labels (the expected classes) are mapped to the output labels (the predicted classes) by the classifier.

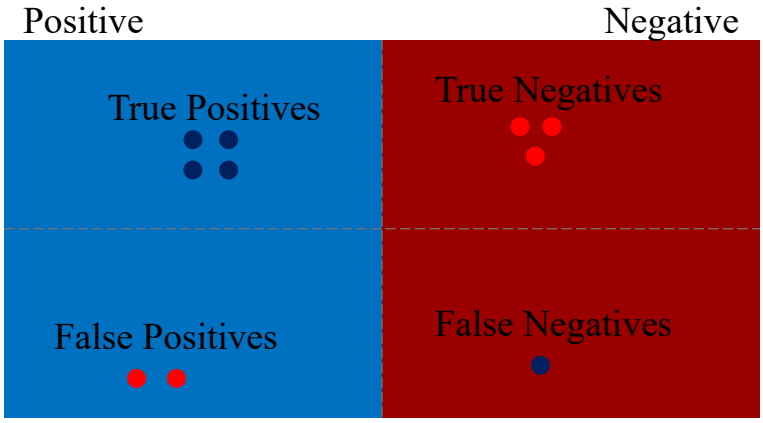

Figure 54 . Possible outcomes in a binary classification problem

We can calculate the precision and recall for the classifier's performance presented in Figure 54. Precision is the true positives divided by the sum of the true and false positives, which serves as measure of how well the classifier is able to correctly make classifications. Recall answers the question of how correct the classifier could have been. It is the fraction of true positives divided by all values that should have been positive.

$$Precision = \frac{TP}{TP+FP} \quad Recall = \frac{TP}{TP+FN}$$

Understanding these quantities is important because they are used to make up other evaluation metrics that will be used to show the results. One of these metrics, which is based upon precision and recall is the BF Score. It is another way to describe how well a prediction, such as the output of a classifier, matches a ground truth for the same data. Compared to the Jaccard index, the BF Score places more value on true positives. This makes it a good candidate for evaluating results that are being averages, such as with the results of the CNNs.

$$BF\ Score = 2 \cdot \frac{Precision \times Recall}{Recall + Precision}$$

One other metric, the Dice score, could be used in the following analysis. Formally known as the Sørensen-Dice similarity coefficient, the Dice score is another measure of similarity between the predicted labels and the ground truth. It is related to the Jaccard Index by:

$$\text{Dice}(A,B) = \frac{2 \cdot \text{Jaccard}(A,B)}{1 + \text{Jaccard}(A,B)} = \frac{2\,|A \cap B|}{|A| + |B|}$$

And due to its similarity with the Jaccard index, it was not used in the following analysis of the results. This can be seen in Figure 55, which shows the distribution of scores in three adjacent histograms. The data for these histograms are the comparison of the ground truth labels of the 'Crack' class and the segmentation produced by one of the first trained CNNs, VGG16-1. In Figure 55, the Jaccard index and Dice score follow fairly similar trends. Alternatively, the BF score shows a much different distribution, heavily skewed to higher values.

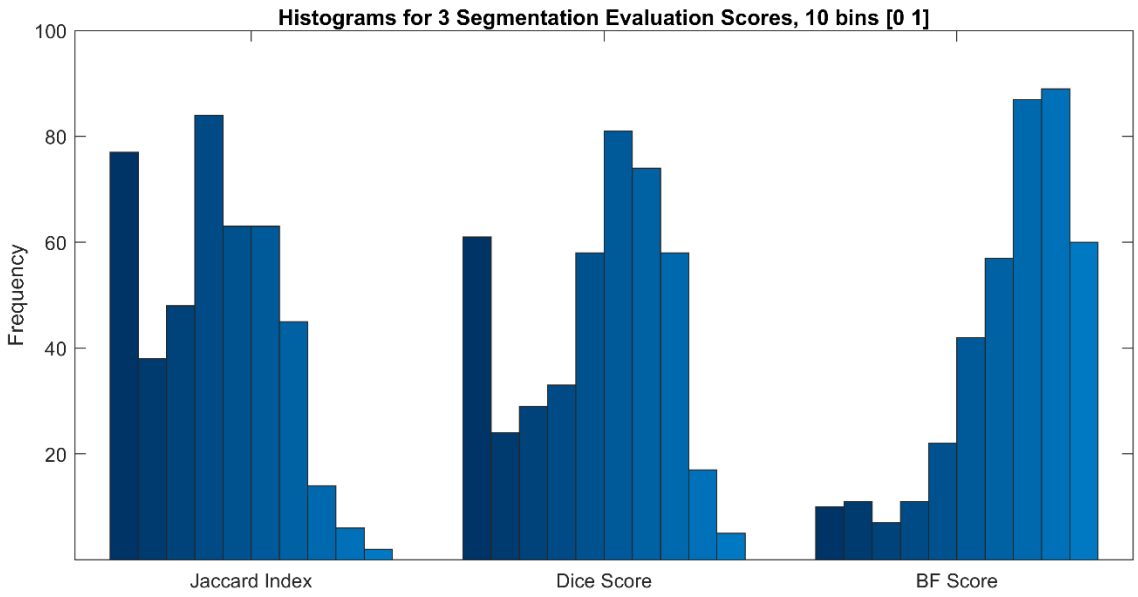

Figure 55: Comparing scores for the Crack class in the image training set

There are four other metrics used for evaluating semantic segmentation and "scene parsing" (Long et al. 2015). They are based on pixel accuracy and are used for overall evaluation of CNNs in Table 15 and Table 17.

1. Pixel Accuracy: $\frac{\Sigma_i n_{ii}}{\Sigma_i t_i}$
2. Mean Accuracy: $\frac{1}{n_{cl}} \cdot \frac{\Sigma_i n_{ii}}{t_i}$
3. Mean IoU: $\frac{1}{n_{cl}} \cdot \frac{\Sigma_i n_{ii}}{t_i + \Sigma_j n_{ji} - n_{ii}}$
4. Frequency Weighted IoU: $\frac{1}{\Sigma_k t_k} \cdot \frac{\Sigma_i t_i n_{ii}}{t_i + \Sigma_j n_{ji} - n_{ii}}$

Where i and j are classes, and n_{ij} is the number of pixels from i that are predicted as j. There are n_{cl} classes, and the number of pixels that belong to class i is given by $t_i = \Sigma_j n_{ij}$.

4.3.3 CNN Results

In total, nine CNNs were trained and refined on the training data to produce classifiers that could segment images into the seven different classes found across the road images. Two of the nine CNNs were trained on data using modified label data. This process is discussed more in Section 4.3.3.1. For now, the results of the first seven CNNs are discussed in this section. Their results are presented from two different perspectives. The first is how the classifier performs overall, and the second is how the classifier performs for the individual classes that it is segmenting.

Table 15 shows the different measures of overall model performance. The first column is the time it took for each classifier to segment the 70 images of the test set. All four of the VGG architecture models take an order of magnitude longer than the other three, but their performance is not drastically better. Global accuracy is a measure of how well each classifier performed across every pixel that makes up the test set. These results are misleading because two classes ('Road' and 'Background') make up over 90% of all pixels, so as long as these two classes are well classified across the test set, the global accuracy will be high.

The mean accuracy counts all of the correctly classified pixels per class and divides that by the total pixels. Then it is averaged over all the classes. Performing poorly at one or two classes can lower this substantially. Figure 56 shows the BF scores for each of the CNNs listed in Table 15. We can see that the ResNet50s have the highest score across all categories and, relatedly, also have the highest mean accuracy scores. The mean IoU is the average of the Jaccard index (aka IoU) for each class across all of the images, similar to the mean accuracy. The weighted IoU is the same as the mean IoU except that the IoU average for each class is weighted based on the number of pixels for that class across the data set. Like the global accuracy, this misrepresents overall model performance because the pixel distribution of the test set is highly imbalanced.

Table 15. Results of the classifiers trained using the original label definitions

	Time (sec)	Global Accuracy (%)	Mean Accuracy (%)	Mean IoU (%)	Weighted IoU (%)	Mean BF Score (%)
VGG16-1	445	90	48	36	83	67
VGG16-2	337	94	52	47	89	63
VGG19-1	447	95	59	53	91	73
VGG19-2	432	95	40	37	90	65
ResNet50-1	46	95	73	62	90	82
Resnet50-2	47	96	76	68	93	85
MobNetV2	20	94	63	54	89	78

The general background of the CNN architectures presented in Table 15 have been discussed in Table 11. How the iterations of the trained models differ is listed in Table 16. The main two points of differentiation between the different models are the number of images used to train them and

how many classes are present in the training data. The appendage of -1 or -2 is used to indicate that the model has been iterated on. For example, VGG16-1 was the first model trained for this research. It was trained on 445 images, and the results were promising enough to warrant training more models on a larger set of training images, totalling 600. This led to the creation of the second VGG16 model, named VGG16-2, and three other models, VGG19-1, ResNet50-1, and MobNetV2. In an effort to improve model performance on the pothole class, we collected more video to add 58 more images with potholes to the training set. These were used to train VGG19-2 and Resnet50-2. The last two models listed in Table 16 are discussed in more detail in Section 4.3.3.1; they were trained using one less class in a hope to improve performance.

Table 16. Differences in training of CNNs evaluated for defect detection

CNN	Training Images	Classes
VGG16-1	445	7
VGG16-2	600	7
VGG19-1	600	7
VGG19-2	658	7
ResNet50-1	600	7
Resnet50-2	658	7
MobNetV2	600	7
ResNet18	658	6
ResNet50-3	658	6

Figure 56 shows a different view of the performance of the classifiers. Each plot is titled for a different type of CNN (e.g., VGG16, VGG19, etc.) and displays the BF score performance for each class as opposed to for just the entire CNN. Where there are two bars in the plot, the light blue one on the right is for the model that was updated with additional training images. In Table 15 this is distinguished by placing either a one or a two at the end of the model name. The MobileNet classifier was only trained once, on the full set of training images.

All of the classifiers except for ResNet are not able to identify a grate in the test set (there are five). Part of the problem is that the training data did not contain many instances of the 'Grate' class. Section 4.3.3.1 discusses how to deal with this problem.

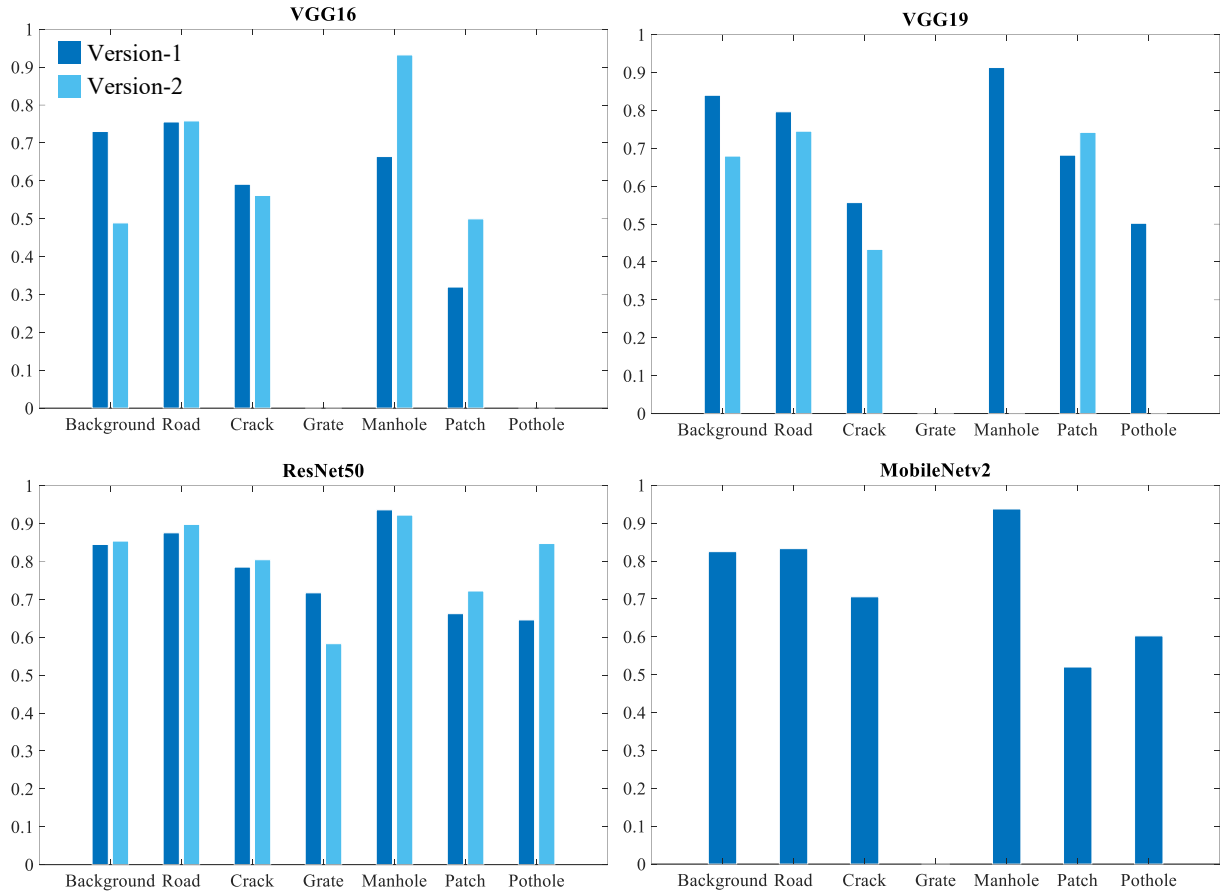

Figure 56. Class BF scores for different CNN architectures

4.3.3.1 Class and Label Refinement

Based on some of the challenges faced with the original seven classifiers, some refinements were made to the data used to train them. The most obvious problem that can be seen in Figure 56 is the poor performance of the 'Grate' class for all the classifiers except the ResNets. Because there are so few of these objects in the training data, the classifiers struggle to identify them. The decision was made to merge all instances of 'Grate' into the 'Manhole' class. Other changes made to include labelling portions of vehicles caught in the frame as background. This would, hopefully, mean that the class for the plain road surface would be more consistent. Another problem we identified was due to paved areas adjacent to road after the concrete curb. Originally this was included in the

background label, but it was switched to road so it would be more consistent with the hue and texture of the already labelled road class. The results of the two classifiers trained on the modified training data are shown in Table 17. The ResNet50-3 show improvement over ResNet50-2 by every measure. The Resnet18 model also performs better than most of the original classifiers.

Table 17. Results of the classifiers trained using the modified label definitions.

	Time (sec)	Global Accuracy (%)	Mean Accuracy (%)	Mean IoU (%)	Weighted IoU (%)	Mean BF Score (%)
ResNet18	36	94	73	63	89	77
ResNet50-3	48	96	73	67	92	83

The class and label refinement shows improved results on the basis of per-class performance too. Figure 57 shows the BF scores for the ResNet18 and ResNet50 models for each of the classes, now totalling just six because of the removal of the grate class. Compared to the ResNet50 performance in Figure 56, the performance of the manhole class has gone down significantly. This is due to it also absorbing the grate labels. Despite the lack of samples of the grate class and some classifiers' poor performance of identifying them, they are important features to identify in the images because they can trigger the anomaly detection threshold for defect detection.

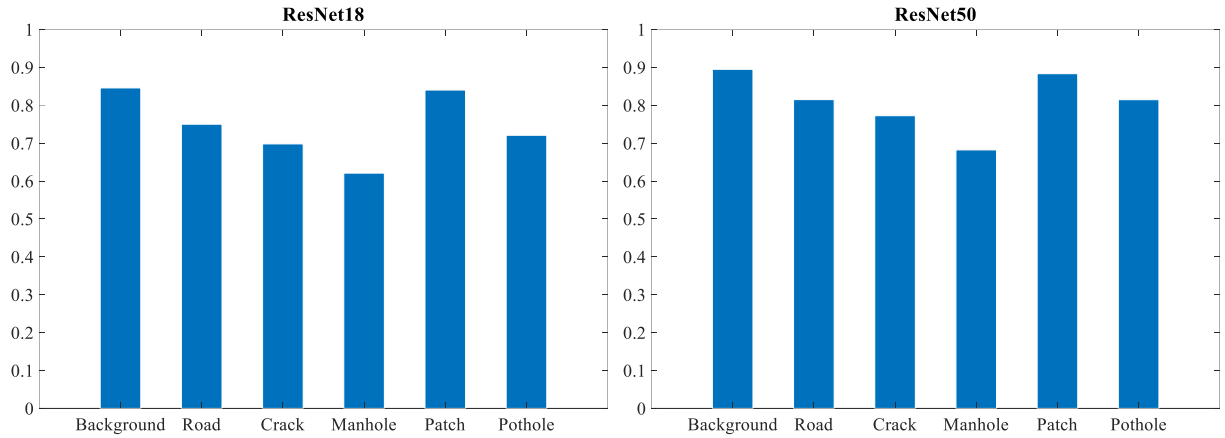

Figure 57. Class BF scores for CNNs trained on modified training data

4.3.3.2 Discussion of CNN Results

CNNs are large, complicated networks of abstract layers that manipulate the pixels of an image. Unlike the results of a thresholding-based approach to segmentation, their results are more difficult

to directly understand. Still, there are a few trends in the data that can be understood. The first is that growing the training set with more labelled data has an impact on performance. The results of ResNet50 in Figure 56 improve substantially at classifying pixels labelled as potholes. They were an under-represented class in the original training set, and many extra examples were sought out to improve performance.

Another trend is in the time it takes for each classifier to segment an image. After all, segmentation accuracy is not the only important result. The speed at which the results are delivered matter, too. For future deployment of a system like this, the time it takes to segment an image could be important. For this work it was not important, because all of the processing was done on a computer after the data had been collected. But if this were to move to a closer-to-real time system, which is feasible, a ResNet style CNN would make for the best choice because of its strong performance and speed. It can process an image in less than a second, and with good accuracy across all classes.

Table 18. Overall performance metrics for a ResNet50 classifier.

Global Accuracy	Mean Accuracy	Mean IoU	Weighted IoU	Mean BF Score
96.7%	50.4%	39.2%	94.8%	61.0%

The last important observation of the results is the distinction between ground truth accuracy and identification accuracy. Ground truth accuracy refers to how well the classifier performs at segmenting the image to match the labelled image used for testing. We can say that identification accuracy is a measure of how well the different classes are identified in the image, even if they do not perfectly match the ground truth. This is important because the objective of this research is to detect and identify defects in the road, not to perfectly match them with the labelled images. It does not need to be perfectly segmented in order for it to be useful in the PMS. This is shown in Table 19, where the precision rates are higher for the defects present in Figure 58 (pothole and crack but the recall rates are much lower.

Table 19. Class-based evaluation metrics for an image segmented by the ResNet50 classifier.

	IoU	Precision	Recall	BF Score
Road	0.9554	0.8974	0.8541	0.8752

Background	0.8926	0.8046	0.7787	0.7914
Pothole	0.3127	0.9317	0.511	0.6601
Patch	0	0	NaN	NaN
Manhole	NaN	NaN	NaN	NaN
Crack	0.2587	0.8755	0.7838	0.8271

Figure 58 shows how this distinction applies to the detection and identification of defects on the road surface. There are several differences to notice between the ground truth image (the middle frame) and the image segmented by the ResNet50-3 classifier (bottom). However, there is not much difference between the boundary between the road surface (orange) and the background class (dark blue). The classifier was able to segment the gravel shoulder and label it as background. Both classes score high in all the class-based metrics, seen in Table 19, used to evaluate the ResNet 0 cla ssifier's performance on this image.

Next, we can compare the differences between the ground truth and segmented image for the crack and pothole classes (shown in pink and red in Figure 58). In Table 19, they both show strong precision but only moderate recall. This is because nearly every pixel that was segmented as either a crack or pothole was done so correctly, but there were a lot more pixels that could have been positively identified with those classes. The pothole class shows especially bad recall because it missed an entire cluster of pixels in middle of the image. Comparing the two coloured frames, though, the segmented one does capture the presence of most defects well enough. Most major cracks and potholes are clearly segmented. One final difference between the two frames: the segmented image shows light blue in several small areas, indicating that the pixels were segmented into the patch category. Arguably, the ground truth image should have some of those regions labelled as patches as well. This minor inaccuracy in the ground truth image is less important here because our objective is to identify defects in the frame. If we also wanted to attribute the defects in the frame to the acceleration peaks, then we would have to make sure every ground truth image was labelled with even more specificity. We used the BF Score because it weighs the true positives higher than the other discussed evaluation metrics, though it is still not a perfect measure of how well the classifier is performing at the task of sufficient defect identification.

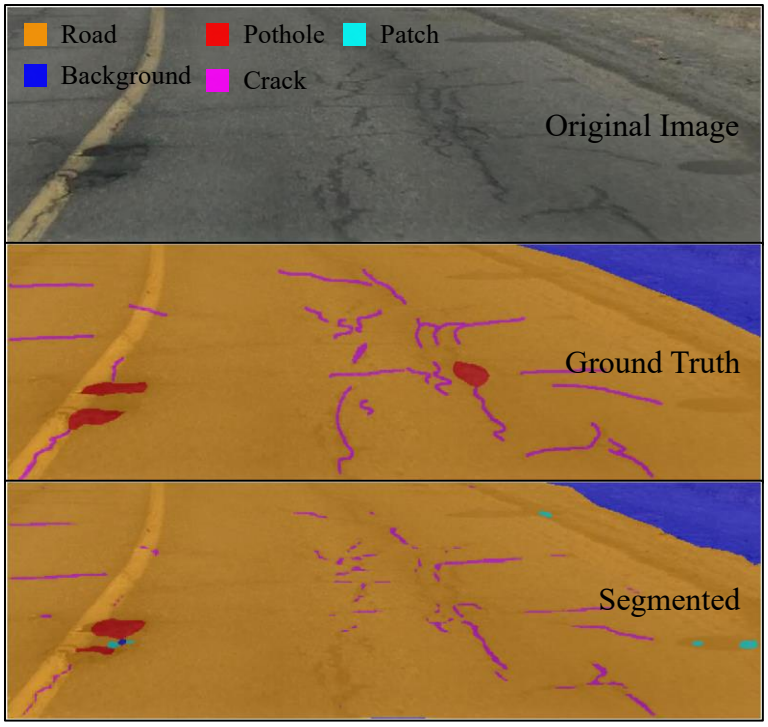

Figure 58. Example of a segmented image with low BF score

4.4 Integration Results

It is important but challenging to evaluate how all the functions work together to integrate the data, as discussed in Section 3.5. It is challenging because there is no clear way to evaluate the performance in an objective manner. If we integrate the data collected on some stretch of road, the output of that will include a defect list and as many segmented images as the length of the defect list. The lack of objectivity for evaluating the defect list and those images stems from the difficulty of establishing a ground truth. However, we can evaluate the system in a subjective manner by performing the data integration for a stretch of road, and then manually reviewing the video footage to validate the results and observe how many defects were missed.

For a stretch of road being reviewed, this approach to evaluation has three main steps:

1. Run the data integration functions, which will record defects along the stretch of road
2. Manually review the video, record the number of defects in the vehicle's lane of travel
3. Compare the difference between the results in Step 1. and 2.

Doing a manual review process allows for the data integration quality to be assessed in two ways. The first assessment is to see how the defect locations identified with the FindFrame function

compare with the number of defect locations counted during the manual review of the video. The second assessment is based on how well the classifier performs at identifying defects within the extracted frames. If the number of defect locations identified and the recognition of the defects at those locations both show good agreement with the manual review, then the data integration will be considered successful. The ResNet50-3 model was used for all image segmentations because it showed some of the best performance in Section 4.3.3.

4.4.1 Integration Example 1

For this 1.5 km section of road, the FindFrame function identified 14 points in the acceleration time series that had high enough magnitudes to warrant extracting the frames and running them through the SegSave function, which created the list of defects on this section of road. Across all of those defect locations, the SegSave function produced a list of 25 total defects. Manually reviewing the footage identified 21 locations in the video where defects were present, and 34 total defects. This implies that, for this example, there are 50% more defect locations and nearly 40% more total defects identified when manually reviewing the video footage as opposed to using the anomaly detection and segmentation functions. There are several reasons for the discrepancy. The first can be seen in Figure 59, which is a snapshot of the video where the vehicle is about to drive over a transverse crack that crosses the entire road. The challenge for detection is that the crack is not wide or deep enough to illicit an acceleration response beyond the threshold used in FindFrame. The acceleration value corresponding to that time is –0.92g, which is not even halfway between the mean (–1g) and the threshold (–0.8g). If the threshold were lowered to accommodate defects like the crack shown in Figure 59, there would be well over 100 defect locations identified on this 1.5 km stretch of road, nearly five times more than the number located in the manual review process.

Figure 59. A difficult to detect transverse crack in the road

The other common reason for the discrepancy between the number of defects is simply that not all defects are driven over by the test vehicle. If they are not driven over, then there is no substantial change in the acceleration to initiate finding the video frame for that point in time. In Figure 60 we can see all of the defect locations along the route marked by either a blue or red icon. Those marked in blue were placed by uploading the KML file created using the MakeKML function, and the ones marked in red were manually placed at the locations where defects were missed by the data integration functions.

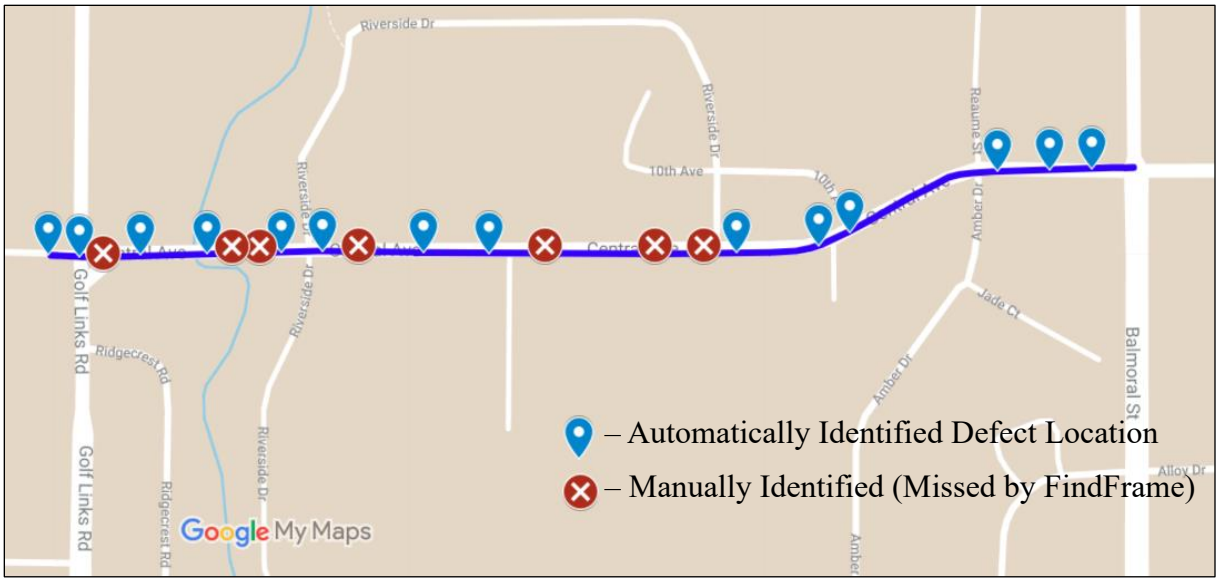

Figure 60. Automatically and manually identified defect locations along a short route

According to the defect list created by the SegSave function, there are three anomaly types found across all of the defect locations on this stretch of road. A manual review of the video footage confirms this. They are the crack, manhole, and patch classes. Figure 61 shows how the different defects were identified, and if they were correct. The bar representing the crack class shows that all cracks that the system identified were correct, but several were missed. The value of the missed bar (light blue) came from the manual review of the video. The misses are largely due to the cracks being too small to produce much of a response in the acceleration signal, as was shown in Figure 59. The manhole class follows the same composition, where there were no false positives, but there were some false negatives uncovered from the manual review. The missed manholes were due to the test vehicle not driving over them. Finally, the patch class shows a combination of incorrectly identified defects and missed ones. Several of the false positives were due to the wet spots on the road, which resemble patches in that they appear darker and smoother than the surrounding road surface.

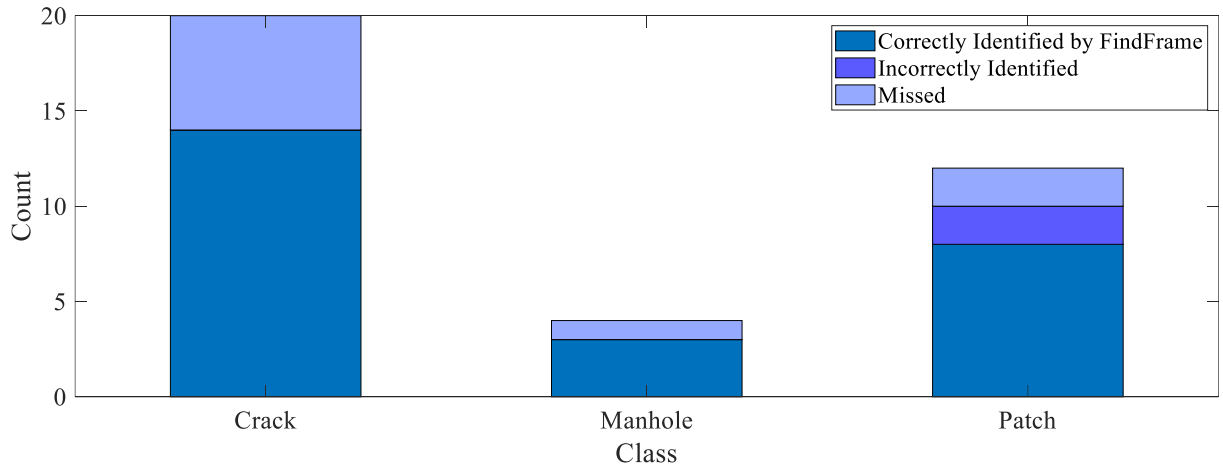

Figure 61. Combination of defects identified automatically and manually

4.4.2 Integration Example 2

In this example, we identified 93 defect locations during a manual review of the video. The stretch of road is 2.1 kilometers long, meaning there is an average of 44 defects per kilometer, which is a significant jump from the 14 defects per kilometer in integration example 1.

This stretch of road was challenging to manually review. The difficulty came from the judgements that must be made around areas where there was a high frequency of defects, or where the defects extend beyond just one frame. An example of this can be seen in Figure 62, where there are multiple types of cracks, all connected, and they extend beyond the ROI frame in which segmentation occurs. Different reviewers could reasonably conclude that there are several different ways to quantify the number of defects, which is why we chose to count defect locations. That way, several frames from now, when the video shows a new view of the same cracks from Figure 62, they will not be double counted, as they are all part of the same defect locations.

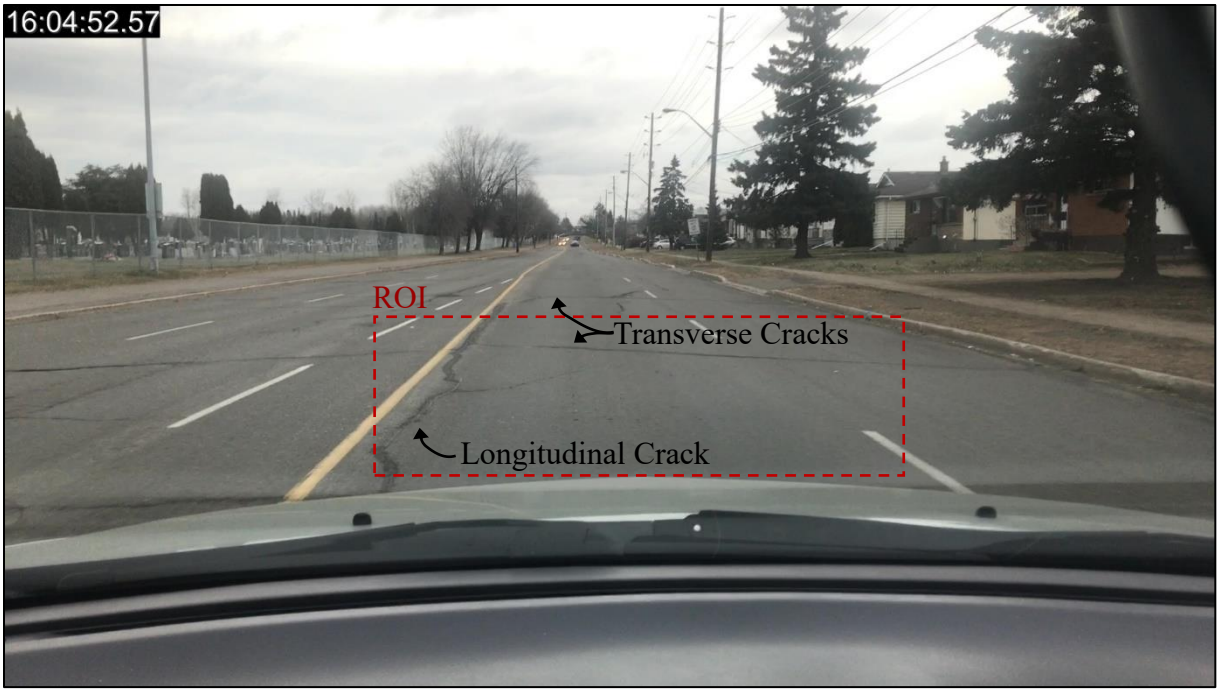

Figure 62. Cracks in the road that could cover multiple extracted frames

Then the problem is to decide how expansive a defect location can be in terms of the video footage. Multiple rounds of video review showed that, for these frame sizes and driving speeds, a new defect location comes into view at most once per second. Any sooner than that, and the defects will end up being double counted. In Figure 63 we can see how many defects the FindFrame function would count based on how quickly a new acceleration value exceeding its threshold occurs. When the time between peaks is set to zero, it counts 677 defect locations (i.e., this acceleration time series contains 677 instances of acceleration greater than or equal to –0.8g). As the function is forced to wait more time before it can count a new defect location, the number of locations counted drops. When the time between peaks is 1-second, the number of defect locations counted is 81, which is relatively close to the manually counted number of defect locations of 93.

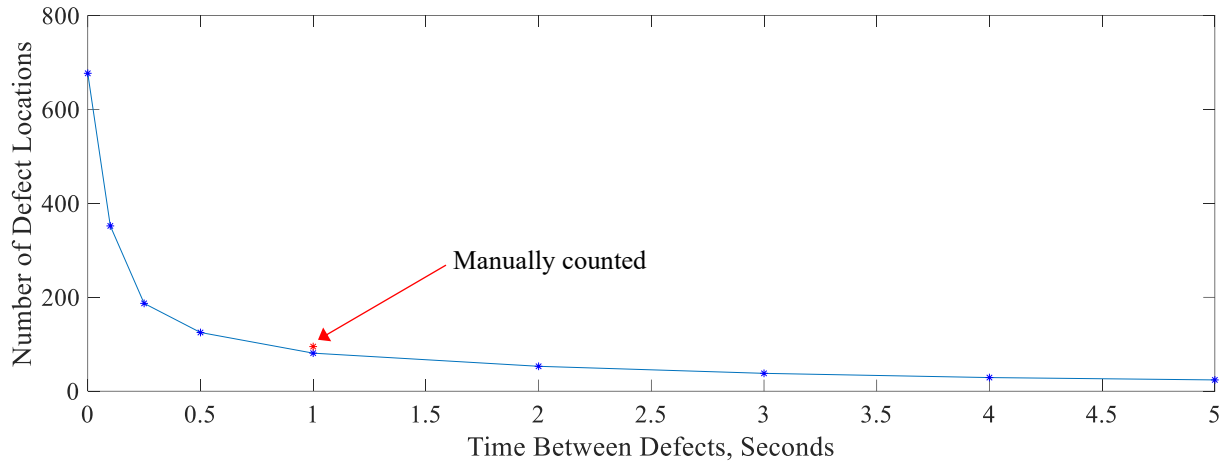

Figure 63. Defect locations automatically detected based on time between acceleration peaks

Now that we have a good sense of where the defects are along this segment of road, we can evaluate how well the ResNet50 model performed at identifying the specific defects at each location. For this stretch of road, there were four types of defects present across all locations: cracks, patches, potholes, and manholes. The manual review and the FindFrame function were in agreement that these were the only defects that meet the acceleration threshold criteria for frame extraction. Figure 64 shows that how well the defects were identified across the defect locations. Each bar is a stack of the number of defects in the frames that were correctly identified by the ResNet50 model, the ones that were incorrectly identified, and a count of how many were missed. The correct and incorrect counts were calculated by manually reviewing the original extracted frames and the overlay applied by the ResNet50, similar to what is shown in Figure 58. This way, each defect can easily be counted as correct or incorrect. The missed defect count comes form the manual review of the video, by including the defect locations that the FindFrame function missed and counting the defects at each of those locations.

The bar representing the crack class in Figure 64 shows a strong result. The classifier did not incorrectly label any cracks, but 12 were missed due to a lack of response in the acceleration signal. Similar to Example 1, this was mostly due to the cracks not being driven over or being too small to create a strong acceleration response. This is also the case for the manholes and potholes that were missed by the FindFrame, but were identified during the manual review. The manholes incorrectly identified by the classifier were all abnormal light patches on the road surface that showed some resemblance of the metal of a manhole cover. Improving the classifier with more

training samples could reduce this type of error. Several of the instances of patches that were incorrectly identified were noticed on blurry extracted frames. In these blurry frames, there were instances which seemed to be smooth patches on road surface due to the motion blur. No instances of potholes were missed for this road segment. All of the incorrectly identified potholes were false positives, which were mostly cracks.

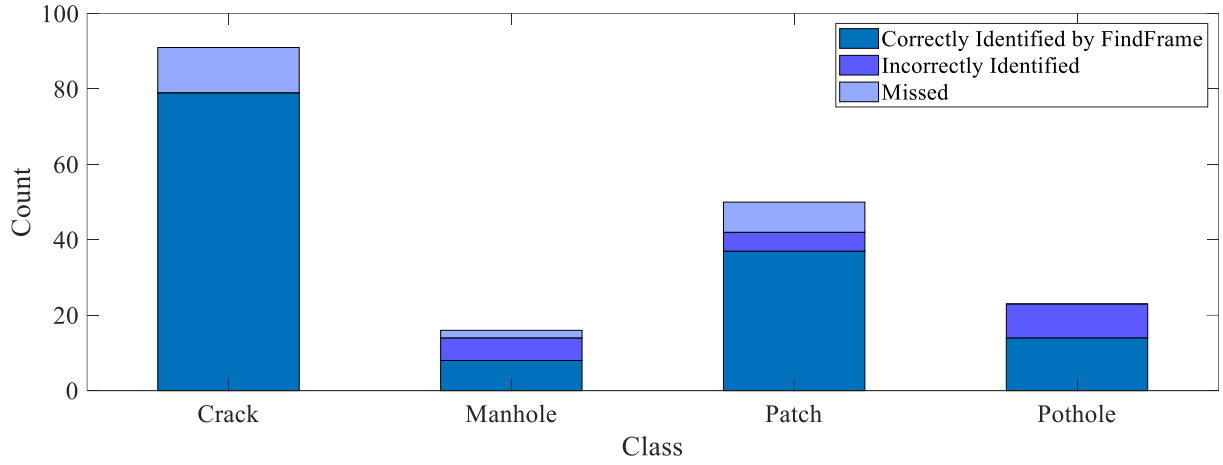

Figure 64. Defect identification in extracted frames compared to the manual review of the video

Overall, the system performed well in these two examples in terms of finding most defect locations (67% and 87% for examples 1 and 2, respectively) when compared to a manual count of the defect locations. The identification of defects at each location was also promising. Cracks, the most prominent class of all the defects, showed good performance in both examples, with no incorrect instances at the defect locations. The other classes all showed well over 50% identification across the defect locations. These results could be improved further with a better trained CNN classifier and by taking multiple passes (e.g., by different commuters) over the same stretch of road.

4.5 Challenges

This research faced many challenges. This section lists and discusses some of the reasons that the methods used in this research can fail.

1. Motion data ground truth
2. Indirect motion
3. Attribution of defects
4. Missed defects

5. Ambiguous defects

The first challenge identified has to do with the motion data and the lack of ground truth data for road roughness. Analysis of the vibration data that was collected by smartphone mounted to the test vehicle is nominally replicable with other vehicles, as was discussed in Section 4. But incorporating official roughness metrics, such as the IRI or PCI, would require data collection by a purpose-built vehicle. This could lead to the development of a calibration process so that more vehicles could be deployed for data collection, thus improving the road condition inventory and also improving the calibration process itself, since new data can be cycled in to better calibration against control data that has been collected by the surface profiler.

Next, consider the method of data collection. There is an inherent compromise made when using one device to collect the two different data types (motion and video) used for this research. Though the accelerometer that collects the motion data performs well enough, in terms of sampling frequency and accuracy, its placement on a windshield mount dampens the motion data, making the data's representation of the road surface less direct. This could be addressed in future work by mounting a separate accelerometer somewhere on the vehicle's wheel mount that interacts directly with the wheel.

Another problem related to the positioning of the smartphone is defect attribution. The problem is that there is no direct connection between the motion data triggering the video frame collection and the defects identified in the frame. Looking back to Figure 58, it depends on how the camera view lines up with the vehicle's wheels to be able to tell if the potholes or the cracks were what instigated the acceleration response sufficient to be considered an anomaly. This could be addressed by calibrating the camera view, so the band of pixels in front of the vehicle wheels are known, or by incorporating other motion sensor data, such as the gyroscope, to infer which wheels interacted with the road defect.

On the other hand, both the non-ideal data collector location and attributability problem are consequences of a benefit of the data collection system: it is easy to setup and it contains all sensing units in one device. The main reason these compromises were made was to address one of the largest challenges faced: missed defects. By activating the computer-vision module with peaks identified in the motion data, the vehicle must physically drive over the defect for there to be any chance that it is captured in the data. But as was mentioned, this problem is in part addressed by

the fact that data collection is fast and easy to set up. The missed data problem could in theory be overcome by the ability to have multiple users collecting data (called participatory sensing), though this is outside the scope of this research.

In addition to missing the defects, there is the challenge associated with misclassifying them. This can be seen in Figure 58. The ground truth pothole on the right side of the frame is identified as a crack in the segmented image. Looking at the original image, it could be argued that the visual feature is actually not a pothole. For these purposes, it can be agreed that the defect has been identified but the classification is ambiguous. Another factor affecting the classifier is motion blur. In especially rough areas, some of the video frames are so blurry that they are nearly unintelligible. This is not the case for most of the video frames though, so it is not likely to be a serious issue. It could be addressed by using a smartphone that has an either optically- or electronically-stabilized camera module.

Chapter 5: Conclusion

This final chapter discusses the findings and contributions of this research. It also summarizes the limitations of the research and what steps could be taken in the future in order to address these limitations.

5.1 Summary and Conclusions

A system was developed that can combine motion data and video footage to describe the road condition. The data can easily be collected with a low-cost smartphone and a mount for securing it to the vehicle's windshield. The data used in this research was collected while driving over 100 kilometers of road in the city of Thunder Bay. Both the motion and video data require minimal preprocessing.

We considered the analysis of the motion data by looking at how the vertical acceleration response captured by a smartphone correlated to overall road roughness and the ability to identify specific instances of anomalies on the road surface. For the overall road roughness, we presented the driving route as a geographic density plot, where the plot's density is influenced by the acceleration data. Anomalies are found in the acceleration time series based on a threshold value and minimum peak distance. The location of these anomalies in the time series was used later to find and extract frames from the video footage that match that moment.

The focus of the video analysis was on the extracted frames, because it can be done with far less computation than if the whole video was analyzed. The purpose of the computer vision-based analysis is to identify road defects in the extracted images. The first attempt at this used image thresholding. The approach showed limited success, so it was modified to use contrast enhancement and morphological operations to better segment defects in the images. Ultimately the thresholding approach was set aside in favour of a deep learning approach, using convolutional neural networks.

The CNNs were trained using a set of 658 images that were manually labelled for seven different classes of pixels found in the images. Classes for the defects included: crack, pothole, and patch. In total, nine CNNs were trained on the set of labelled training images. We used a test set of 50 images to evaluate the CNNs by inputting the test images and observing how the output label arrays compared to the ground truth label arrays. Based on these results and observations, we

modified the training set data, used it to train two more CNNs and evaluated them. They showed improved performance, especially at successfully identifying the types of defects in the image.

Last, several functions were created in MATLAB that allow the work previously discussed to be automatically integrated and packaged in a format that can be published on the web. The motion and vision data are synchronized to match their start and stop times. Then the key frames are extracted based on the spikes in the motion data. The frames are geotagged using the GPS data from the phone at that time, and then segmented by a CNN to create a list of defects in each frame. The frames, GPS data, and list of defects are all packaged into a KML file, which can be published online via the Google Maps Platform.

Several conclusions can be made from the completion of this work:

1. Good data replicability has been shown, an indication that this work could be scaled up if multiple vehicles were collecting data across an entire road network.
2. Thresholding methods are easier to understand and iterate when used to identify road defects in image, but they are limited and do not perform as well as CNNs.
3. The CNNs trained using our labelled images demonstrated a promising ability to identify defects in the video frames that were extracted based on the acceleration data.

The main contributions of this work include:

1. Creating a strong training set of labelled images for training CNN classifiers. The training set could be expanded on in the future and used to train new models or retrain existing ones.
2. Training several CNNs that demonstrated promising performance at segmenting images of a road surface into several classes, including the most common defects: cracks and potholes.
3. Creating an integrated process that combines video footage and motion data.

5.2 Limitations of the Research

A number of limitations were noticed in this research which are discussed in details in Section 4.4. They come from three main areas in the research.

1. Data Collection—There are trade offs associated with collecting multiple data types from a single device. In order for the data collector (the smartphone) to be well-positioned for

video capture, the quality of the motion data suffers in terms of its closeness to the road surface. Also, there is an inherent limitation of the data collection process whereby the vehicle must drive over a defect on the road in order for it to be identified; otherwise it will be missed.

2. Motion Data—Though the general trends of the motion data could be replicated with different vehicles, the acceleration data is an abstraction of the road surface. This is due to the distance and mechanical linkages between the phone and the road surface. The lack of ground truth for the motion data make it difficult to work toward any type of official road surface quality index rating, such as the IRI.
3. Vision Data—A minor, intermittent limitation is the presence of motion blur in the video footage. Usually, the motion blur only exists as a single frame in the video before it refocuses, but if the acceleration data lead to that frame being extracted, the CNN would perform poorly at segmenting, since all of the training images were collected from in-focus video footage. Moreover, a major limitation is the distribution of labels in the training images. Most of the surface of the road network is regular, with minimal defects. This created large imbalances among the different classes in the training dataset.

5.3 Recommendations for Future Work

There are several opportunities to improve and expand on the work done in this thesis. The most important one, in the researcher's view, is to build on the road roughness assessment by collecting road surface data from more direct sources than a smartphone mounted to a windshield. This could carry the benefit of assigning official roughness index values to roads across the municipality's entire road network.

Opening up a system to data contributions from other users could drastically increase the amount of data available for analyzing, but potentially at the cost of data quality and reliability. The variability in vehicles, smartphones, and driving behaviour could make comparisons difficult. On the other hand, though, an abundance of data could enable better decision making for the city engineers, who would have more opportunities to identify problem areas in the road network and prioritize maintenance resources accordingly.

The developed CNN models for this research were able to meet a main objective of this research: to identify road defects in an image of the road's surface. Still, we have some recommendations

for future improvement. One would be to collect more images that would improve the problem of class imbalance. When there are too few examples of a class, it leads to false negatives and poor classification. Another recommendation is to build the capability to attribute road defects identified in the extracted images to the anomalies in the acceleration signal.

References

Abulizi, N., Kawamura, A., Tomiyama, K., and Fujita, S. (2016). "Measuring and evaluating of road roughness conditions with a compact road profiler and ArcGIS." *Journal of Traffic and Transportation Engineering (English Edition)*, 3(5), 398–411.

Addis, R. (2002). *COST 334: effects of wide single tyres and dual tyres*. EU Directorate General Transport.

Ahmed, A., Bai, Q., Lavrenz, S., and Labi, S. (201). "Estimating the marginal cost of pavement damage by highway users on the basis of practical schedules for pavement maintenance, rehabilitation and reconstruction." *Structure and Infrastructure Engineering*, 11(8), 1069–1082.

Ahmed, M., HDaas, C. T., and Haas, R. (2011). "Toward low-cost 3D automatic pavement distress surveying: the close range photogrammetry approach." *Canadian Journal of Civil Engineering*, 38(12), 1301–1313.Alauddin, M., and Tighe, S. L. (2008). "Incorporation of surface texture, skid resistance and noise into pms." *Proc: 7th International Conference on Managing Pavement Assets. Calgary, Canada (June 24-28, 2008)*, Citeseer.

Amekudzi, A. A., and Attoh-Okine, N. O. (1996). "Institutional Issues in Implementation of Pavement Management Systems by Local Agencies." *Transportation Research Record*, 1524(1), 10–15.

ASCE. (2017). *ASCE's 2017 Infrastructure Report Card: Roads*.

Bandara, N., and Gunaratne, M. (2001). "Current and Future Pavement Maintenance Prioritization Based on Rapid Visual Condition Evaluation." *Journal of Transportation Engineering*, 127(2), 116–123.

Bektas, F., Smadi, O. G., and Al-Zoubi, M. (2014). "Pavement management performance modeling: Evaluating the existing PCI equations."

Berndt, D. J., and Clifford, J. (1994). "Using dynamic time warping to find patterns in time series." *KDD workshop*, Seattle, WA, 359–370.

Bhoraskar, R., Vankadhara, N., Raman, B., and Kulkarni, P. (2012). "Wolverine: Traffic and road condition estimation using smartphone sensors." *2012 fourth international conference on communication systems and networks (COMSNETS 2012)*, IEEE, 1–6.

Cabral, F. S., Pinto, M., Mouzinho, F. A., Fukai, H., and Tamura, S. (2018). "An Automatic Survey System for Paved and Unpaved Road Classification and Road Anomaly Detection using Smartphone Sensor." *2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, IEEE, 65–70.

Cackler, E. T., Ferragut, T., Harrington, D. S., Rasmussen, R. O., Wiegand, P., and others. (2006). "Evaluation of US and European concrete pavement noise reduction methods."

Campillo, J. R. M. (2018). "A Simplified Pavement Condition Assessment and Its Integration to a Pavement Management System." PhD Thesis, Arizona State University.

Canuma, P. (2018). "Image Pre-processing." *Medium*, <https://towardsdatascience.com/image-pre-processing-c1aec0be3edf> (Oct. 27, 2019).

Cao, M.-T., Tran, Q.-V., Nguyen, N.-M., and Chang, K.-T. (2020). "Survey on performance of deep learning models for detecting road damages using multiple dashcam image resources." *Advanced Engineering Informatics*, Elsevier, 46, 101182.

Chang, J.-R., Su, Y.-S., Huang, T.-C., Kang, S.-C., and Hsieh, S.-H. (2009). "Measurement of the international roughness index (IRI) using an autonomous robot (P3-AT)." *Proceedings of the 26th International Symposium on Automation and Robotics in Construction (ISARC 2009), Austin, TX, USA*, 24–27.

Cheng, H. D., and Miyojim, M. (1998). "Automatic pavement distress detection system." *Information Sciences*, 108(1), 219–240.

Chong, G., Phang, W., and Wrong, G. (2016). *Manual for Condition Rating of Flexible Pavements*. Ministry of Transportation Ontario.

De Zoysa, K., Keppitiyagama, C., Seneviratne, G. P., and Shihan, W. (2007). "A public transport system based sensor network for road surface condition monitoring." *Proceedings of the 2007 workshop on Networked systems for developing regions*, ACM, 9.

Douangphachanh, V., and Oneyama, H. (201 a). "A Study on the Use of Smartphones for Road Roughness Condition Estimation." *Journal of the Eastern Asia Society for Transportation Studies*, 10, 1551–1564.

Douangphachanh, V., and Oneyama, H. (201 b). "Estimation of road roughness condition from smartphones under realistic settings." *2013 13th International Conference on ITS Telecommunications (ITST)*, 433–439.

Du, Y., Liu, C., Wu, D., and Jiang, S. (2014). “Measurement of International Roughness Index by Using -Axis Accelerometers and GPS.” *Mathematical Problems in Engineering*.

Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., and Balakrishnan, H. (2008). “The pothole patrol: using a mobile sensor network for road surface monitoring.” *Proceedings of the 6th international conference on Mobile systems, applications, and services*, ACM, 29–39.

Feffer, S. (2017). “It’s all about the features - Machine Learning and Signal Processing tips.” *Reality AI*.

Forslöf, L., and Jones, H. (201). “Roadroid: Continuous road condition monitoring with smart phones.” *Journal of Civil Engineering and Architecture*, 9(4), 485–496.

Fwa, T. F. (Ed.). (2005). *The Handbook of Highway Engineering*. CRC Press, Boca Raton.

Gadelmawla, E. S., Koura, M. M., Maksoud, T. M. A., Elewa, I. M., and Soliman, H. H. (2002). “Roughness parameters.” *Journal of Materials Processing Technology*, 123(1), 133–145.

Gopalakrishnan, K., Khaitan, S. K., Choudhary, A., and Agrawal, A. (2017). “Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection.” *Construction and Building Materials*, 157, 322–330.

Gorges, C., Öztürk, K., and Liebich, R. (2019). “Impact detection using a machine learning approach and experimental road roughness classification.” *Mechanical Systems and Signal Processing*, 117, 738–756.

Hadjidemetriou, G. M., Christodoulou, S. E., and Vela, P. A. (2016). “Automated detection of pavement patches utilizing support vector machine classification.” *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, IEEE, 1–5.

Hadjidemetriou, G., Vela, P., and Christodoulou, S. (2018). “Automated Pavement Patch Detection and Quantification Using Support Vector Machines.” *Journal of Computing in Civil Engineering*, 32(1), 04017073.

Han, S. (2010). “Measuring displacement signal with an accelerometer.” *Journal of Mechanical Science and Technology*, 24(6), 1329–1335.

Hanson, T., Cameron, C., and Hildebrand, E. (2014). “Evaluation of low-cost consumer-level mobile phone technology for measuring international roughness index (IRI) values.” *Canadian Journal of Civil Engineering*, 41(9), 819–827.

Harikrishnan, P. M., and Gopi, V. P. (2017). “Vehicle Vibration Signal Processing for Road Surface Monitoring.” *IEEE Sensors Journal*, 17(16), 5192–5197.

Hartnett, K. (2019). “Computers and Humans ‘See’ Differently. Does It Matter?” *Quanta Magazine*, <https://www.quantamagazine.org/computers-and-humans-see-differently-does-it-matter-20190917/> (Oct. 3, 2019).

He, K., Zhang, X., Ren, S., and Sun, J. (2016). “Deep residual learning for image recognition.” *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hicks, R. G., Moulthrop, J. S., and Daleiden, J. (1999). “Selecting a preventive maintenance treatment for flexible pavements.” *Transportation research record*, 1680(1), 1–12.

Hjort, M., Haraldsson, M., and Jansen, J. (2008). *Road Wear from Heavy Vehicles*. NVF.

Hoang, N.-D. (2019a). “Image processing based automatic recognition of asphalt pavement patch using a metaheuristic optimized machine learning approach.” *Advanced Engineering Informatics*, 40, 110–120.

Hoang, N.-D. (2019b). “Automatic detection of asphalt pavement raveling using image texture based feature extraction and stochastic gradient descent logistic regression.” *Automation in Construction*, 105, 102843.

Hugo, D., Heyns, P. S., Thompson, R. J., and Visser, A. T. (2008). “Haul road defect identification using measured truck response.” *Journal of Terramechanics*, 45(3), 79–88.

Jang, J., Smyth, A. W., Yang, Y., and Cavalcanti, D. (201). “Road surface condition monitoring via multiple sensor-equipped vehicles.” 201 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 43–44.

Koch, C., and Brilakis, I. (2011). “Pothole detection in asphalt pavement images.” *Advanced Engineering Informatics*, Special Section: Engineering informatics in port operations and logistics, 25(3), 507–515.

Koch, C., Georgieva, K., Kasireddy, V., Akinci, B., and Fieguth, P. (201). “A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure.” *Advanced Engineering Informatics*, Infrastructure Computer Vision, 29(2), 196–210.

Koutsopoulos, N., and Downey, B. (199). “Primitive-Based Classification of Pavement Cracking Images.” *Journal of Transportation Engineering*, 119(3), 402–418.

Kulkarni, R. B., and Miller, R. W. (200). “Pavement management systems: Past, present, and future.” *Transportation Research Record*, 1853(1), 65–71.

Kyriakou, C., Christodoulou, S. E., and Dimitriou, L. (2019). “Smartphone-Based Pothole Detection Utilizing Artificial Neural Networks.” *Journal of Infrastructure Systems*, 25(3), 04019019.

Kyungwon, P., Natacha, T., and Lee, W. (2007). “Applicability of the International Roughness Index as a Predictor of Asphalt Pavement Condition1.” *Journal of Transportation Engineering*, 133(12), 706–709.

Lay, M. G. (2009). *Handbook of Road Technology*. CRC Press, London ; New York.

Liu, J., Yang, X., Lau, S., Wang, X., Luo, S., Lee, V. C.-S., and Ding, L. (2020). “Automated pavement crack detection and segmentation based on two-step convolutional neural network.” *Computer-Aided Civil and Infrastructure Engineering*, Wiley Online Library, 35(11), 1291–1305.

Loizos, A., and Plati, C. (2008). “An alternative approach to pavement roughness evaluation.” *International Journal of Pavement Engineering*, 9(1), 69–78.

Long, J., Shelhamer, E., and Darrell, T. (201). “Fully Convolutional Networks for Semantic Segmentation.” *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mednis, A., Strazdins, G., Zviedris, R., Kanonirs, G., and Selavo, L. (2011). “Real time pothole detection using Android smartphones with accelerometers.” *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, 1–6.

Mei, Q., and Gül, M. (2020). “A cost effective solution for pavement crack inspection using cameras and deep neural networks.” *Construction and Building Materials*, Elsevier, 256, 119397.

Mohan, P., Padmanabhan, V. N., and Ramjee, R. (2008a). “Nericell: rich monitoring of road and traffic conditions using mobile smartphones.” *Proceedings of the 6th ACM conference on Embedded network sensor systems*, ACM, 323–336.

Mohan, P., Padmanabhan, V. N., and Ramjee, R. (2008b). “Trafficsense: Rich monitoring of road and traffic conditions using mobile smartphones.” *Tech. Rep. no. MSR-TR-2008–59*.

Nguyen, T. S., Avila, M., and Begot, S. (2009). "Automatic detection and classification of defect on road pavement using anisotropy measure." *2009 17th European Signal Processing Conference*, IEEE, 617–621.

Nguyen, V. K., Renault, É., Milocco, R., and others. (2019). "Environment monitoring for anomaly detection system using smartphones." *Sensors*, 19(18), 3834.

Ngwangwa, H. M., Heyns, P. S., Labuschagne, F. J. J., and Kululanga, G. K. (2010). "Reconstruction of road defects and road roughness classification using vehicle responses with artificial neural networks simulation." *Journal of Terramechanics*, 47(2), 97–111.

Nienaber, S., Booysen, M. J., and Kroon, R. (201). "Detecting potholes using simple image processing techniques and real-world footage."

Ouma, Y. O., and Hahn, M. (2017). "Pothole detection on asphalt pavements from 2D-colour pothole images using fuzzy c-means clustering and morphological reconstruction." *Automation in Construction*, 83, 196–211.

Perttunen, M., Mazhelis, O., Cong, F., Kauppila, M., Leppänen, T., Kantola, J., Collin, J., Pirttikangas, S., Haverinen, J., Ristaniemi, T., and Riekki, J. (2011). "Distributed Road Surface Condition Monitoring Using Mobile Phones." *Ubiquitous Intelligence and Computing*, Lecture Notes in Computer Science, C.-H. Hsu, L. T. Yang, J. Ma, and C. Zhu, eds., Springer Berlin Heidelberg, 64–78.

Popov, A. A., and Geng, Z. (200). "Modelling of vibration damping in pneumatic tyres." *Vehicle System Dynamics*, 43(sup1), 145–155.

Radopoulou, S., and Brilakis, I. (2017). "Automated Detection of Multiple Pavement Defects." *Journal of Computing in Civil Engineering*, 31(2), 04016057.

Rogers, M., and Enright, B. (2016). *Highway Engineering*. John Wiley & Sons.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). "Mobilenetv2: Inverted residuals and linear bottlenecks." *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Sayers, M., Gillespie, T., and Queiroz, C. (1986). *The International Road Roughness Experiment (IRRE) : establishing correlation and a calibration standard for measurements*. The World Bank.

Sayers, M. W. (1984). "Guidelines for the conduct and calibration of road roughness measurements."

Sayers, M. W. (1998). “The little book of profiling: basic information about measuring and interpreting road profiles.”

Shamsabadi, S. S. (2019). “Constant Pavement Monitoring without Disrupting Traffic.” *Esri*.

Simonyan, K., and Zisserman, A. (2014). “Very deep convolutional networks for large-scale image recognition.” *arXiv preprint arXiv:1409.1556*.

Soloviev, A., Sobol, B., and Vasiliev, P. (2019). “Identification of Defects in Pavement Images Using Deep Convolutional Neural Networks.” *Advanced Materials*, Springer Proceedings in Physics, I. A. Parinov, S.-H. Chang, and Y.-H. Kim, eds., Springer International Publishing, Cham, 615–626.

Souza, V. M., Cherman, E. A., Rossi, R. G., and Souza, R. A. (2017). “Towards Automatic Evaluation of Asphalt Irregularity Using Smartphone’s Sensors.” *International Symposium on Intelligent Data Analysis*, Springer, 322–333.

Souza, V. M., Giusti, R., and Batista, A. J. (2018). “Asfault: A low-cost system to evaluate pavement conditions in real-time using smartphones and machine learning.” *Pervasive and Mobile Computing*, 51, 121–137.

Strubell, E., Ganesh, A., and McCallum, A. (2019). “Energy and Policy Considerations for Deep Learning in NLP.” *arXiv:1906.02243 [cs]*.

Tai, Y., Chan, C., and Hsu, J. Y. (2010). “Automatic road anomaly detection using smart mobile device.” *Conference on technologies and applications of artificial intelligence, Hsinchu, Taiwan*.

Terzi, S. (2007). “Modeling the pavement serviceability ratio of flexible highway pavements by artificial neural networks.” *Construction and Building Materials*, Fracture, Acoustic Emission and NDE in Concrete (KIFA-4), 21(3), 590–593.

Thompson, R., and Visser, A. (2000). “The functional design of surface mine haul roads.” *Journal of the Southern African Institute of Mining and Metallurgy*, 100(3), 169–180.

Treiber, M. A. (2010). *An Introduction to Object Recognition: Selected Algorithms for a Wide Variety of Applications*. Springer Science & Business Media.

Wahlström, J., Skog, I., and Händel, P. (2017). “Smartphone-based vehicle telematics: A ten-year anniversary.” *IEEE Transactions on Intelligent Transportation Systems*, 18(10), 2802–2825.

Walker, D., Entine, L., and Kummer, S. (2002). *Pavement surface evaluation and rating - Asphalt PASER manual*. Transportation Information Center, University of Wisconsin, Madison, WI.

Wambold, J., Defrain, L., Hegmon, R., Macghee, K., Reichert, J., and Spangler, E. (1981). "State of the art of measurement and analysis of road roughness." *Transportation research record*, 836, 21–29.

Wolters, A., Zimmerman, K., Schattler, K., and Rietgraf, A. (2011). *Implementing pavement management systems for local agencies*.

Yeganeh, S. F., Mahmoudzadeh, A., Azizpour, M. A., and Golroo, A. (2019). "Validation of smartphone based pavement roughness measures." *arXiv preprint arXiv:1902.10699*.

Yoder, E. J., and Witczak, M. W. (1991). *Principles of Pavement Design*. John Wiley & Sons.

Yousaf, M. H., Azhar, K., Murtaza, F., and Hussain, F. (2018). "Visual analysis of asphalt pavement for detection and localization of potholes." *Advanced Engineering Informatics*, 38, 527–537.

Zakeri, H., Nejad, F. M., and Fahimifar, A. (2017). "Image based techniques for crack detection, classification and quantification in asphalt pavement: a review." *Archives of Computational Methods in Engineering*, 24(4), 935–977.

Zandbergen, P. A. (2009). "Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning." *Transactions in GIS*, 13(s1), 5–25.

Zhang, D., Li, Q., Chen, Y., Cao, M., He, L., and Zhang, B. (2017). "An efficient and reliable coarse-to-fine approach for asphalt pavement crack detection." *Image and Vision Computing*, 57, 130–146.

Zou, Q., Cao, Y., Li, Q., Mao, Q., and Wang, S. (2012). "CrackTree: Automatic crack detection from pavement images." *Pattern Recognition Letters*, 33(3), 227–238.

Appendix A – MATLAB Code

Motion Data

Route Distances (Section Error! Reference source not found.)

```
function distances = RouteDistance(sensorData)
%RouteDistance calculates distance travelled from a set of lat-lon pairs

% Take lat and lon from data table and remove NaN values
lat = sensorData.lat;
lat = rmmissing(lat);

lon = sensorData.lon;
lon = rmmissing(lon);

% Define reference ellipsoid
wgs84InMeters = wgs84Ellipsoid;

% for n lat-lon pairs we will have n-1 distances
nPoints = length(lat);
nSpaces = nPoints - 1;

% calculate d, a vector of distances between the points in sequence
% d=zeros(nSpaces,1);
for i = 1:nSpaces
    d(i) =  distance(lat(i),lon(i),lat(i+1),lon(i+1),wgs84InMeters);
end

distances = d;

end
```

Image Data

Adaptive Threshold Function (Section 3.4.1.2)

```
function [enhancedImage,mask,maskedImage] = AdaptiveThreshold(image)
%AdaptiveThreshold -- Adaptively enhances the image contrast by
% accepting only the bottom 10% of intensity values and using a
% a high gamma correction factor. Then threshold the bottom 1%
% of pixel values to create a mask.

% check if image is rgb or gray
imageDimensions = size(image);
if length(imageDimensions) == 3
    % image is rgb, convert to grayscale
    grayImage = rgb2gray(image);
else
    grayImage = image;
end

% Set limits for input pixels.
% Remaining 90% will take on value of 1 (white).
low_percentile = prctile(grayImage,0,'all');
low_percentile = double(low_percentile)/255;
low_in = low_percentile/255;
high_percentile = prctile(grayImage,10,'all');
high_percentile = double(high_percentile);
high_in = high_percentile/255;

% Set limits for output pixels, full range of 8-bit values.
low_out = 0;
high_out= 1;

% Set Gamma correction factor
gamma = 2;

% Run image and parameters though imadjust function
% to produce contrast-enhanced image
enhancedImage = imadjust(grayImage,...
                         [low_in high_in],...
                         [low_out high_out],...
                         gamma);

% Threshold the enhance image
mask = enhancedImage < 0.8*255;
maskedImage = labeloverlay(image,mask);

end
```

Model Training (Section 3.4.2.3)

```
%%%% Setup new ground truth file and check parameters
load("...\gTruth.mat");
gTruth.DataSource
oldPathDataSource = "...\Training images";
newPathDataSource = "...\framesToLabel";

gTruth.LabelData.PixelLabelData
oldPathPixelLabel = "...\PixelLabelData";
newPathPixelLabel = "...\PixelLabelData";

alterPaths = {[oldPathDataSource newPathDataSource];
              [oldPathPixelLabel newPathPixelLabel]};

unresolvedPaths = changeFilePaths(gTruth,alterPaths)

imds = imageDatastore(”...\framesToLabel”);
pxds = pixelLabelDatastore(gTruth);
classes = pxds.ClassNames;

I = readimage(imds, 1);

imshow(I);

cmap1 = [
         120 120 50     % Class 1
         100 250 240    % Class 2
         192 150 122    % Class 3
         80  70  180    % Class 4
         30  200 180    % Class 5
         170 60  120    % Class 6
         190 120 180    % Class 7
        ];
cmap1 = cmap1 ./ 255;

C = readimage(pxds,1);

B = labeloverlay(I,C,'ColorMap',cmap1);
imshow(B);
pixelLabelColorbar(cmap1,classes);

%% training preparation

labelIDs = 1:numel(classes);
%%%[imdsTrain,         imdsTest,         pxdsTrain,         pxdsTest]         =
partitionCamVidData1(imds,pxds,labelIDs);

%function      [imdsTrain,      imdsTest,      pxdsTrain,      pxdsTest]      =
partitionCamVidData1(imds,pxds,labelIDs)
            % Partition CamVid data by randomly selecting 99% of the data for
training. The
            % rest is used for testing.
```

```
            % Set initial random state for example reproducibility.
            rng(0);
            numFiles = numel(imds.Files);
            % Returns a row vector containing a random permutation of the
integers from 1 to n inclusive.
            shuffledIndices = randperm(numFiles);

            % Use 99% of the images for training.
            N = round(0.99 * numFiles);
            trainingIdx = shuffledIndices(1:N);

            % Use the rest for testing.
            testIdx = shuffledIndices(N+1:end);

            % Create image datastores for training and test.
            trainingImages = imds.Files(trainingIdx);
            testImages = imds.Files(testIdx);
            imdsTrain = imageDatastore(trainingImages);
            imdsTest = imageDatastore(testImages);

            % Extract class and label IDs info
            classes = pxds.ClassNames;
%             labelIDs = 1:numel(pxds.ClassNames);

            % Create pixel label datastores for training and test.
            trainingLabels = pxds.Files(trainingIdx);
            testLabels = pxds.Files(testIdx);
            pxdsTrain    =    pixelLabelDatastore(trainingLabels,    classes,
labelIDs);
            pxdsTest = pixelLabelDatastore(testLabels, classes, labelIDs);
       % end

imageSize = [250 800 3];

% Specify the number of classes.
numClasses = numel(classes);

% Create Segnet Layers

lgraph = segnetLayers(imageSize, numClasses, 'vgg16');

tbl = countEachLabel(pxds);
imageFreq = tbl.PixelCount ./ tbl.ImagePixelCount;
classWeights = median(imageFreq) ./ imageFreq

pxLayer                                                                    =
pixelClassificationLayer('Name','labels','Classes',tbl.Name,'ClassWeights',cl
assWeights);
lgraph = replaceLayer(lgraph,"classification",pxLayer);

options = trainingOptions('sgdm', ...
    'Momentum', 0.9, ...
    'InitialLearnRate', 1e-2, ...
    'L2Regularization', 0.0005, ...
```

```
    'MaxEpochs', 100, ...
    'MiniBatchSize', 1, ...
    'Shuffle','every-epoch', ...
    'CheckpointPath', tempdir,...
    'Verbose', false,...
    'Plots','training-progress');

augmenter = imageDataAugmenter('RandXReflection',true,...
    'RandXTranslation',[-10 10], 'RandYTranslation' ,[-10 10]);

datasource = pixelLabelImageDatastore(imdsTrain, pxdsTrain, ...
    'DataAugmentation',augmenter);


%%%% start training

tic
[net, info] = trainNetwork(datasource, lgraph, options);
toc

disp('NN trained');

save('C:\Users\ehsan\Desktop\William\CNN_vgg16.mat',      'net',      'info',
'options');

%%%% end training

%%%% classification

analyzeNetwork(net)
```

Label Distribution Function (Section 0)

```
function [pixelDist,instanceDist] = LabelDistributions(pixelDatastore)
%LabelDistributions outputs two arrays, based on a given pixel label
%datastore. The first array is the count of pixels in each category across
%all images in the datastore. The second is the number of instances that
%the category is present in an image across the datastore. Outputs are per
%image. Summing them will give the distribution for the entire datastore.

nImages = length(pixelDatastore.Files);
nClasses= length(pixelDatastore.ClassNames);
classes = 1:nClasses;

%Pixel Counts
pixelDist=zeros(nImages,nClasses);
for i = 1:nImages
    labeledImage = readimage(pixelDatastore,i); %cat array of image i labels
    labeledImage = uint8(labeledImage);         %convert cat to uint8
    uniqueLabels = unique(labeledImage);
        nUniques = length(uniqueLabels);
    for j = 1:nUniques
        classLabel = uniqueLabels(j);
        pixelDist(i, classLabel) = sum(labeledImage(:) == classLabel);
    end
end

%Instance Counts
instanceDist = zeros(nImages,nClasses);
for i = 1:nImages
    labeledImage = readimage(pixelDatastore,i);
    labeledImage = uint8(labeledImage);
    uniqueLabels = unique(labeledImage);
    instanceDist(i,:) = sum(classes == uniqueLabels);
end

end
```

```
function results = ConvNetResults(imds,pxds,CNN,j)
%ConvNetResults -- Evaluate CNN on several metrics against ground truth
%   Before running ConvNetResults,
%   imds   = imageDataStore('.../framesToLabel');
%   pxds   = pixelLabelDatastore(gTruth);
%   CNN    ~ already exists from training process
%
%   ConvNetResults produces a structured array, results, that contains
%       labeled    -> categorical array(imgWidth,imgHeight,nImgs) from the
%       pixel label datastore--it is an array of all image ground truths
%       for comparing against the CNN's segmentation results.
%
%       segmented  -> categorical array(imgWidth,imgHeight,nImgs) from
%       running semanticseg() with the CNN that is being evaluated.
%
%       Jacc,Dice,BF
%       Precision, Recall
%       and more...
tic     % last run: 1593.854788 seconds -- 26.5 minutes

%Setup
net = CNN.net;
nImages = length(pxds.Files);
classes = pxds.ClassNames;
[imgHeight,imgWidth] = size(readimage(pxds,1));

%Preallocation
jacc = zeros(nImages,length(unique(classes)));
sdsc = jacc;
bfsc = jacc;
cm = zeros(length(classes),length(classes),nImages);

%get labeled and segmented categorical arrays, and other scores
for i = 1:nImages
    I = readimage(imds,i);
    labeled(:,:)   = readimage(pxds,i);
    segmented(:,:) = semanticseg(I,net);
    segSave(segmented,i,j);
    jacc(i,:) = jaccard(labeled(:,:),segmented(:,:))';
    sdsc(i,:) =    dice(labeled(:,:),segmented(:,:))';
    bfsc(i,:) = bfscore(labeled(:,:),segmented(:,:))';
    cm(:,:,i) = confusionMat(labeled(:,:),segmented(:,:),classes);
    I=0;
end

%precision & recall & others
[TPtoFN,accuracy,precision,recall,TNrate,balAcc,F1] = PrecisionRecall(cm);

%prepare to store in struct array
jacc = array2table(jacc,'VariableNames',classes);
sdsc = array2table(sdsc,'VariableNames',classes);
bfsc = array2table(bfsc,'VariableNames',classes);
accuracy = array2table(accuracy,'VariableNames',classes);
precision = array2table(precision,'VariableNames',classes);
recall = array2table(recall,'VariableNames',classes);
TNrate = array2table(TNrate,'VariableNames',classes);
balAcc = array2table(balAcc,'VariableNames',classes);
```

```
F1 = array2table(F1,'VariableNames',classes);

%cell array of tables w scores for each image
ImageScores = cell(nImages,1);
for i = 1:nImages
    Table(1,:) = jacc(i,:);
    Table(2,:) = sdsc(i,:);
    Table(3,:) = bfsc(i,:);
    Table(4,:) = accuracy(i,:);
    Table(5,:) = precision(i,:);
    Table(6,:) = recall(i,:);
    Table(7,:) = TNrate(i,:);
    Table(8,:) = balAcc(i,:);
    Table(9,:)  = F1(i,:);
    Table.Properties.RowNames = {'Jacc' 'Dice' 'BFscore' 'Accuracy' 'Precision'
'Recall' 'TNrate' 'balAcc' 'F1score'};
    ImageScores(i) = {Table};
end

%store all in a structure array
%results.Labeled = labeled; removed
%results.Segmented = segmented;
results.Jaccard = jacc;
results.Dice = sdsc;
results.BF = bfsc;
results.ConfMat = cm;
results.TPtoFN = TPtoFN;
results.Accuracy = accuracy;
results.Precision = precision;
results.Recall = recall;
results.TNrate = TNrate;
results.balAcc = balAcc;
results.ImageScores = ImageScores;

toc
end
```

Data Integration

Data Synchronization (Section 3.5.1)

```
function clippedData = DataClipper(sensorData,video)
%DataClipper -- Summary of this function goes here

% Bottom right position of video timestamp
timeCornerX = 246; %vals set for 1080p
timeCornerY =  52;

% Extract first frame's timestamp as gray image
firstFrame = read(video,1);
firstFrame = rgb2gray(firstFrame);
firstFrameTime = firstFrame(1:timeCornerY,1:timeCornerX);

% Extract last frame's timestamp as gray image
lastFrame = read(video,video.NumFrames(end));
lastFrame = rgb2gray(lastFrame);
lastFrameTime = lastFrame(1:timeCornerY,1:timeCornerX);

% Extract the text from each using Optical Character Recognition
characterSet = '0123456789:.';
opticalCharRecResults_Start = ocr(firstFrameTime,...
                                  "TextLayout","line",...
                                  "CharacterSet",characterSet);
opticalCharRecResults_End   = ocr(lastFrameTime,...
                                  "TextLayout","line",...
                                  "CharacterSet",characterSet);

% Convert result from character to datetime value type
videoStartTime = opticalCharRecResults_Start.Text;
videoStartTime = datetime(videoStartTime);

videoEndTime = opticalCharRecResults_End.Text;
videoEndTime = datetime(videoEndTime);

% yyyymmdd of videoStart/EndTime will be run date, not video date
% get true yyyymmdd from sensorLog data
times = sensorData.time;
y = times(1).Year;
m = times(1).Month;
d = times(1).Day;

videoStartTime.Year = y;  videoEndTime.Year = y;
videoStartTime.Month= m;  videoEndTime.Month= m;
videoStartTime.Day  = d;  videoEndTime.Day  = d;

% find start/end times and clip the data table
startDiffs = times - videoStartTime;
startPosition = find(startDiffs == min(abs(startDiffs)));

endDiffs = times - videoEndTime;
endPosition = find(endDiffs == min(abs(endDiffs)));
```

```
sensorVariableNames = sensorData.Properties.VariableNames;

clippedData = sensorData(startPosition:endPosition,sensorVariableNames);

end
```

Frame Extraction (Section 3.5.2)

```
function frameData = FindFrame(clippedData,video)
%FindFrame -- Identify frames in video that correspond to high acceleration
% instances in clippedDataBased on acceleration values in clippedData.
%   frameData : a table including the frame # in video that peak accels
%   occurred and also the lat lon values for where that frame was captured.

% Values & Constants
numVideoFrames = video.NumFrames;
data = clippedData;
data2FrameRatio = height(data)/numVideoFrames;

acc = data.accX;
peakHeight = -0.8;  %fix
peakDist = 1000;   %fix

speed = data.speed;
speedIndex = find(~isnan(speed));
speed = rmmissing(speed);

% Find acceleration peaks and their location within the series
[peaks,peakLocs] = findpeaks(acc,...
                        'MinPeakHeight',peakHeight,...
                        'MinPeakDistance',peakDist);

[lat,lon] = peakLocs2LatLon(peakLocs,clippedData);

% Match the peak locations to the video frame locations
videoFrameLocs = peakLocs./data2FrameRatio;
videoFrameLocs = round(videoFrameLocs);

% Find instantaneous speed during each peak reading
for i = 1:length(peakLocs)
    minIndexDiff = min(abs(speedIndex - peakLocs(i)));
    instSpeedLoc = find(abs(speedIndex-peakLocs(i)) == minIndexDiff);
    instSpeed = speed(instSpeedLoc);
    correction = round(-5.8*instSpeed + 77.3);
    videoFrameLocs(i) = videoFrameLocs(i) - correction;
end

frameLocations = videoFrameLocs;
frameData = table(frameLocations,lat,lon);

figure
plot(acc)
pbaspect([3,1,1]);
hold on
scatter(peakLocs,peaks)
hold off

end

function [frameLats,frameLons] = peakLocs2LatLon(peakLocs,clippedData)
%peakLocs2LatLon -- Uses the locations of the acceleration peaks (peakLocs)
```

```
% to get the latlon for each corresponding video frame. There is 1 GPS
% reading for every 100 acceleration reading, so the function will
% interpolate between the two closest latlon pairs.

lat = clippedData.lat;
idx = find(~isnan(lat));
lat = rmmissing(lat);
lon = clippedData.lon;
lon = rmmissing(lon);

frameLats = interp1(idx,lat,peakLocs);
frameLons = interp1(idx,lon,peakLocs);

end
```

Geotag Images (Section 3.5.3)

```
function GeoTag(frameData,video)
%GeoTag -- Extracts frames from video based on frameData, saves them, and
% adds lat/lon metadata using exiftool.

% vars
frames = frameData.frameLocations;
nFrames = length(frameData.frameLocations);
lat = frameData.lat;
lon = frameData.lon;
date=video.Name(1:8);

% save images
for i = 1:nFrames
    frame = read(video,frames(i));
    fileName = [date,sprintf('_%03d.jpg',i)];
    imwrite(frame,fileName);
end

% metadata consts
exiftool = 'exiftool ';
overwrt  = '-overwrite_original ';
gpsLatRef = '-exif:gpslatituderef=N ';
gpsLonRef = '-exif:gpslongituderef=W ';

% add the metadata
for i = 1:nFrames
    frame = read(video,frames(i));
    fileName = [date,sprintf('_%03d.jpg',i)];
    gpsLat = sprintf('-exif:gpslatitude=%.12f ',lat(i));
    gpsLon= sprintf('-exif:gpslongitude=%.12f ',lon(i));
    addLatData = [exiftool,overwrt,gpsLat,gpsLatRef,fileName];
    addLonData = [exiftool,overwrt,gpsLon,gpsLonRef,fileName];
    system(addLatData);
    system(addLonData);
end

end
```

Image Defect Identification (Section 3.5.4)

```
function defectList = SegSave(clippedData,frameData,video,CNN)
%SegAndTag -- With sensor data, a video, and CNN, extract the key frames
% from the video, segmenent them using the CNN and geotag them

% get dates for filename
data = clippedData;
dateInfo = data.time(1);
year=string(dateInfo.Year);
month=string(dateInfo.Month);
day=string(dateInfo.Day);
date=strcat(year,month,day);

nFrames = height(frameData);
frameLocations = frameData.frameLocations;
classes                                                                    =
{'Background','Road','Crack','Grate','Manhole','Patch','Pothole'}';
defects =   [3:7]; %exclude 1 and 2 spot of classes

CMap = [
           255 150 000   % Background -- Orange
           000 000 255   % Road       -- Blue
           255 000 000   % Crack      -- Red
           255 255 000   % Grate      -- Yellow
           000 255 255   % Manhole    -- Blue (light)
           000 255 000   % Patch      -- Green
           255 000 255   % Pothole    -- Purple
         ]./255;
trans = 0.5;

defectList=cell(nFrames,1);
for i = 1:3%nFrames
    frame = read(video,frameLocations(i));
    croppedFrame = frame(500:749,600:1399,:);
    labels = semanticseg(croppedFrame,CNN);
    overlay                                                                =
labeloverlay(croppedFrame,labels,'ColorMap',CMap,'IncludedLabels',defects,'Tr
ansparency',trans);
    frame(500:749,600:1399,:) = overlay;
    filename = strcat(date,sprintf('_%03d_Labelled.jpg',i));
    imwrite(frame,filename);
    defectList{i} = CreateDefectList(labels);
end

end

function defectsList = CreateDefectList(labels)
allDefects = {'Crack','Grate','Manhole','Patch','Pothole'};
uniqueLabels = unique(labels)
defects = removecats(uniqueLabels,["Road","Background"])
defects = uint8(defects)
defects(defects==0)=[]
defectsList = strjoin(allDefects(defects))
end
```

Make KML

```
function MakeKML(clippedData,frameData,defectList)
%MakeKML -- Produce .kml files of key defect locations and route taken.

% yyyymmdd_route-taken.kml
data = clippedData;
speed = data.speed;
zeroSpeeds = find(speed==0);
data(zeroSpeeds,:) = [];
routeLats = rmmissing(data.lat);
routeLons = rmmissing(data.lon);
dateInfo = data.time(1);
year=string(dateInfo.Year);
month=string(dateInfo.Month);
day=string(dateInfo.Day);
date=strcat(year,month,day);

folder    =    'C:\Users\William    Sprague\OneDrive    -    lakeheadu.ca\1-
Model\~FinalFolder\outputs\kml';
routeFile = strcat(date,'_route-taken.kml');

kmlwriteline(fullfile(folder,routeFile),routeLats,routeLons,...
                                        'Name','Driving Route',...
                                        'LineWidth',5,...
                                        'Color','b');

% yyyymmdd_defect-frames.kml
defectLats=frameData.lat;
defectLons=frameData.lon;
nPoints = length(defectLons);

imageURLs = {'https://raw.githubusercontent.com/user/...';...
             'https://raw.githubusercontent.com/user/...';...
             'https://raw.githubusercontent.com/ user/...';...
             'https://raw.githubusercontent.com/ user/...';...
             'https://raw.githubusercontent.com/ user/...'};

for i = 1:nPoints
    PointID{i} = sprintf('Defect Location %d',i);
    url{i} = sprintf('<img src="%s" height="300" width="auto" /> Identified
Defects:    %s    <br><br>          latitude:    %.10f    <br>longitude:
%.10f<br><br>',imageURLs{i},defectList{i},defectLats(i),defectLons(i));
end

pointsFile = strcat(date,'_defect-frames.kml');
kmlwritepoint(fullfile(folder,pointsFile),defectLats,defectLons,...
                                        'Name',PointID,...
                                        'Description',url);

end
```

Appendix B – Extended Data & Tables

Raw Outputs of SensorLog app, as seen in the Methodology.

label(N)	motionUserAccelerationX(G)
loggingTime(txt)	motionUserAccelerationY(G)
loggingSample(N)	motionUserAccelerationZ(G)
identifierForVendor(txt)	motionAttitudeReferenceFrame(txt)
deviceID(txt)	motionQuaternionX(R)
locationTimestamp_since1970(s)	motionQuaternionY(R)
locationLatitude(WGS84)	motionQuaternionZ(R)
locationLongitude(WGS84)	motionQuaternionW(R)
locationAltitude(m)	motionGravityX(G)
locationSpeed(m/s)	motionGravityY(G)
locationCourse(Â°)	motionGravityZ(G)
locationVerticalAccuracy(m)	motionMagneticFieldX(ÂµT)
locationHorizontalAccuracy(m)	motionMagneticFieldY(ÂµT)
locationFloor(Z)	motionMagneticFieldZ(ÂµT)
locationHeadingTimestamp_since1970(s)	motionMagneticFieldCalibrationAccuracy(Z)
locationHeadingX(ÂµT)	activityTimestamp_sinceReboot(s)
locationHeadingY(ÂµT)	activity(txt)
locationHeadingZ(ÂµT)	activityActivityConfidence(Z)
locationTrueHeading(Â°)	activityActivityStartDate(txt)
locationMagneticHeading(Â°)	pedometerStartDate(txt)
locationHeadingAccuracy(Â°)	pedometerNumberofSteps(N)
accelerometerTimestamp_sinceReboot(s)	pedometerAverageActivePace(s/m)
accelerometerAccelerationX(G)	pedometerCurrentPace(s/m)
accelerometerAccelerationY(G)	pedometerCurrentCadence(steps/s)
accelerometerAccelerationZ(G)	pedometerDistance(m)
gyroTimestamp_sinceReboot(s)	pedometerFloorAscended(N)
gyroRotationX(rad/s)	pedometerFloorDescended(N)
gyroRotationY(rad/s)	pedometerEndDate(txt)
gyroRotationZ(rad/s)	altimeterTimestamp_sinceReboot(s)
magnetometerTimestamp_sinceReboot(s)	altimeterReset(bool)
magnetometerX(ÂµT)	altimeterRelativeAltitude(m)
magnetometerY(ÂµT)	altimeterPressure(kPa)
magnetometerZ(ÂµT)	IP_en0(txt)
motionTimestamp_sinceReboot(s)	IP_pdp_ip0(txt)
motionYaw(rad)	deviceOrientation(Z)
motionRoll(rad)	batteryState(R)
motionPitch(rad)	batteryLevel(Z)
motionRotationRateX(rad/s)	avAudioRecorderPeakPower(dB)
motionRotationRateY(rad/s)	avAudioRecorderAveragePower(dB)
motionRotationRateZ(rad/s)	