TABLE OF CONTENTS

Page

INTRODUCTION 1

CHAPTER 1 BACKGROUND AND RELATED WORK 5
1.1 Pavement and Distress Types 5
1.1.1 Pavement with Asphalt Concrete Surface 6
1.1.2 Pavement with Jointed Portland Cement Concrete Surface 7
1.1.3 Pavement with Continuously Reinforced Concrete Surface 9
1.2 Manual Inspection 11
1.3 Automated Inspection 12
1.3.1 2D Image Processing 13
1.3.2 3D Based Reconstruction Method 16

CHAPTER 2 SELF-DRIVING CAR SENSORS AND HARDWARE CONFIGURATION 21
2.1 Self-Driving Car Sensors 21
2.1.1 LiDARs 21
2.1.2 RGB Camera 23
2.2 Motivation 23
2.3 The Hardware Setup 23
2.3.1 LiDAR Selection 24
2.3.2 LiDAR Scanning Visualization 25
2.3.3 2D Camera Selection 28
2.3.4 Data Processing 29
2.3.5 Hardware Configuration 30

CHAPTER 3 SELF-DRIVING CAR'S SENSORS: TESTS AND RESULTS 33
3.1 Experiments on the LiDAR 33
3.1.1 Indoor Experiments 34
3.1.1.1 Cloud to Cloud/Mesh Distance 34
3.1.1.2 Results Interpretation 43
3.1.1.3 Screened Poisson Surface Reconstruction 44
3.1.1.4 Results Interpretation 46
3.1.2 Outdoor Experiments 46
3.1.2.1 Depth Information from the Reconstructed Surface 48
3.1.2.2 Results Interpretation 53
3.1.3 Conclusion 53
3.2 RGB Camera 53
3.2.1 Object Detection and Instance Segmentation with RPN 54
3.2.1.1 Regions with Convolutional Neural Network (R-CNN) 54
3.2.1.2 Fast R-CNN 55
3.2.1.3 Faster R-CNN 56

3.2.1.4 Mask R-CNN 59
3.2.2 Mask R-CNN in Detecting Cracks: Experiments and Results..... 70
3.2.2.1 Training and Validating the Results on Mask R-CNN 73
3.2.3 Conclusion 86

CONCLUSION AND RECOMMENDATION..... 89

ANNEX I ACQUIRING THE LIDAR DATA 93

ANNEX II LIDAR CHANNEL ANGLES 95

ANNEX III THE READABLE FORMAT FILE 97

ANNEX IV MASK R-CNN CONFIGURATION..... 99

ANNEX V PREPARING THE DATASET, TRAINING AND EVALUATING MASK R_CNN..... 101

LIST OF BIBLIOGRAPHICAL REFERENCES..... 103

LIST OF TABLES

Page

Table 1.1 ACP distress types 7

Table 1.2 JCP distress types....................9

Table 1.3 CRCP distress types....................10

Table 2.1 LCMS and Ouster OS-1 specifications....................22

Table 2.2 Velodyne and Ouster LiDARs specifications and prices....................25

Table 3.1 Crack segmentation precision, recall, and AIU in different scenarios 83

LIST OF FIGURES

Page

Figure 1.1 JCP Cross section....8

Figure 2.1 Ouster-128 channels LiDAR mounted horizontally 26

Figure 2.2 Ouster-128 channels LiDAR directed downward....26

Figure 2.3 Ouster-64 channels LiDAR directed downward....27

Figure 2.4 Ouster-16 channels LiDAR directed downward....27

Figure 2.5 The LiDAR and the camera installed on the top of the car 31

Figure 3.1 Environment 2, surface with mixed reflectivity 35

Figure 3.2 Environment 1, high reflectivity point cloud....36

Figure 3.3 Environment 2, mixed reflectivity point cloud....36

Figure 3.4 Reference point cloud 37

Figure 3.5 Cloud to cloud distance....38

Figure 3.6 Cloud to Cloud distance. High reflectivity surface compared to its reference point cloud....39

Figure 3.7 Histogram of the absolute distance between the scanned point cloud (high-reflectivity surface) and its reference 39

Figure 3.8 Cloud to Cloud distance. Chess board compared to its reference point cloud....40

Figure 3.9 Histogram of the absolute distance between the scanned point cloud (mixed-reflectivity surface) and its point cloud reference....40

Figure 3.10 Cloud to mesh distance. High-reflectivity surface....41

Figure 3.11 Histogram of the absolute distance between the scanned point cloud (high-reflectivity surface) and its plane reference 42

Figure 3.12 Cloud to mesh distance. Mixed-reflectivity surface 42

Figure 3.13 Histogram of the signed distance between the scanned point cloud (mixed-reflectivity surface) and its reference 43

Figure 3.14 Different types of factors that affect the reconstruction of the surface......44

Figure 3.15 Reconstructed map of the high-reflective surface 45

Figure 3.16 Reconstructed map of the mixed-reflective surface..................................46

Figure 3.17 RGB frame with its equivalent ambient LiDAR data...............................47

Figure 3.18 RGB frame with its equivalent reflectivity LiDAR data 48

Figure 3.19 RGB frame and its LiDAR equivalent; pavement with no depth deformation 49

Figure 3.20 Pavement reconstruction with faulty depth deformation 49

Figure 3.21 RGB frame and its LiDAR equivalent; pavement with a low severity pothole...50

Figure 3.22 Pavement reconstruction with low severity pothole, the blue and red areas are faulty detections 51

Figure 3.23 RGB frame and its LiDAR equivalent; pavement with high severity pothole...52

Figure 3.24 Pavement reconstruction with high severity pothole 52

Figure 3.25 R-CNN model architecture 55

Figure 3.26 Fast R-CNN architecture..56

Figure 3.27 Faster R-CNN architecture 58

Figure 3.28 Prediction at different feature layers...59

Figure 3.29 Residual block...60

Figure 3.30 FPN diagram 61

Figure 3.31 RPN diagram showing the two classification and the bounding boxes branches 62

Figure 3.32 Intersection over Union...64

Figure 3.33 Box head feature extraction 66

Figure 3.34 2 x 2 ROI max pooling..67

Figure 3.35 Bilinear interpolation in RoIAlign 68

Figure 3.36 The classification head with the loss functions...69

Figure 3.37 Mask R-CNN block diagram 70

Figure 3.38 The conventional pipeline used for supervised machine learning algorithms 71

Figure 3.39 RGB image (left) and its corresponding ground truth (right). Different colours refer to different fractions of the crack, each colour in the GT image is considered as a separate object....72

Figure 3.40 Cracks in different lighting conditions, with random objects and different pavement materials....73

Figure 3.41 Architecture of Mask R-CNN....74

Figure 3.42 From left to right: RGB image with the GT; the top 40 mini proposals generated by ROI network; and the segmentation....75

Figure 3.43 From left to right: RGB image with the GT; crack segmentation before applying the merge function; and after applying the merge function....76

Figure 3.44 Top 150 proposals with their confidence scores....77

Figure 3.45 Final detection, before refinement (dotted lines), after refinement (solid lines)....77

Figure 3.46 The final mask detection....78

Figure 3.47 A selection of the C2 feature maps output....79

Figure 3.48 A selection of the C3 feature maps output....79

Figure 3.49 A selection of the C4 feature maps output....80

Figure 3.50 A selection of the C5 feature maps output....80

Figure 3.51 Alligator crack segmentation. Left: Ground truth segmentation. Middle: bounding box detection. Right: mask segmentation 82

Figure 3.52 Low noise crack segmentation. Left to right columns: RGB image; GT; mask segmentation....84

Figure 3.53 Moderate noise crack segmentation. Left to right columns: RGB image; GT; mask segmentation 85

Figure 3.54 High noise crack segmentation. Left to right columns: RGB image; GT; mask segmentation....86

LIST OF ABREVIATIONS

1D	1 Dimensions
2D	2 Dimensions
3D	3 Dimensions
LiDAR	Light Detection And Ranging
DIM	Distress Identification Manual
LTPP	Long-Term Pavement Performance
ACP	Asphalt Concrete-surface Pavement
JCP	Jointed Portland cement concrete-surfaced pavement
PCC	Portland Cement Concrete
CRCP	Continuously Reinforced Concrete Pavement
PCI	Pavement Condition Index
DNN	Deep Neural Network
CNN	Convolutional Neural Network
HNE	Holistically-Nested Edge
FPHBN	Feature Pyramid Hierarchical Boosting Network
RGB-D	Red Green Blue Depth
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
DMI	Distance Measuring Instrument
MLS	Mobile Laser Scanner
LCMS	Laser Crack Measurement System
IT	Information Technology

FoV	Field of View
ADAS	Advanced Driver-Assistance Systems
USD	United States Dollar
GPU	Graphic Processing Units
C2C	Cloud to Cloud
C2M	Cloud to Mesh
R-CNN	Region Convolutional Neural Network
mAP	Mean Average Precision
FC	Fully Connected
RoI	Region of Interest
RPN	Region Proposal Network
FPN	Feature Pyramid Network
COCO	Common Objects in Context
GT	Ground Truth
TP	True Positive
FP	False Positive
AIU	Average Intersection over Union
M	Million
BB	Bounding Box
NMS	Non Maximum Suppression
FCN	Fully Convolutional Network
CCE	Categorical Cross Entropy

BCE	Binary Cross Entropy
CE	Cross Entropy
FHWA	Federal Highway Administration
CFD	CrackForest Dataset
IOU	Intersection Over Union
HD	High Definition
SFF	Shape from Focus
SFDF	Shape from Defocus

LIST OF SYMBOLS

φ	LiDAR angle with an horizontal plane
d	LiDAR height to the ground
δ	Channels’ direction angle inside the LiDAR
θ	Laser beam firing angle
r	LiDAR range
x, y, z	3D coordinate

LIST OF MEASUREMENT UNITS

s	Second (time unit)
m	Meter (length unit)
mm	Millimeter (length unit)
cm	Centimeter (length unit)
Km/h	Kilometer per hour (speed unit)
Hz	Hertz (frequency unit)
^{o}	Degree (angular unit)

INTRODUCTION

Pavement distresses substantially impact the driving comfort, the vehicle operating costs and the road safety. The main causes of pavement distresses are: poor construction, fatigue due to heavy traffic, and natural factors such as water action and extreme temperature fluctuations. A reliable pavement distress detection tool would help to collect efficiently pavement distress data, to characterize road conditions and to identify the causes of pavement deterioration. The diagnostic of the causes of pavement deterioration is needed to select the appropriate intervention by solving the source of the problem rather than applying an inadequate treatment which will deteriorate rapidly.

Conventional assessment methods such as visual inspections, 2D computer vision methods, and 3D reconstruction methods, can detect pavement distresses and evaluate the pavement quality. Nevertheless, these methods are time-consuming and expensive, and may be inefficient especially in cities where road network is dense and subject to high volume traffic. Therefore, finding an effective tool to detect and classify pavement distress, such as deformation and cracks, is crucial to maintain roads in good condition, reduce user costs, and improve traffic safety.

Many inspection methods for pavement distress detection and classification are developed for commercial use. These methods fall in two categories: automated inspection and manual inspection. Automated methods rely on special equipment to detect pavement distresses. For instance, cameras are used to detect distresses such as cracks, potholes, and patches. However, cameras are not suitable to detect polished aggregate, raveling, and water bleeding. 3D sensors are more suitable to detect almost all distresses. Other sensors are also deployed to detect pavement distresses. For instance, accelerometer, microphone, sonar, and pressure sensors are used. However, these sensors are limited to specific types of distresses.

2D images taken by a camera and 3D images taken by 3D sensors, alongside the manual methods, are the common methods used in the detection of pavement distresses. Nevertheless, these three techniques face several limitations related to their efficiency, reliability, and price. Therefore, researchers are constantly trying to overcome those limitations by developing an efficient, reliable, and a low-cost pavement distress detection tool.

In this research, we address the aforementioned limitations by testing a low-cost solution based on 2D and 3D imagery to detect pavement distresses. We expect that this solution is more convenient by adopting onboard self-driving car sensors. And since self-driving cars are expected to be widely used in the future, exploiting their onboard technologies to assess road conditions may simplify the pavement data collection process, reduce its associate costs, allow to collect more data and therefore, improve the pavement inspection service. Furthermore, using the onboard self-driving cars' sensors for pavement distress detection eliminates the need for a dedicated external and expensive equipment to evaluate the pavement conditions.

Self-driving cars adopt multiple sensing technologies to map the surrounding environment, these technologies use a Light Detecting and Ranging, also known as LiDARs, radio detection and ranging, radars, cameras, and ultrasonic sensors in addition to other different sensors. This research focuses on using the two main vision sensing components of self-driving cars: 3D LiDAR technology and 2D camera.

Self-driving car's vision sensors are dedicated for obstacle detection and object recognition. To use these sensors for road condition assessment activity, their performance in detecting pavement distress has to be tested and validated. Thus, in this research, we address the following problematics:

- are the conventional self-driving car's 3D LiDAR and camera convenient for pavement distress detection?
- what are the limitations of the 3D LiDAR and of the camera in pavement assessment application and what are the methods to overcome the limitations?

- is the pavement distress detection tool, based on self-driving car's sensors, feasible to be deployed on current cars equipped with the adequate sensors?

To answer the above questions, we conduct several experiments inside the laboratory and in a real environment with a 3D LiDAR, similar to the one used in autonomous vehicles, and a 2D camera. We install the two sensors on a car and we drive on arterial roads in Montreal City. With the LiDAR, we collect the surrounding point cloud from which 3D points that belong to the pavement surface are extracted. With the camera, we capture the pavement surface. We test the extracted data to validate the capacity of both the 3D LiDAR and the camera in detecting pavement distresses.

The remainder of this report is organized as follows:

In chapter 1, we present the research background and the relevant literature review. First, we introduce the different types of pavements and associated types of surface deterioration that affect them. Next, we present the different methods to measure and assess the different types of pavement distress. Then, we present a critical review of the current methods deployed on the market and developed by researchers to detect pavement distress, highlighting advantages and limitations of each method.

In chapter 2, we present the motivation, the challenges, and the approach adopted to tackle the research problematics. First, we show the current self-driving car sensing technologies, then, we present the equipment used in this research for data collection.

In chapter 3, we show the experiments conducted on the LiDAR and the camera. We conduct experiments inside the laboratory and experiments in a real environment to test the capacity of the 3D LiDAR in detecting deformations on road surface. We address reliability, effectiveness, and price limitations of the state-of-the-art 2D image-based methods by introducing a complete approach for crack detection. We present the adopted machine learning algorithms and test it in detecting cracks. Finally, we discuss the results of the tests.

In the final section, we summarise and conclude our research, and propose recommendations for future works.

CHAPTER 1

BACKGROUND AND RELATED WORK

Designing durable road networks is a challenge for construction engineers. Different techniques, construction materials, and mixes are available to enhance the quality of the pavement and reduce its susceptibility to deteriorate under heavy traffic or climate conditions. However, pavements, regardless their composition, tend to deteriorate and require a periodic inspection and preventive maintenance. Therefore, several pavement inspection methods had been developed to characterize road condition. Chapter 1 addresses the different types of pavement surfaces and their specific distresses. It also reviews the current manual and automated pavement surface distress detection methods. Chapter 1 is divided into three sections. In Section 1.1, we show common types of pavement surfaces, and distresses that can affect them. In Sections 1.2 and 1.3, we review the measurement and assessment tool, and the current methods for pavement distress detection (i.e., manual and automated inspection).

1.1 Pavement and Distress Types

Construction engineers study different factors while designing or rehabilitating roads. The cost and the lifetime of the used materials, the maintenance cost, the traffic volume, and the climatic conditions are some of these factors. The Distress Identification Manual (DIM) for the Long-Term Pavement Performance Program (LTPP) (Miller & Bellinger, 2014), inspect pavement distresses on three different types of pavements: (1) pavements with asphalt concrete surfaces, (2) pavements with jointed Portland cement concrete surfaces, and (3) pavements with continuously reinforced concrete surfaces. In the following section, we briefly present the aforementioned types of pavements and what distresses are prone to develop on each, according to the LTPP Manual.

1.1.1 Pavement with Asphalt Concrete Surface

Asphalt Concrete-surface Pavement (ACP) is a common type of roads. Since the beginning of the twentieth century, road construction engineers use ACP (Polaczyk et al., 2019), which has relatively low-cost materials and it is easy to maintain compared to other pavements surfaces; however, it is considered as the less environmental friendly among the other common pavement surfaces. According to LTPP, ACP is susceptible to 15 different distress types that fall into five categories:

1. Cracking - Many possible reasons lead to different types of cracking, mainly fatigue in the asphalt surface under high traffic load, or harsh weather conditions;
2. Patching and potholes - Patching: a road area covered with new materials. Potholes: Depressions in the surface characterized by their depth and their area;
3. Surface deformation - Grooves on the wheel path caused by the high traffic load, overweight vehicles, or a failure in the pavement construction material;
4. Surface defects - Bleeding: It occurs when the asphalt binder expel through the aggregate due to high temperature. A bleeding surface may be characterized by its shiny, reflective surface, its fading texture and surface discolouring. Polished aggregate: the aggregate level above the asphalt surface starts to erode. Ravelling is characterized by the loss of surface aggregate;
5. Miscellaneous distresses - Lane to shoulder drop off: a drop off in the level of the road between the lane and the shoulder. Water bleeding: water leaks from joints or cracks.

Table 1.1 summarises all the different types. Each type has a unique description, severity levels, and unique measurement methods. For instance, fatigue cracking is labelled according to its severity level (i.e., low, moderate, and high) and measured by the surface of the affected area. Potholes, on the other hand, are measured by their surface and their depth.

Table 1.1 ACP distress types
Adapted from Miller et Bellinger (2014)

Distress category	**Distress type**	
A. Cracking	1. Fatigue cracking	
	2. Block cracking	
	3. Edge cracking	
	4. Longitudinal cracking	4.a. Wheel path longitudinal cracking
		4.b. Non-wheel path longitudinal cracking
	5. Reflection cracking at joints	
	6. Transverse cracking	
B. Patching and potholes	7. Patch/patch deterioration	
	8. Pothole	
C. Surface deformation	9. Rutting	
	10. Shoving	
D. Surface defects	11. Bleeding	
	12. Polished aggregate	
	13. Raveling	
E. Miscellaneous distress	14. Lane-to-shoulder dropoff	
	15. Water bleeding and pumping	

1.1.2 Pavement with Jointed Portland Cement Concrete Surface

Jointed Portland cement concrete-surfaced pavement, or JCP, is another popular choice for road construction. JCP is composed of Portland cement concrete (PCC) constructed on top of the subgrade or the base course. Figure 1.1 (Szymański et al., 2017) shows a cross section of a JCP. JCP is known for its strong characteristics, the low-cost maintenance, the capacity of load carrying, and its long life span. However, the initial cost of the JCP construction is high, and it is subject to crack and warp in harsh weather and high temperature fluctuations (Sautya, 2018).

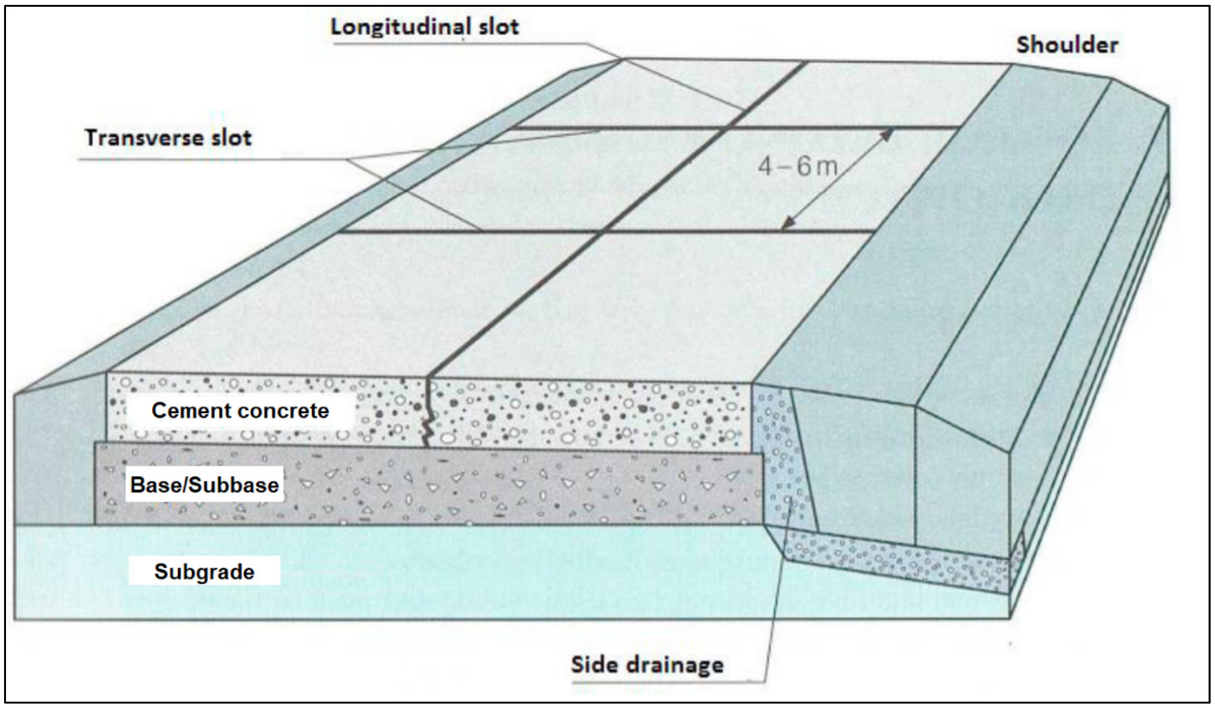


Figure 1.1 JCP Cross section
Adapted from Szymański et al. (2017)

LTPP manual groups the different types of JCP distress into four different categories: (1) cracking, (2) joint deficiencies (i.e. deficiencies between the concrete slabs), (3) surface defects, and (4) miscellaneous distresses. Table 1.2 summarizes the different types of JCP distresses.

Table 1.2 JCP distress types
Adapted from Miller et Bellinger (2014)

Distress category	**Distress type**	
A. Cracking	1. Corner break	
	2. Durability cracking	
	3. Longitudinal cracking	
	4. Transverse cracking	
B. Joint deficiencies	5. Joint seal damage	a. Transverse joint seal damage
		b. Longitudinal joint seal damage
	6. Spalling of longitudinal joints	
	7. Spalling of transverse joints	
C. Surface defects	8. Map cracking and scaling	a. Map cracking
		b. Scaling
	9. Polished aggregate	
	10. Popout	
D. Miscellaneous distress	11. Blowup	
	12. Faulting of transverse joints and cracks	
	13. Lane-to-shoulder dropoff	
	14. Lane-to-shoulder separation	
	15. Patch/patch deterioration	
	16. Water bleeding and pumping	

1.1.3 Pavement with Continuously Reinforced Concrete Surface

Continuously reinforced concrete pavement or CRCP is another type of pavements that has a long-term service life under harsh environmental conditions, temperature fluctuations, and heavy traffic (Tyson & Tayabji, 2012). CRCP has no joints, it's a single rigid construction reinforced with steel bars. A well-performing CRCP has a uniform transversal crack pattern. The uniform crack pattern reflects the consistency of the concrete mixture (Tyson & Tayabji, 2012). LTPP categorize CRCP distresses into three categories: (1) cracking, (2) surface defects, and (3) Miscellaneous distresses. Table 1.3 lists the different types of CRCP distress.

Table 1.3 CRCP distress types
Adapted from Miller et Bellinger (2014)

Distress category	**Distress type**	
A. Cracking	1. Durability cracking	
	2. Longitudinal cracking	
	3. Transverse cracking	
B. Surface defects	4. Map cracking and scaling	c. Map cracking
		d. Scaling
	5. Polished aggregate	
	6. Popout	
C. Miscellaneous distress	7. Blowup	
	8. Transverse construction joint deterioration	
	9. Lane-to-shoulder dropoff	
	10. Lane-to-shoulder separation	
	11. Patch/patch deterioration	
	12. Punchout	
	13. Spalling of longitudinal joints	
	14. Water bleeding and pumping	
	15. Longitudinal joint seal damage	

We should also mention that unpaved/dirt road is another type of pavement construction. It is mainly built in rural areas with harsh climate and temperature below freezing where the other pavement types fail or need a frequent maintenance.

1.2 Manual Inspection

Collecting pavement distress data is a crucial task to maintain roads and assess their condition. Different researchers and organizations develop distress catalogues that provide descriptions and methods for measuring pavement distresses and quantify their severity level. For instance, the Distress Identification Manual for the Long-Term Pavement Performance Program (LTPP Manual) developed by the US Department of Transportation, targeting United States and Canada road networks, offers a consistent method for a manual distress data collection (Miller & Bellinger, 2014; Ragnoli et al., 2018). ASTMD-6433 (D6433-18, 2018) also provides a standardization to identify and classify pavement distresses. It quantifies the pavement conditions with visual surveys using the Pavement Condition Index (PCI). In Europe, standards on distress identification and management are limited to the French Institute of Science and Technology for Transport, the Swiss Association of Road and Transport Professionals, and studies in Ireland to evaluate the surface of the pavement (Ragnoli et al., 2018).

Manual inspection methods consist of conducting a visual inspection or road surveys. By evaluating the road while driving, the inspectors report distress conditions according to their severity, density, and type. They also report the smoothness and ride comfort of the road. Visual inspection typically measures the Pavement Condition Index: an index of 100 reflects the best road condition and a zero index reflects the worst. This method is widely used due to its simplicity, but it does not return accurate information on the road distress (e.g., dimension and number of the encountered distress). It is also considered as a subjective evaluation method that depends on the inspector's experience and capability of accurately detecting all types of pavement distress.

Another manual method consists in conducting surveys. For instance, LTPP Manual introduces a survey method that consists in building a distress map showing the pavement distress location and severity level by walking or driving while searching for surface defects.

Manual methods are considered as a slow road inspection process, labour intensive, and may expose the workers and the driver safety to serious risks. They are also susceptible for human subjectivity and errors. And since obtaining the right distress information from the road is crucial for accurate surveys, service providers and researchers highly invest in developing the right technologies and in training the staff for the manual inspection (Ragnoli et al., 2018).

1.3 Automated Inspection

Due to the several limitations of manual inspection methods, researchers and service providers develop many systems to automate the process of collecting pavement distress data and to provide a non-subjective, more efficient, and accurate data. The automated inspection methods are based on quantifying each pavement distress. These methods often measure the width, the depth, the severity level, the road elevation, and classify distresses according to their types.

Automated methods offer more options in detecting road deteriorations. According to (Coenen & Golroo, 2017) and (Kim & Ryu, 2014), automated techniques can be classified into different categories: (1) Vision based method including 2D images and/or video processing, (2) 3D reconstruction method based on 3D sensor, (3) vibration-based method including accelerometer, microphone, and pressure sensors that reacts to the vibration force resulted from road deformation. These methods differ by the detection equipment and the type of road deterioration that they detect. For instance, the vision-based method and the 3D reconstruction methods have the highest amount of distress detection, hence, they are commonly used by pavement inspection service providers.

1.3.1 2D Image Processing

2D image processing methods are based on collecting road images with a moving or stationary camera. A moving data acquisition platform requires a high-speed and high-definition camera mounted on an inspection vehicle. This method allows inspectors to collect data while driving at a high speed. While this method provides high-quality images, it is expensive to be deployed on multiple cars. Moreover, collecting pavement data with a low number of equipped cars is inefficient especially while scanning a large road network. On the other hand, collecting still images is not an effective method since it is time-consuming, and it adds human interference at each distress location.

Following the data collection phase, 2D image processing algorithms are used to detect distresses. Oliveira et Correia (2014) develop a toolbox, CrackIT, to detect and classify cracks. The toolbox is based on four modules: (1) image preprocessing (e.g., image smoothing, white lane detection, etc.), (2) crack detection with pattern recognition techniques, (3) crack classification into types, and (4) evaluation routines to evaluate results (i.e., precision, recall, and Fm metrics). As reported in Oliveira et al. paper, CrackIT toolbox algorithms achieve a precision of 95.5% and a recall of 98.4%.

The CrackTree method is another crack detection method developed by Zou et al. (2012). The authors address the low contrast problem between cracks and the surrounding pixels (i.e., the shadow problem). The CrackTree method achieves a 79% precision and a 92% recall. However, CrackIT and CrackTree methods did not address the generalization problem: detecting cracks from images collected with moving vehicles and from images with different intensity, luminosity, blurring, and noise levels. Moreover, Koch et al. (2015) in their review on computer vision-based detection methods, list the limitations of such methods. The authors mention that the generalization is still a problem for the classic computer vision-based methods.

Akagic et al. (2018) work on an unsupervised method for detecting pavement crack using Otsu thresholding. The authors in their work split the developed method into two phases. In the first phase, the authors split their image into four small images. They calculate the mean and standard deviation for each small image. In the second phase, they calculate the image histogram and the Otsu thresholding. Otsu thresholding is a method to find an adaptive threshold that binarize the image into two classes: a background and a foreground. This is done by finding the values of the within-class variance and the between-class variance of an image. According to Akagic et al. (2018) the Otsu thresholding method achieves 77.77% recall and 77.27% precision on their own 50 images dataset. The authors reported that unwanted noise in the background can result after applying the Otsu thresholding. Moreover, we noticed that the segmented cracks miss a significant number of pixels that should be considered as a crack object. This is due to the difference in the luminosity level and the colour contrasts in a crack.

Classic computer vision methods highly depend on illumination and weather conditions, even shadows may affect the results of pavement distress extraction, these methods may return faulty detection results and/or fail to detect others. In addition, the lack of exact geospatial tagging for each image and the image distortion may also affect the accuracy and the efficiency of this method (Guan et al., 2016).

More recent works in computer vision-based methods focus on crack detection using supervised machine learning techniques. L. Zhang et al. (2016) apply Deep Neural Network model, DNN, to detect cracks. They train their model with 500 still images collected with a smartphone. They achieve a precision of 87% and a recall of 92.5% on their own-collected dataset. Furthermore, the authors reported that Convolutional Neural Network (CNN) provides superior crack detection performance compared to handcraft feature extraction methods. The results in (L. Zhang et al., 2016) are reported on crack patches with 99 × 99 pixels in size.

Gopalakrishnan et al. (2017) use DNN with transfer learning to detect cracks in 212 images from the Federal Highway Administration (FHWA) and LTPP database. The authors use

VGG-16 (Simonyan & Zisserman, 2014), a deep convolutional neural network pre-trained on the massive ImageNet database (Deng et al., 2009). They use the first 15 layers of this network as a feature generator network to extract features from 760 images. Then, they use the extracted features to train another classifier to categorize crack and non-crack images. The highest accuracy reported in this paper is about 90% for VGG-16 followed by a neural network classifier. Gopalakrishnan et al. experiment with crack/non-crack classification but they did not address the detection nor the segmentation challenge. Furthermore, the authors in (Alfarrarjeh et al., 2018) use region-based deep learning approaches for road damage detection. Alfarrarjeh et al. (2018) report a 62% F1 score (F1 score corresponding to a weighted average of precision and recall) while detecting bounding boxes around the damaged road area. Gopalakrishnan et al. (2017) and Alfarrarjeh et al. (2018) did not report scores on crack segmentation.

Attard et al. (2019) use Mask Regional Convolutional Neural Network or Mask R-CNN, which is a DNN for object detection (He et al., 2017), to detect cracks on concrete wall surfaces. They report a 93.94% precision and a 77.5% recall. They prove that Mask R-CNN can be used for detecting cracks. However, their research is based on still images for concrete walls and not for concrete or asphalt pavement surface. Mask R-CNN is a multi-stage deep neural network for object detection and segmentation. It uses ResNet (Kaiming He et al., 2016) to generate feature maps. It also uses the feature pyramid network architecture (Lin et al., 2017) to preserve the semantic values in the feature map from the deep convolutional layer and the resolution from the first convolutional layers. For the pixel wise detection or the segmentation task, Mask R-CNN uses a fully convolutional networks (Long et al., 2015). More details on Mask R-CNN architecture is presented in Chapter 3.

A recent work by Yang et al. (2019), report a maximum F1 score equal to 60.4% on the CRACK500 dataset (L. Zhang et al., 2016) and 68.3% on the CrackForest Dataset, CFD dataset, (Shi et al., 2016). Their method is based on feature pyramid method (Lin et al., 2017). The authors in (Yang et al., 2019) adopt the Holistically-Nested Edge (HNE) method (Xie &

Tu, 2015). According to Yang et al. (2019), crack detection task is very similar to the edge detection task since they share common features. The Feature Pyramid Hierarchical Boosting Network (FPHBN), (Yang et al., 2019), achieves the best results on different datasets (whether for CRACK500, Cracktree200, CFD, GAPs384 (Eisenbach et al., 2017), Aigle-RN (Amhaz et al., 2016)) compared to the other state-of-the-art methods.

1.3.2 3D Based Reconstruction Method

The reconstruction method based on 3D sensor can be divided into different categories: (1) visualization using Microsoft Kinect sensor, (2) stereo vision, (3) 3D laser scanner, (4) shape from focus and defocus (SFF, SFDF), and (4) photometric stereo method. The first method is based on Microsoft Kinect sensor which is a low-cost infrared, Red Green Blue – Depth, (RGB-D), camera. Microsoft Kinect is mainly used in gaming and robotics application. Joubert et al. (2011) and Moazzam et al. (2013) respectively, investigate the detection of potholes using a Kinect sensor. Y. Zhang et al. (2018) also investigate the Kinect sensor on a fixed platform; the acquired results from the three references show its potential in pavement inspection. However, Kinect sensor visualization is susceptible to infrared saturation when it is directly exposed to the sun light (Abbas & Muhammad, 2012). Moreover, this method needs further research and development in terms of detecting different types of pavement distress, integrating the Kinect sensor with different sensors (e.g., Global Navigation Satellite Systems (GNSS), Inertial Measurement Unit (IMU), and Distance-Measuring Instruments (DMI)), deploying the entire system on a moving platform, and performance validation in a real environment.

Stereo vision method captures 3D images by using two 2D cameras and epipolar geometry. To build a 3D surface, a spatial relationship based on epipolar geometry is established between points from the two 2D images. This technique faces several limitations such as motion blur, object occlusion, and needs sophisticated equipment. According to Mathavan et al. (2015), this method is only suitable to detect potholes and rutting. Compared to 3D laser scanners, stereo vision method has limited types of pavement distress detection.

Shape from focus (SFF) method consists of computing the depth by measuring the circle of confusion of a point in the captured image. When an object in the captured image is not in the range of the camera focus it will appear blurred. Once the object in-focus, the distance of the object from the camera is computed. Algorithms such as Tenengrad (W. Huang & Jing, 2007), (Sun et al., 2005) and the modified Laplacian (Nayar & Nakagawa, 1994) are used to compute the distance of an object in an image. This method is hard to implement on a moving vehicle due to the need of taking multiple images from different optical axis positions from a stationary device (Mathavan et al., 2015). This method is only suitable to capture macro or micro texture from a standstill device (Mathavan et al., 2015)

Shape from defocus (SFDF) method uses special algorithm, e.g. zero-mean Gaussian depth-defocus (Kuhl et al., 2006), to compute the distance from a camera by measuring the amount of blurring or defocusing in an image, in contrast to SFF method, SFDF requires fewer images (Subbarao & Surya, 1994).

Photometric stereo is another method that consists of taking multiple image of the same object from a stationary camera while exposing the captured scene to different light sources. This method can only be used with special equipment and from a stationary devise.

3D laser methods can be extended to detect different types of pavement distress with high precision and accuracy. 3D laser scanner has undergone intense study since it is capable to acquire an enormous accurate number of data at a high rate; the collected data are called point cloud. The larger the number of acquired points is, the denser the point cloud is. Each point belonging to the point cloud mainly has a 3D coordinate and a geotagging reference. Three types of navigation sensors involve in referencing these points: (1) a Global Navigation Satellite System (GNSS), (2) an Inertial Measurement Unit (IMU), and (3) Distance

Measurement Instrument (DMI). All these components besides a 3D laser scanner(s), camera(s), and a control unit are mounted on a moving platform (mainly on a moving vehicle in road inventory) form a Mobile Laser Scanner (MLS) or land-based, which is also known as mobile LiDAR (Light Detection And Ranging) (Guan et al., 2016). According to Coenen et Golroo (2017) and Mathavan et al. (2015), MLS in pavement inspection is usually used to detect cracks, potholes, rutting, patching, bleeding, macro texture, shoving, raveling, joint faulting, and spalling.

Another method based on laser scanner technology is used in pavement inspection. This method is called a high-resolution 3D line laser scanner. This technology is deployed and commercialized by several companies such as “ROMDAS,” “Arrb group,” “Mandli communications,” etc. These companies collect high-resolution data for road profile by using a Laser Crack Measurement System (LCMS) developed by “Pavemetrics” (ROMDAS, 2016). Laurent et al. (2018) address this technology in their article. According to Laurent et al. two laser profilers are used to cover up to 4-meter road lane profile. 4,000 3D points are acquired for each profile to form a 3D transverse profile. The profile rates of line laser scanner sensors can attend a frequency up to 28,000 Hz allowing a longitudinal resolution of 1 mm at vehicle speed of 100 km/h (Laurent et al., 2018). This high resolution and high accuracy system detects different pavement distresses, and even analyses surface texture.

Many different techniques are used in calculating the depth of pavement distress by the 3D laser scanner method. The triangulation method is mainly used in line laser scanner; phase-shift and time of flight are mainly used in LiDAR technology. The triangulation technique is the fastest and most accurate and precise technique, followed by the phase shift, and finally the time of flight. For this reason, the line laser scanner has been adopted by many companies to detect pavement profiles at a high resolution and a high speed. However, the LiDAR technology is mainly employed in road inventory, to build a complete 3D model of roads and its elements (e.g., vehicles, pedestrian, pavement, joints, trees, building, road, signs, etc.). These methods of acquiring road information are not cost effective.

In this project, we explore 2D image processing, and 3D reconstruction method to detect and classify pavement distress using a conventional platform, which is self-driving cars. Self-driving cars have similar techniques for sensing the surroundings. They often use 3D LiDARs, and cameras. However, the technology commonly used to detect pavement distress is much more accurate and is dedicated for building high resolution and highly accurate maps of the pavement as opposed to the self-driving car equipment designed for obstacle detection, which does not take in consideration the fine details of the pavement. We collect data from the same sensors deployed on self-driving cars to extract pavement data. We test the capacity of self-driving car's LiDAR in detecting pavement distress. We extract RGB images and we test image processing methods based on machine learning models to detect, classify and segment pavement distresses from the RGB images. For validation purpose we assume that the acquired data using our platform is similar to the data acquired by conventional self-driving car's sensors.

CHAPTER 2

SELF-DRIVING CAR SENSORS AND HARDWARE CONFIGURATION

Self-driving cars are expected to prevail in the intelligent transportation markets. Leveraging their onboard technology for pavement distress detection may contribute to the advancement of automated pavement distress assessment techniques. In the last decade, several Information Technology (IT) companies started to develop their own self-driving cars. For instance, Uber and Waymo (Google formerly), as well as car manufacturers are developing self-driving vehicles with 3D LiDARs and cameras; Tesla Incorporation uses radar, cameras, and ultrasonic sensors in their autopilot system; and comma-ai uses an RGB camera, and ultrasonic sensors (Santana & Hotz, 2016). As such, the current market of self-driving car is mainly based on LiDARs, radars, ultrasonic sensors, and cameras. In the following, the self-driving car's 2D LiDAR, 3D LiDARs, and RGB cameras are discussed as a potential solution for pavement distress detection.

2.1 Self-Driving Car Sensors

2.1.1 LiDARs

Light detection and ranging sensor, also known as LiDAR, is a sensor that measures the distance between the sensor and the target. A time of flight LiDAR measures the distance by calculating the time between emitting and receiving the reflected laser beam. We can distinguish three types of LiDARs: 1D, 2D, and 3D LiDARs. A 1D LiDAR uses one laser beam fixed at one axis. A 2D LiDAR uses one laser beam spinning in one plane, the measured points are defined by their x and y coordinates. A 3D LiDAR uses an array of laser beams fixed (e.g., a solid-state LiDAR or a flash LiDAR) or spinning (e.g., mechanical LiDAR). A 3D LiDAR measures the distance and generates a point cloud in a 3D space.

3D LiDARs deployed on vehicles can be separated into two categories: spinning LiDARs with a rotational head, and solid-state LiDARs with no moving parts. The spinning LiDARs are mainly used in self-driving cars for short or long-range scanning. Solid state LiDARs are for short distance scanning with a limited field of view (FoV) and less resolution than the spinning LiDARs. Solid state LiDARs are used in Advanced Driver-Assistance Systems (ADAS) for near obstacle detection.

The spinning LiDARs adopted in self-driving cars technology are developed for obstacle avoidance (i.e., detecting pedestrians, traffic signs, etc.) and depth assessment. Therefore, their depth accuracy can be limited to few centimeters instead of sub millimeters, in contrast to the accurate 3D laser scanners designed for pavement distress detection. Table 2.1 shows a comparison between a 3D laser scanner used for crack detection: Pavemetrics sensors for LCMS (Laurent et al., 2018); and a 3D LiDAR deployed on self-driving car: Ouster OS-1 16 channels.

Table 2.1 LCMS and Ouster OS-1 specifications

Specifications	**LCMS (Laurent et al., 2012)**	**3D LiDAR (Ouster OS-1 16ch) (Ouster, 2018)**
Sampling rate	5,600 to 11,200 profiles/s	160 - 320 profiles/s
Vehicle speed	100 km/h	70 - 100 km/h
Profile spacing	1 to 5 mm	7 - 15 cm
3D points per profile	4096 points	200 to 400 (4 m FoV)
Transverse FoV	4 m	Up to 100 m
Z-axis (depth) accuracy	0.5 mm	3 cm
X-axis (transverse) resolution	1 mm	1.1 to 2.4 cm

2.1.2 RGB Camera

RGB cameras are commonly installed on self-driving cars. They capture high resolution images and recreate a 2D visual representation of the surrounding. Cameras are placed around the vehicle so they can capture a 360^{o} image. However, cameras have some limitations: they lack depth information, thus, the distance between the vehicles and the captured obstacle is unknown. Therefore, the data returned from an RGB camera is usually fused with data from other depth sensors such as LiDAR, radar, and ultrasonic sensors.

2.2 Motivation

The previous work discussed in Chapter 1 shows several methods to detect different types of pavement distress. These methods embolden further research to overcome some of their limitations. For instance, a high quality laser scanner such as LCMS is not a cost effective solution to be used frequently to detect pavement distresses, and conventional LiDARs are less accurate. 2D image-based techniques show a good solution to detect cracks from still images; however, faster methods are needed to collect data such as sequential images from a conventional video camera mounted on a moving vehicle. Moreover, the recent work developed with deep convolutional neural network, the availability of pre-trained deep models alongside with deep learning tools such as *PyTorch, Keras, TensorFlow*, and the accessibility to Graphic Processing Units (GPUs), make this venue popular upon many other classification methods to solve problems in many different domains (e.g., medical imagery, transportation technologies, etc.).

2.3 The Hardware Setup

In this project, we test two self-driving car sensors: a 3D spinning LiDAR, and an RGB camera. The LiDAR is tested to build a 3D road image and to extract road profiles, whereas the camera is used to capture pavement images.

2.3.1 LiDAR Selection

A mechanical self-driving car's LiDAR is characterized by the number of channels or the number of lasers, e.g., a 16 channel LiDAR has an array of 16 firing laser beams. It is also characterized by the field of view, FoV, which is the coverage area that the LiDAR exposes. The horizontal resolution is the angle between two consecutive firing, and the vertical resolution is the angle between two adjacent laser beams in the same row. And finally, the rotation rate is the laser beams spinning speed; a higher spinning speed may reduce the horizontal resolution.

We investigate two mechanical LiDARs' manufacturers for self-driving cars: Velodyne and Ouster. Velodyne is one of the popular LiDARs provider for self-driving car companies. In 2016, this Silicon Valley-based LiDAR company, worked with 25 self-driving car programs (Cava, 2016). Ouster Inc. is another LiDARs provider company founded in 2015, in San-Francisco. These two similar companies design LiDARs for autonomous vehicles, robotics, drones, mapping, mining, and defense application. However, they are not involved in pavement assessment / distress detection applications. Table 2.2 shows these two companies LiDARs' specifications.

Table 2.2 Velodyne and Ouster LiDARs specifications and prices

LiDAR type	Number of channels	Range accuracy	FoV (Vertical)	Angular resolution (Vertical)	Angular resolution (Horizontal)	Rotation rate (Hz)	Price (USD)
Velodyne Puck	16	± 3 cm	30°	2.0°	0.1° – 0.4°	5 – 20	4,999
Velodyne Puck Hi-Res	16	± 3 cm	20°	1.33°	0.1° – 0.4°	5 – 20	8,500
Velodyne HDL-32E	32	< 2 cm	41.34°	1.33°	Not Available	5 – 20	29,900
Velodyne HDL-64E	64	± 2 cm	26.9°	0.4°	0.08° – 0.35°	5 – 20	85,000
Ouster OS-1 16	16	± 3 cm	33.2°	2.0°	0.17° – 0.7°	10 – 20	3,500
Ouster OS-1 64	64	± 3 cm	33.2°	0.5°	0.17° – 0.7°	10 – 20	12,000 8,000[1]
Ouster OS-1 128	128	± 3 cm	45°	0.35°	0.17° – 0.7°	10 – 20	18,000 12,000[1]

2.3.2 LiDAR Scanning Visualization

To visualize different LiDAR configurations we made the following drawings. The drawings show different Ouster LiDARs configurations that are mounted two meters on top of the ground. Figure 2.1 corresponds to Ouster-128ch mounted horizontally. The minimum distance between two consecutive profiles is 98 mm. Figure 2.2 shows the same LiDAR directed downward (30° angle), the distance between two consecutive profiles ranges between 12.6 mm and 35mm, the vertical FoV covered by this LiDAR is equal to 2.4 m. Figure 2.3 shows *Ouster-64ch* directed as the same previous drawing. The minimum distance of two consecutive

[1] non-profit research

profiles is 19.14 mm, and the largest one is 37.68 mm; the vertical FoV covered by this LiDAR is equal to 1.6 m. Figure 2.4 shows Ouster-16ch with the same configuration as in figures 2.2 and 2.3. The distance between two consecutive beams ranges between 82.3 mm and 156 mm, and the distance covered by the vertical FoV is 1.6 m. Thus, the higher the number of channels, the better the resolution.

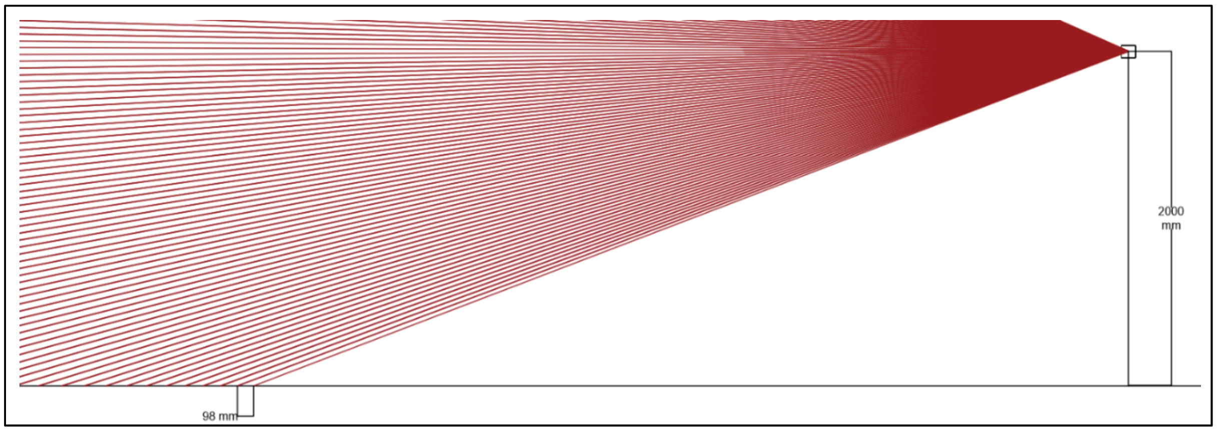


Figure 2.1 Ouster-128 channels LiDAR mounted horizontally

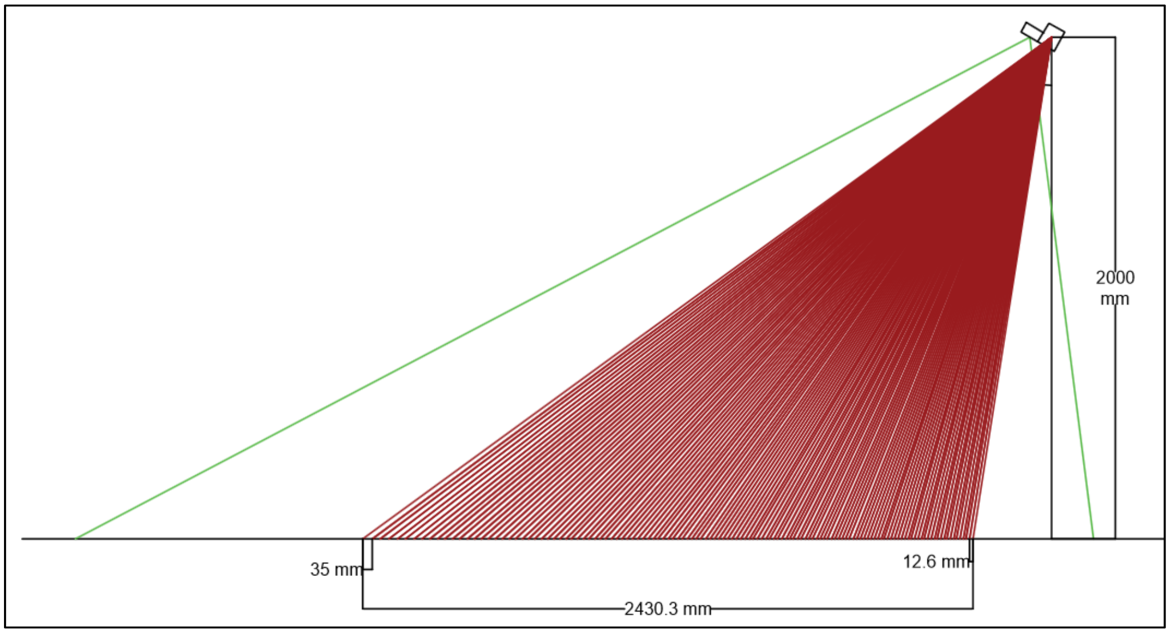


Figure 2.2 Ouster-128 channels LiDAR directed downward

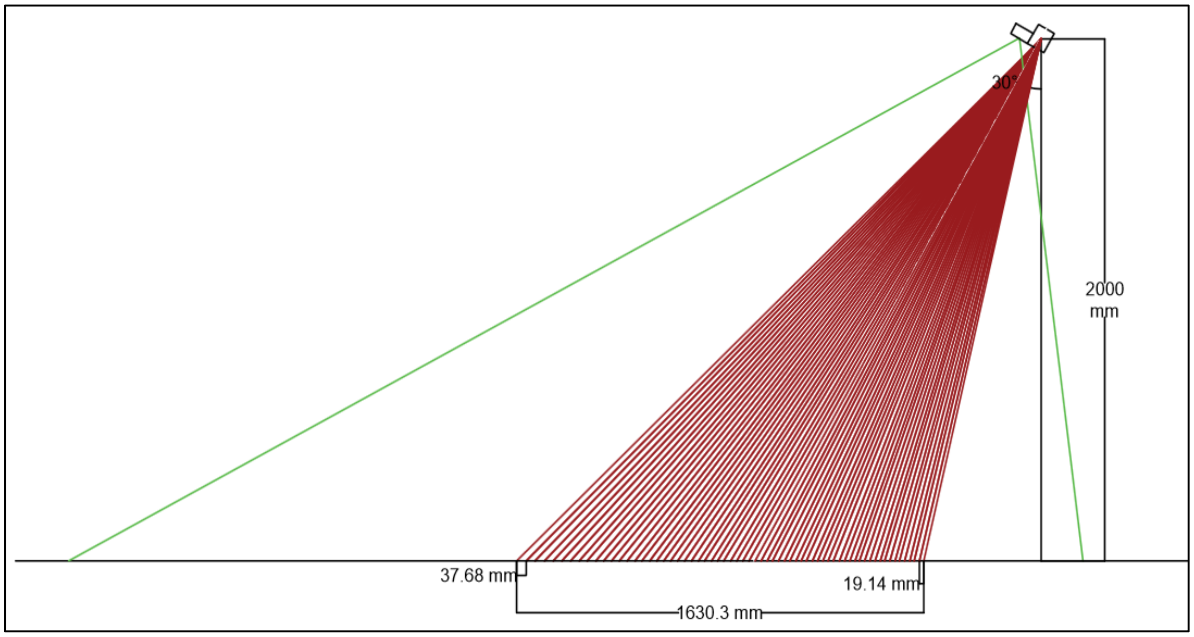


Figure 2.3 Ouster-64 channels LiDAR directed downward

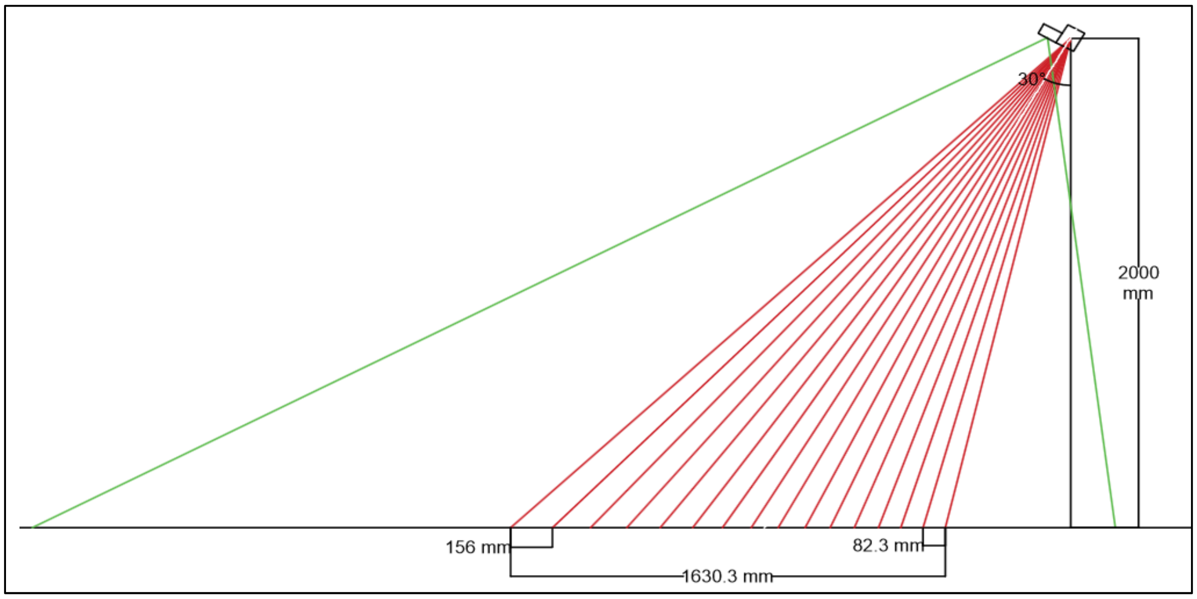


Figure 2.4 Ouster-16 channels LiDAR directed downward

When the LiDAR is directed downward, it is expected to extract road profiles with a distance ranging between 82 mm and 156 mm. At each LiDAR's rotation, 1.6 m transversal distance will be scanned by 16 laser beams. Each one rotation is considered a frame.

The resolution and the accuracy of 3D rotational LiDARs are not adequate to detect cracks. Therefore, by only using a 3D LiDAR, significant distress types are missed. To compensate the poor performance of the conventional LiDAR in detecting pavement distress, an additional technique is going to be used in this project along with the LiDAR. This technique will therefore improve the system's performance. The technique to be adopted is acquiring image data of the road.

2.3.3 2D Camera Selection

Photos captured from a camera can be mainly taken by two different methods: (1) taking still photos from a non-moving camera device and (2) taking images from a moving camera. Still photos taken by a non-moving camera are easier to capture (i.e., they can be taken by a smartphone cam such as in (L. Zhang et al., 2016). This method does not require sophisticated cameras to capture high-resolution clear images since there are no moving objects taken in still photos. As for the second method, recording moving images from a moving camera usually requires more sophisticated techniques in order to capture high speed moving objects images. High speed cameras are often used to capture far objects moving at high speed (e.g., capturing a racing car), and high-power strobe lights are used for near moving objects. These two techniques are used to reduce the effect of blurring resulted from the moving object. To illustrate that, a photo taken from a moving car at 100 km/h by a camera with a shutter speed equal to 1/2000 second results in a blurring image with 13.8 mm blurring effect (speed of the car times shutter speed, Equation 2.1). A faster shutter speed, such as 1/50000 second reduces the blurring effect to 0.5 mm which is enough to capture high-resolution pictures at 100 km/h with the right lighting conditions. If a camera does not have these specifications (fast shutter speed) strobe lights are used. These lights illuminate the targeted object with a high power light for a short amount of time (e.g., 1/50000 second), so even with a slower shutter speed the majority of light detected by the camera sensor results from the light reflected by the high power strobe light and not from the ambient light. This results in a freezing image for high speed near moving objects. Regarding far objects, this technique is not effective since the strobe light power is inversely proportional to the distance (i.e., the strobe light fades with more significant distance). The two aforementioned techniques are often used to capture high speed images; however, they cannot be adopted in this project since non-of these techniques is used for conventional self-driving cars.

$$Blur\ (mm) = Car\_Speed\ (km/h) * shutter\_speed\ (sec) * 277.77 \qquad (2.1)$$

Action cameras (e.g., GoPro, Sony dsc-rx0) are often used as dash-cams to capture high resolution videos; however, these cams have a slower shutter-speed comparing to high-speed cameras (e.g., Sony dsc-rx0 has a min shutter-speed equal to 1/32000 second, GoPro4 has a min shutter-speed equal to 1/8192 second). Thus, with a Sony dsc-rx0 at 100 km/h, a motion blur effect is obtained and it is equal to 0.86 mm and 3.39 mm for a GoPro4. However, research work has been developed to reduce motion blur effect from pictures; most recently, Kupyn et al. (2018) used *DeblurGAN* to recover sharp images from blurred ones. This model can be used in the application of this project to recover blurring images.

Different action cameras can be suitable for capturing 2D images such as: GoPro5-6-7, Sony dsc-rx0, YI 4K, etc. These camera prices can range between 200 USD for YI 4K and 600 USD for the Sony. In this research we use "Akasso V50" action camera.

2.3.4 Data Processing

From the 3D LiDAR sensor, 3D point cloud of the surrounding will be extracted. The 3D point cloud will be pre-processed to extract the surface of the pavement (i.e., all points that do not belong to the pavement surface will be ignored – for example, if the points belong to roadsides, road signs, trees, other vehicles, etc.), this is done by limiting the field of view of the LiDAR. Points belonging to the pavement surface will then be sampled into frames and each frame will cover a road section. The camera will return 2D video that will be pre-processed to extract frames. Each frame will be categorized as (1) a frame with pavement distress, or a frame with no pavement distress. The work will focus on cracks, yet, it can be further extended to detect different distress categories. A pre-trained Deep Convolutional Neural network such as VGG-16 (Simonyan & Zisserman, 2014), DensNet (G. Huang et al., 2017), or ResNet (K. He et al., 2016) will be used as feature generator, then the last layer of the network (the classifier) will be trained on the train set, and then tested on the validation set.

Therefore, in this project we followed three main phases: The first phase is data collection whereby a vehicle equipped with the appropriate equipment captures road images and pavement 3D profiles. The second phase consists of evaluating the LiDAR, i.e., its capability in detecting pavement distresses. The third phase tests the capacity of RGB images by adopting machine learning techniques in computer vision; a pre-trained deep convolutional neural network is adopted to extract features from the RGB frames, detect, and segment cracks.

To this end, the required data collection vehicle needs to be equipped first with 3D LiDAR, and camera. With these two sensors different information can be extracted from the road and can later be tested to detect pavement distresses.

2.3.5 Hardware Configuration

To acquire the road data we build a device that can be placed on top of a car. The device consists of a LiDAR and an action camera assembled together to a solid frame. Ouster os-16 channel is the LiDAR installed for data collection. It is based on time of flight technology. With this LiDAR, four different signals are acquired (Ouster, 2018): the first one is “range”, it measures the sensor distance from the target. The second one is “reflectivity”, it provides an indication of the target reflectivity by returning measurements scaled based on the measured range and sensor sensitivity at that range. The third one is “signal”, it measures the reflected signal intensity. The fourth one is “ambient”, it returns the measurement intensity from the ambient light (infrared sunlight with wavelength equal to 850 nm).

The hardware setup is mounted on the top of a car 2 m above the ground. The LiDAR and the camera are pointed toward the ground with an angle of 50 degrees. Both sensors cover a common 2 m x 4 m surface. The LiDAR scanning resolutions are 512 x 16 and 256 x 16 at 10 Hz and 20 Hz. The camera resolution is 1920 x1080 pixels at 120 frames per seconds. The car speed is up to 40 km per hour. Figure 2.5 shows the setup installed on the top of a car.

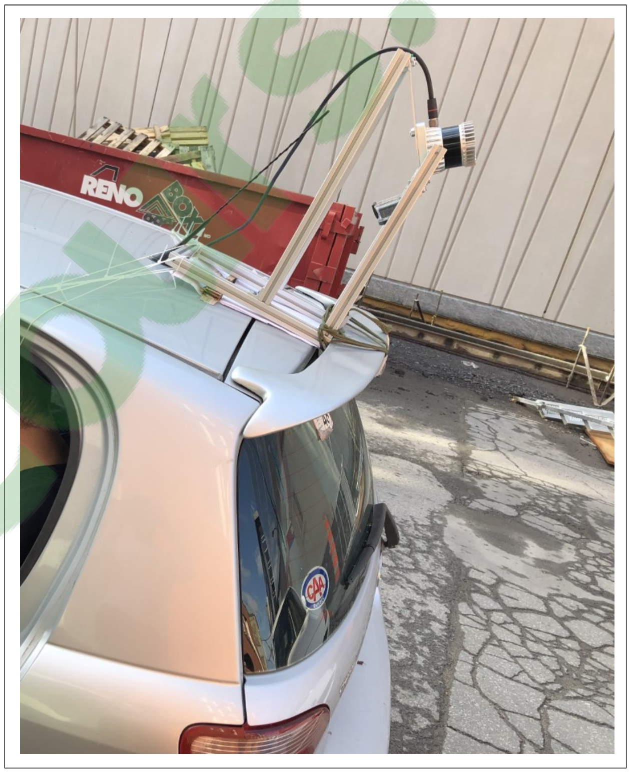


Figure 2.5 The LiDAR and the camera installed on the top of the car

CHAPTER 3

SELF-DRIVING CAR'S SENSORS: TESTS AND RESULTS

We conduct several experiments to test self-driving car's sensors capability in detecting pavement distresses. We first run indoor experiments on the LiDAR to test its accuracy and resolution. Then, we test its capacity in detecting pavement elevation. Then a second experiment on the camera is conducted. We collect pavement videos and we test a Region-based Deep Neural network in detecting cracks.

In this chapter, we present the experiments conducted on self-driving car's sensors, the results, and their interpretation.

3.1 Experiments on the LiDAR

To test the LiDAR accuracy, we conduct indoor and outdoor tests. In the indoor tests, we evaluate the effect of the reflectivity of a scan: We scan a high reflective surface (i.e., a white board) and a low-high reflective surface (i.e., chess board) and we evaluate the performance of the LiDAR in both experiments. We set the LiDAR 1.6 m above the ground and we collect measurements at 10 frames per seconds. We use three methods to assess the LiDAR accuracy: in the first one, we measure the Euclidian distance between a reference and the measured point cloud; in the second, we measure the distance between the point cloud and the best fit plane; in the third, we reconstruct the surface from the scanned point cloud and we evaluate the surface elevation of the reconstructed 3D map. In the outdoor tests, we collect LiDAR data from a moving vehicle and we test the measured point clouds in different scenarios by reconstructing the pavement surface and evaluating its elevation.

The 3D measurements acquired from the LiDAR are in range format, i.e., the measurements represent how far is the LiDAR from the measurement point. To convert the measured range into 3D coordinates we follow Equation 3.1 extracted from Ouster manual (Ouster, 2018): *r* is the measured range in mm, θ is the azimuth angle in radian, 90112 is the encoder ticks maximum number, φ is the beam altitude angle in radian, i is the number of channels (i.e., [1; 16]), x, y, and z are the 3D coordinates of the measured points. We use the 3D coordinates to generate a point cloud and compare it to a reference point cloud and a reference plane.

$$
\begin{gathered}
r = range\ mm \\
\theta = 2\pi \left(\frac{encoder\_count}{90112}\right) \\
\varphi = 2\pi \,\frac{beam\_altitude\_angles[i]}{360} \\
x = r\cos(\theta)\cos(\varphi) \\
\mathrm{y} = -\mathrm{r}\sin(\theta)\cos(\varphi) \\
z = r\, sin(\varphi)
\end{gathered}
\tag{3.1}
$$

3.1.1 Indoor Experiments

3.1.1.1 Cloud to Cloud/Mesh Distance

We create two indoor environments to test the accuracy of the LiDAR. In the first one, we scan a white board. By doing so we assess the accuracy of the scanned point cloud while scanning a surface with high reflectivity. In the second environment, we scan a mixed reflectivity board, Figure 3.1. We use CloudCompare software (CloudCompare, 2018) to measure the cloud to cloud (C2C) and cloud to mesh (C2M) distance to assess the accuracy of the LiDAR. C2C measures the distance between an input point in the point cloud and its nearest in the reference point cloud. C2M measures the distance between an input point in the point cloud and a mesh (plane surface). By computing the discrepancy between the scanned point cloud and its references we assess the accuracy of the LiDAR measurements.

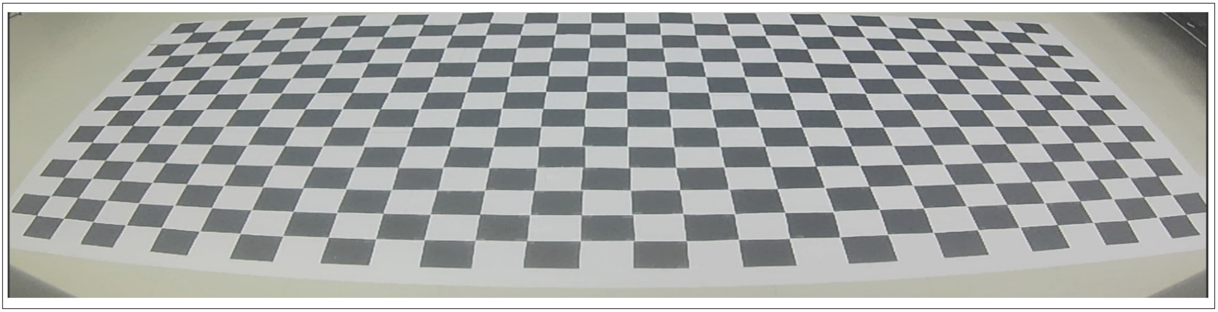


Figure 3.1 Environment 2, surface with mixed reflectivity

The generated point clouds for both environments are shown in figures 3.2 and 3.3. To test the accuracy of the LiDAR, we compute the Euclidian distance between the generated point cloud and the reference point cloud, Figure 3.4. The reference point cloud is generated by simulating the LiDAR beams firing into a plane surface 1.6 m below the LiDAR. Equation 3.2 represents the parametric equation of the beams firing on a flat surface. This equation is obtained by calculating the trajectories that the LiDAR beams take while scanning a surface. The trajectories are based on the intersection of cones with a plane surface. The LIDAR position is the cone vertex. The variables in Equation 3.2 are defined as follows: x, y, and z are the 3D coordinates of the points in the reference point cloud. φ is a constant value that represents the LiDAR angle in which it is directed to the ground, d is a constant value that represents the sensor height to the ground. δ is the channel angle as defined in the data sheet of the sensor (i.e., δ has 16 angle values that represents the distribution of the laser beams inside the LiDAR), ANNEX II. And θ is the angle where the beam is fired (i.e., θ has 512 or 256 values depending on the vertical resolution of the scan), it varies between -90^o and 90^o with a fixed step (i.e., 0.7^o for 256 resolution and 0.35^o for a 512 resolution).

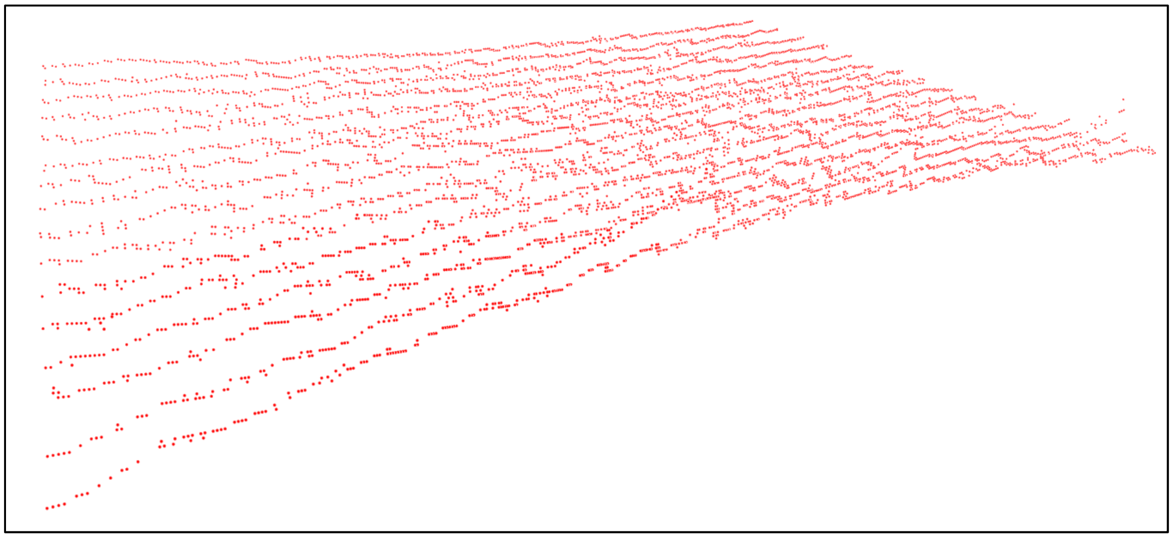


Figure 3.2 Environment 1, high reflectivity point cloud

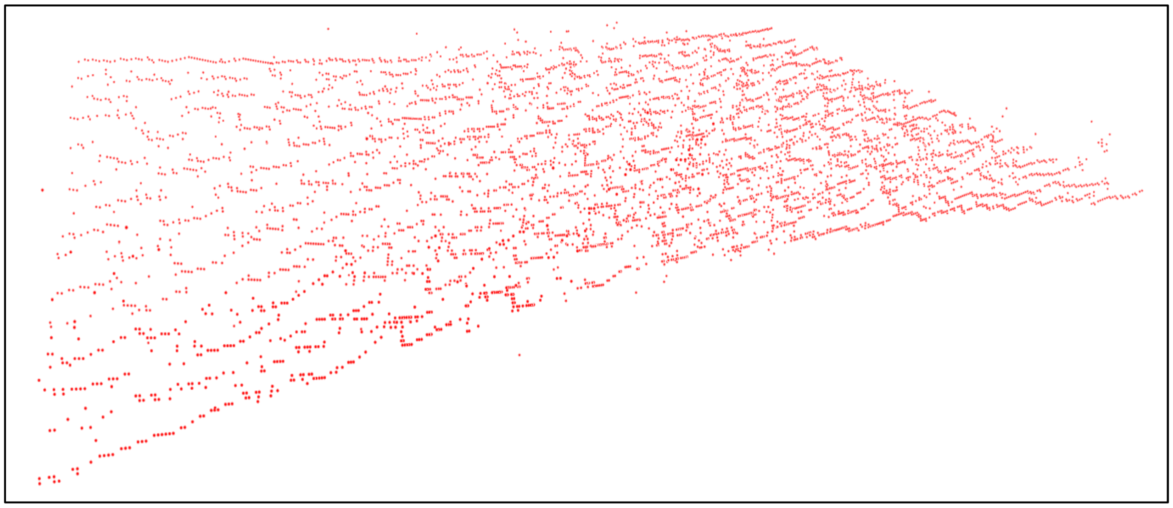


Figure 3.3 Environment 2, mixed reflectivity point cloud

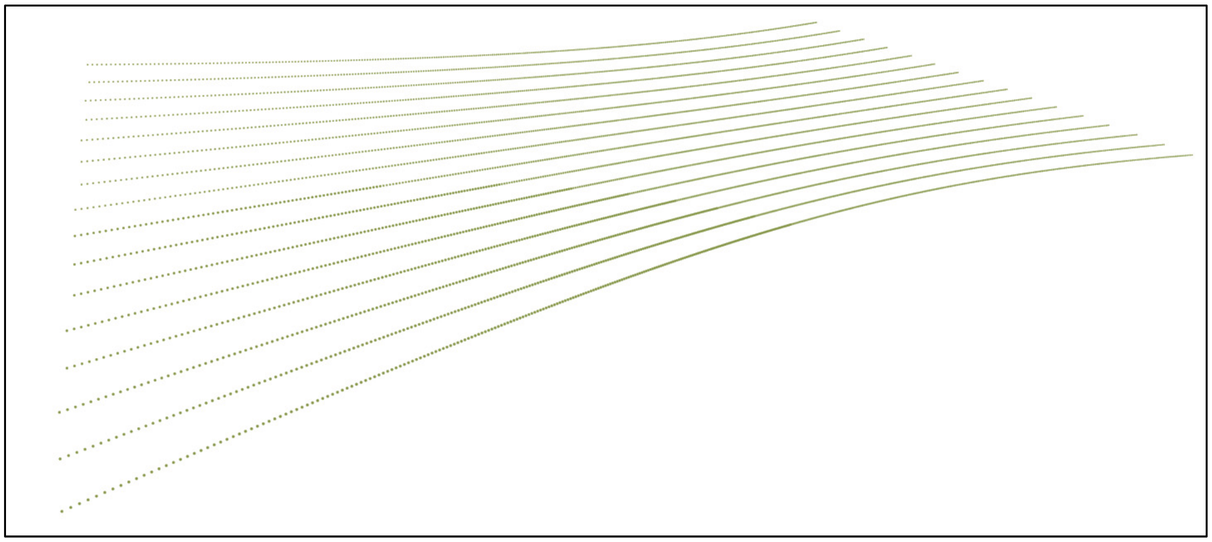


Figure 3.4 Reference point cloud

$$x = \frac{\cos\delta * \cos\theta * \frac{\frac{-d}{\cos\varphi}}{\sin\delta}}{1 - \frac{\tan\varphi * \cos\delta * \cos\theta}{\sin\delta}} \tag{3.2}$$

$$y = \pm\sqrt{abs(\frac{(x * \tan\varphi - \frac{d}{\cos\varphi})^2}{(\tan\delta)^2} - x^2)}$$

$$z = -x * \tan\varphi - \frac{d}{\cos\varphi}$$

To compute the accuracy of the LiDAR we measure the absolute distance between the scanned and the reference point clouds. The absolute distance computed between the reference and the scanned point cloud is based on the nearest neighbor distance, i.e., for each point on the scanned point, CloudCompare computes the Euclidian distance between a target point and its nearest in the reference point cloud, Equation 3.3. (x_{comp}, y_{comp}, z_{comp}) and (x_{ref}, y_{ref}, z_{ref}) are the 3D coordinates of an input point in the measured point cloud (compared point cloud) and its reference respectively. Figure 3.5 shows the nearest distance between the reference and the scanned point cloud. Figure 3.6 represents the error between the reference point cloud and the high-reflectivity surface point cloud. Figure 3.8 shows the error between the mixed-reflectivity surface and its reference.

$$d = \sqrt{\left(x_{comp} - x_{ref}\right)^2 + \left(y_{comp} - y_{ref}\right)^2 + \left(z_{comp} - z_{ref}\right)^2} \tag{3.3}$$

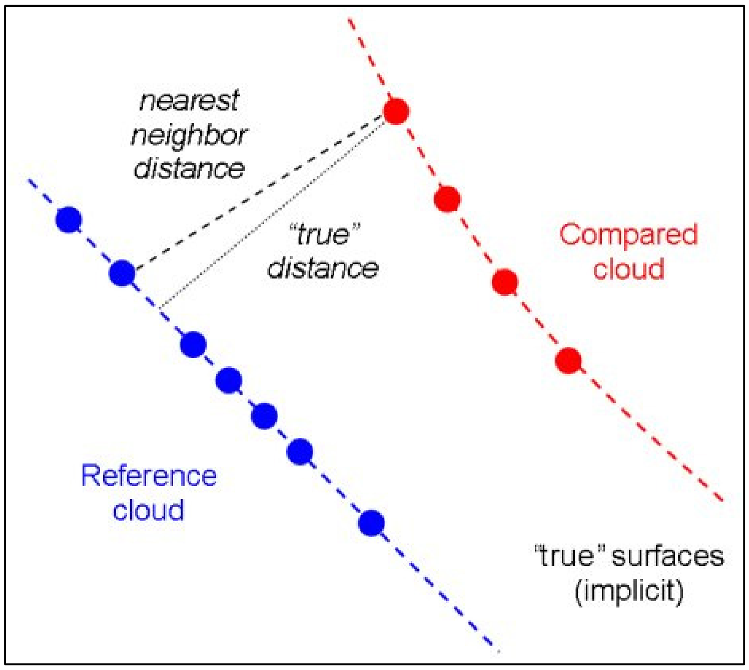


Figure 3.5 Cloud to cloud distance
Taken from CloudCompare

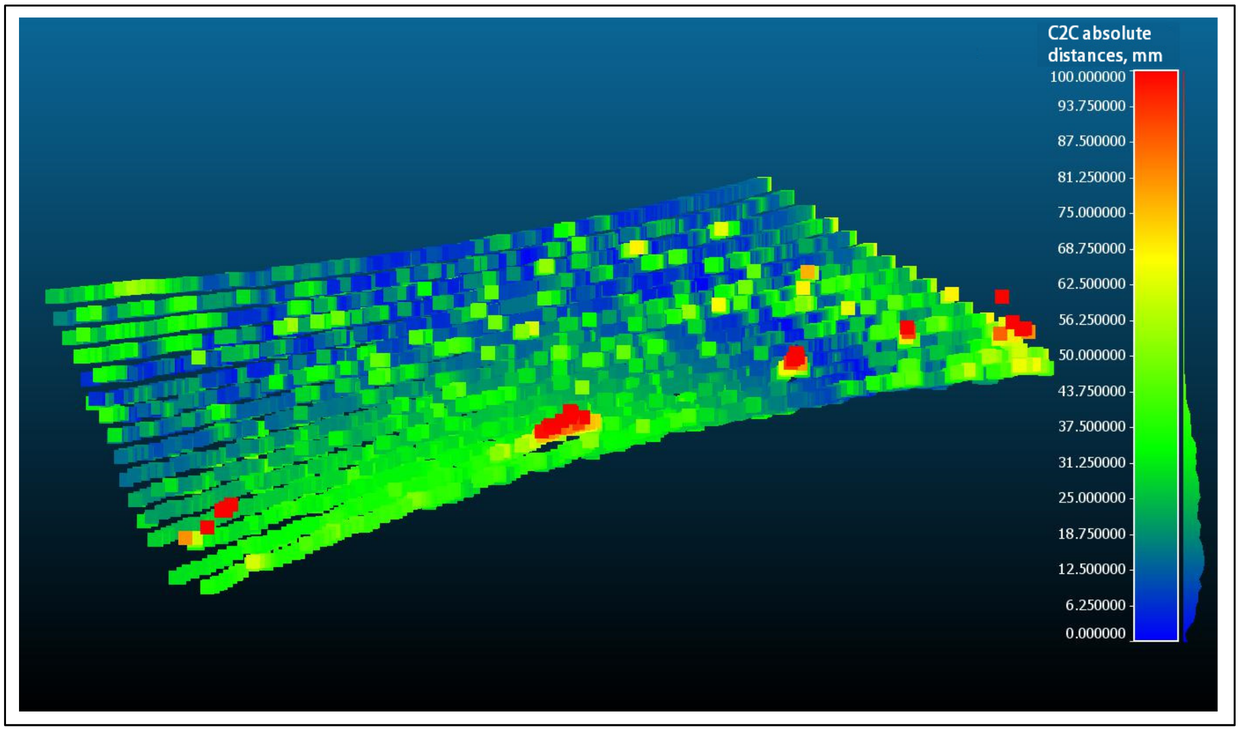


Figure 3.6 Cloud to Cloud distance. High reflectivity surface compared to its reference point cloud

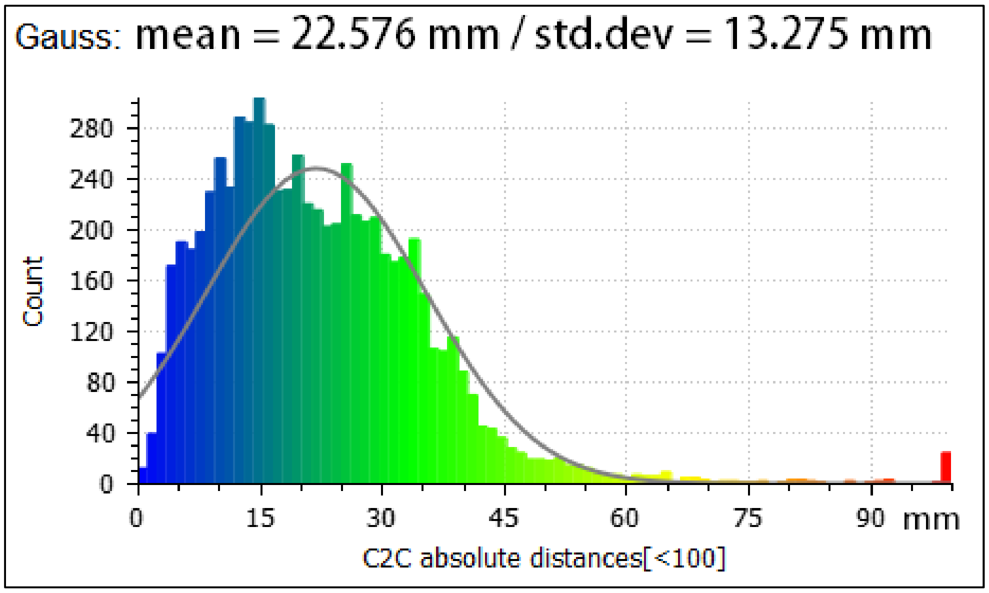


Figure 3.7 Histogram of the absolute distance between the scanned point cloud (high-reflectivity surface) and its reference

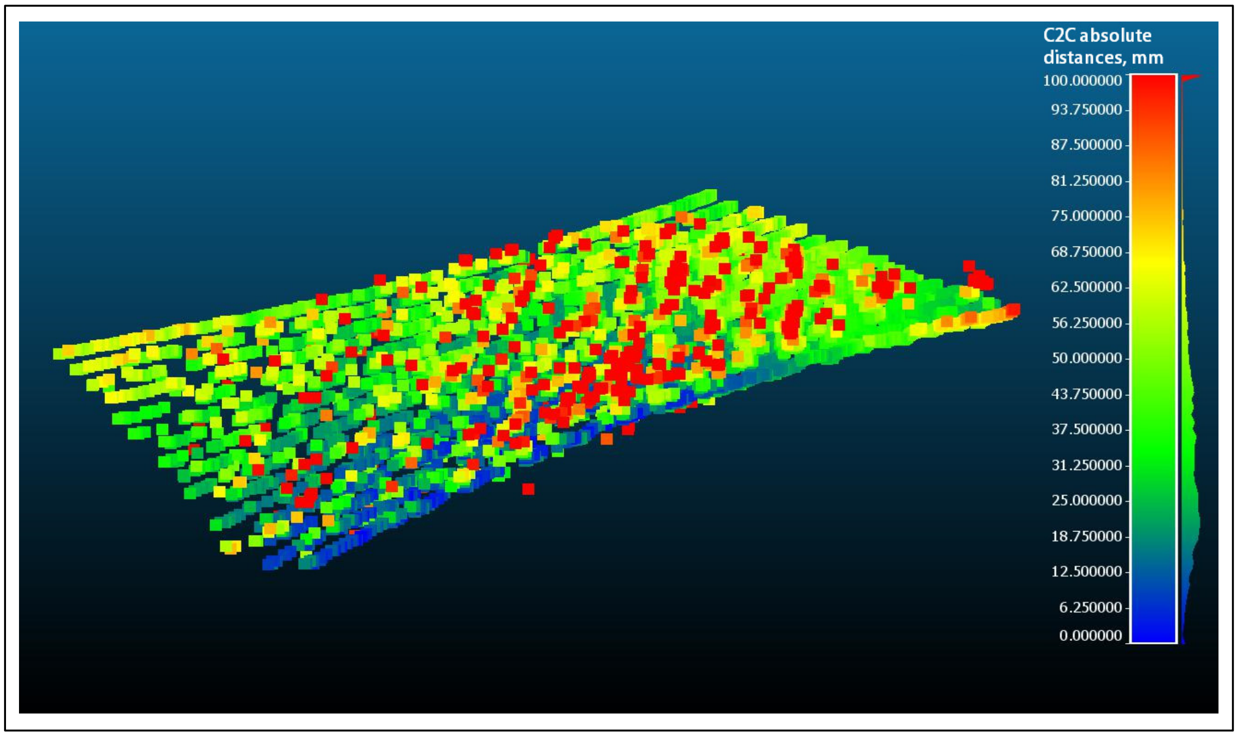


Figure 3.8 Cloud to Cloud distance. Chess board compared to its reference point cloud

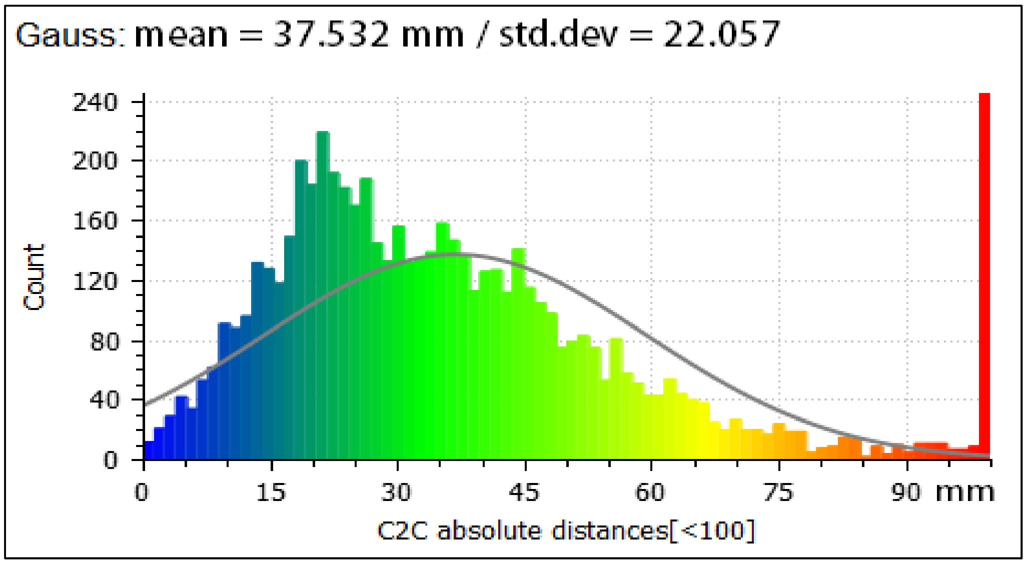


Figure 3.9 Histogram of the absolute distance between the scanned point cloud (mixed-reflectivity surface) and its point cloud reference

To compute the error between the scanned and the reference plane, we use the fit plane feature in CloudCompare software (CloudCompare, 2018) to fit a plane on the scanned point cloud and we measure the distance between each point and the fitted plane. Figures 3.10 to 3.13 show the distribution of the distance between the plane and the point cloud.

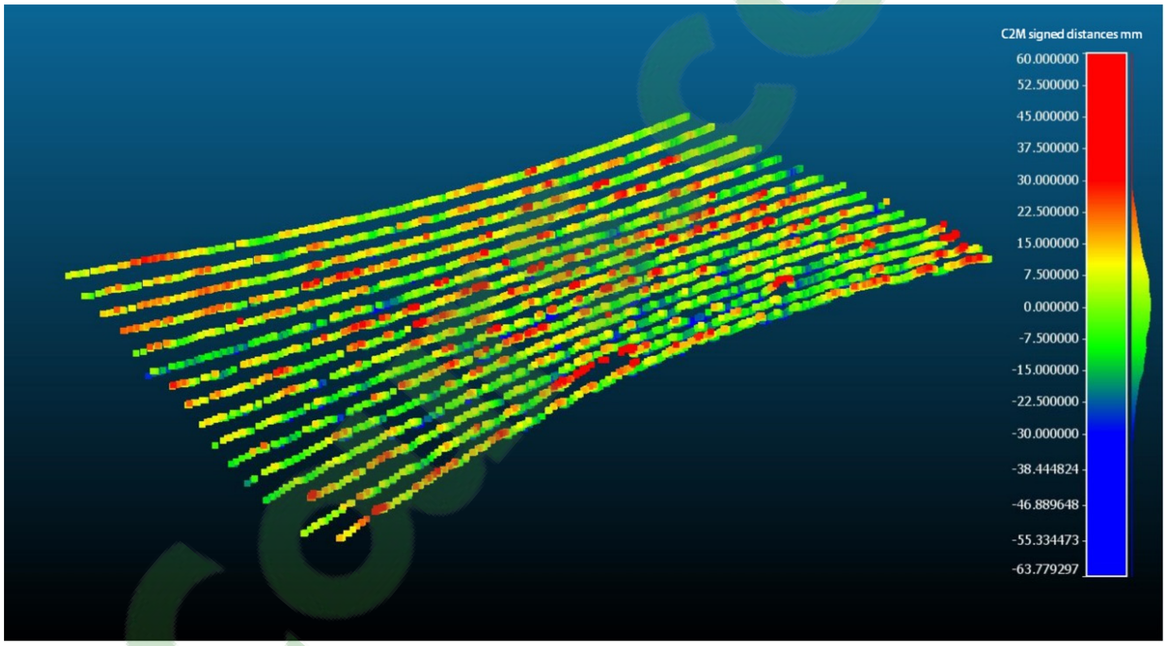


Figure 3.10 Cloud to mesh distance. High-reflectivity surface

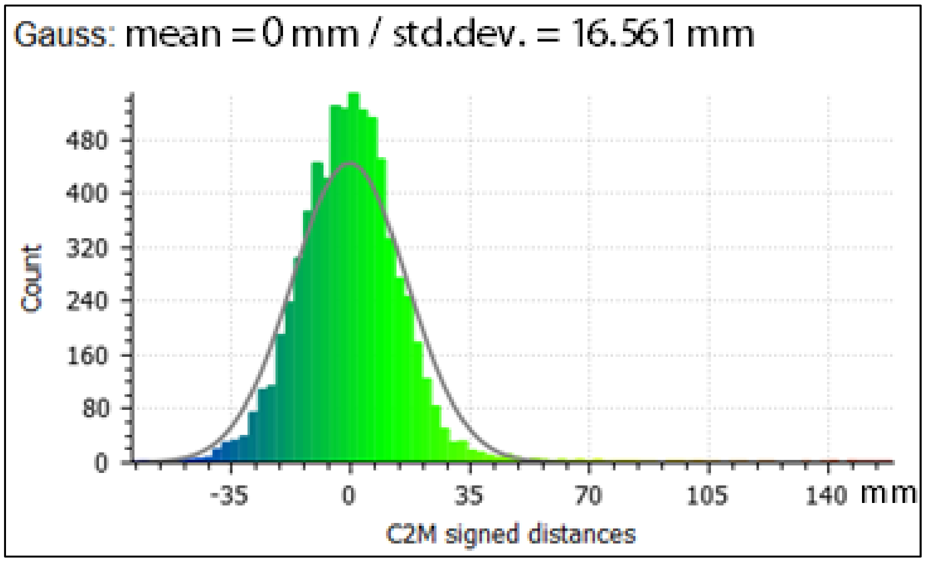


Figure 3.11 Histogram of the absolute distance between the scanned point cloud (high-reflectivity surface) and its plane reference

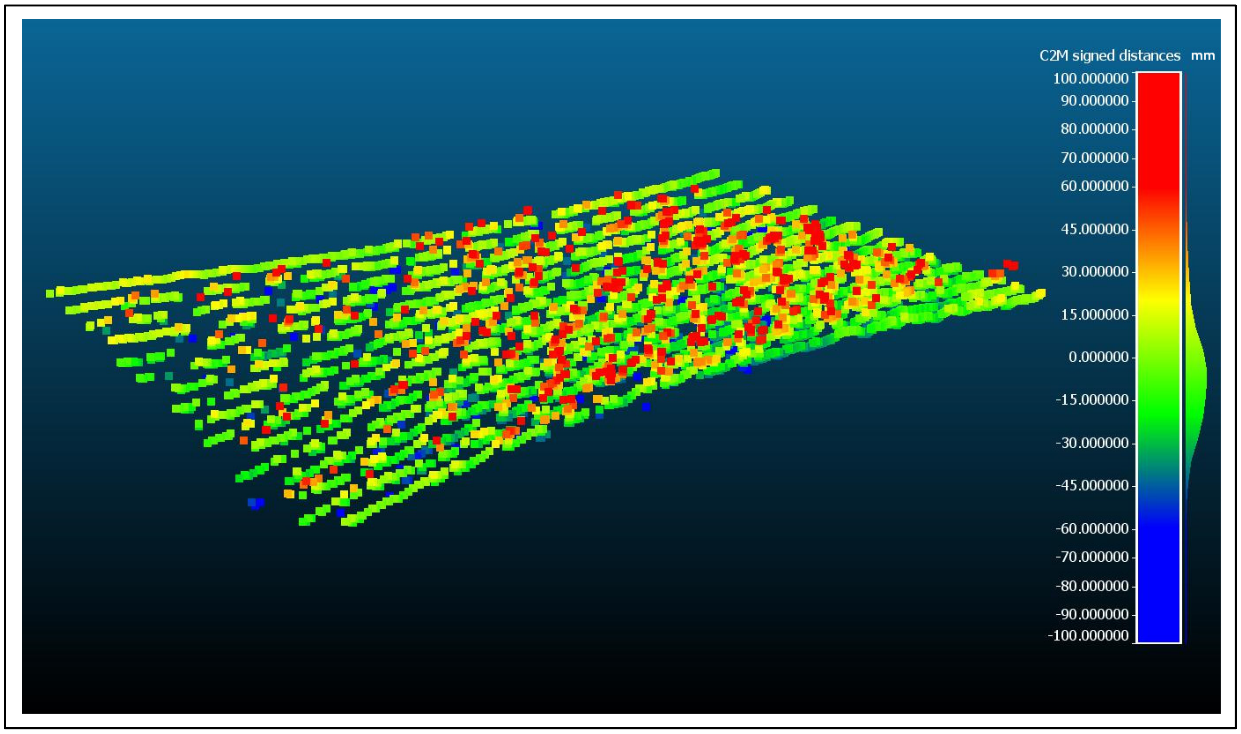


Figure 3.12 Cloud to mesh distance. Mixed-reflectivity surface

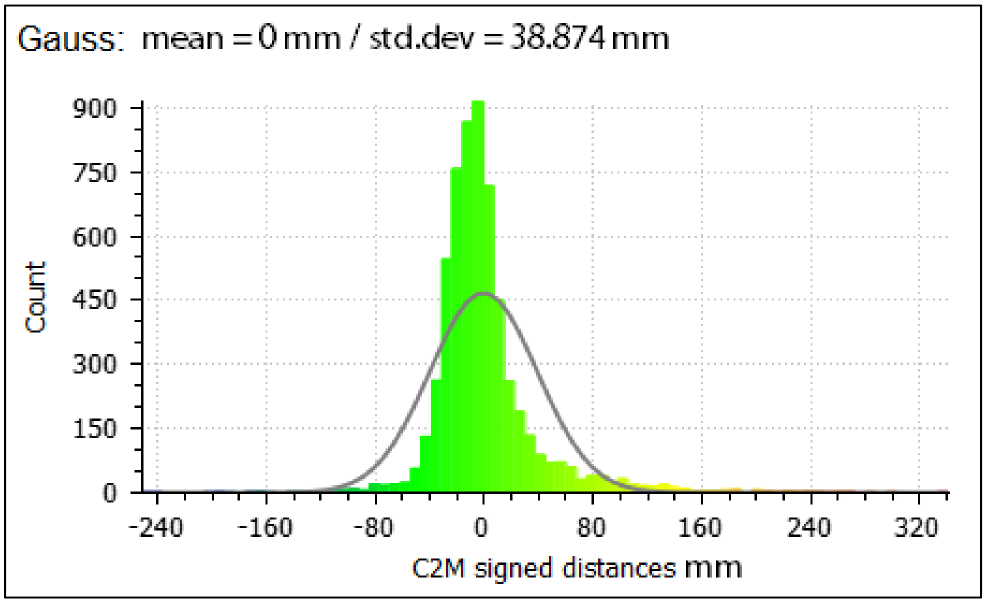


Figure 3.13 Histogram of the signed distance between the scanned point cloud (mixed-reflectivity surface) and its reference

3.1.1.2 Results Interpretation

Cloud to cloud distance returns the absolute distance between points in the scan and their reference point cloud. The number of points in each point cloud is around 7000 points. While scanning a high reflectivity surface, the mean error is equal to 22.5 mm, and the standard deviation is equal to 13.27 mm (Figure 3.7). Thus, the accuracy of the LiDAR is 22.5 mm ± 13.27 mm. Moreover, we noticed that more than 80% of the points have an error larger than 10 mm. The accuracy of the LiDAR drops to 37.53 mm while introducing low-reflectivity objects to the surface (i.e., scanning the chess board), and the precision also drops to 22 mm. A significant increasing in outliers appears while scanning a low reflective surface (Figure 3.9).

Similar results appear while we measure the distance between the scanned point cloud and the best fitted plane. The precision drops from 16.5 mm on a high reflectivity surface to 38.8 mm while scanning a surface with mixed reflectivity. Figures 3.8 and 3.12 show that the drop in the precision is mainly due to points located on a low reflectivity surface. From these tests we notice 37% of the points scanned on a mixed reflectivity surface, and 15% of the points scanned on a high reflectivity surface have an error larger than 20 mm. Thus, a low reflectivity surface degrades the accuracy of the measurements.

3.1.1.3 Screened Poisson Surface Reconstruction

To reconstruct a surface from the point cloud, we should first study the characteristics of the input point cloud. According to Berger et al. (2016), many factors affect the surface reconstruction, some of these factors are: point cloud density, noisy data, outliers, missing data, etc. Figure 3.14 extracted from Berger et al. (2016) shows some of the factors that affect the reconstruction.

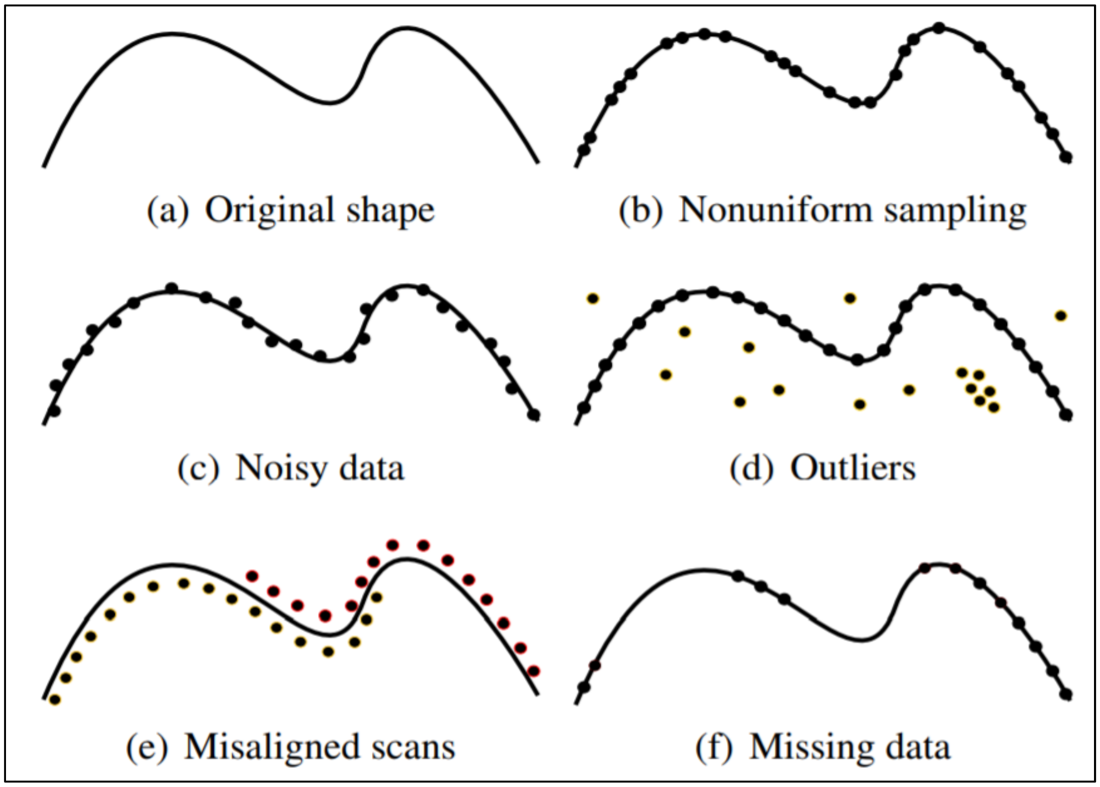


Figure 3.14 Different types of factors that affect the reconstruction of the surface
Taken from Berger et al. (2016)

The scanned point cloud is characterized by several types of deformation: it is a low resolution point cloud i.e., the scanner profiles do not cover the entire surface. It has also a noisy data where random points are distributed close to the surface and outliers far from the true surface. According to Berger et al. (2016), Screened Poisson surface reconstruction (Kazhdan & Hoppe, 2013) is robust to noise, missing data, and outliers.

Screened Poisson Surface reconstruction consists in reconstructing a watertight mesh from oriented point cloud, i.e., point set with normal vectors, to evaluate the elevation of the reconstructed mesh. For each point (input point) in the point cloud, a normal vector is computed by fitting a sub plane into a subset of points. The subset of points consists of 50 neighboring points to the input point. After reconstructing the surface, we compute the depth map, i.e. we measure the elevation along the z axes. Figures 3.15 and 3.16 show the results on the high reflectivity and mixed reflectivity point clouds.

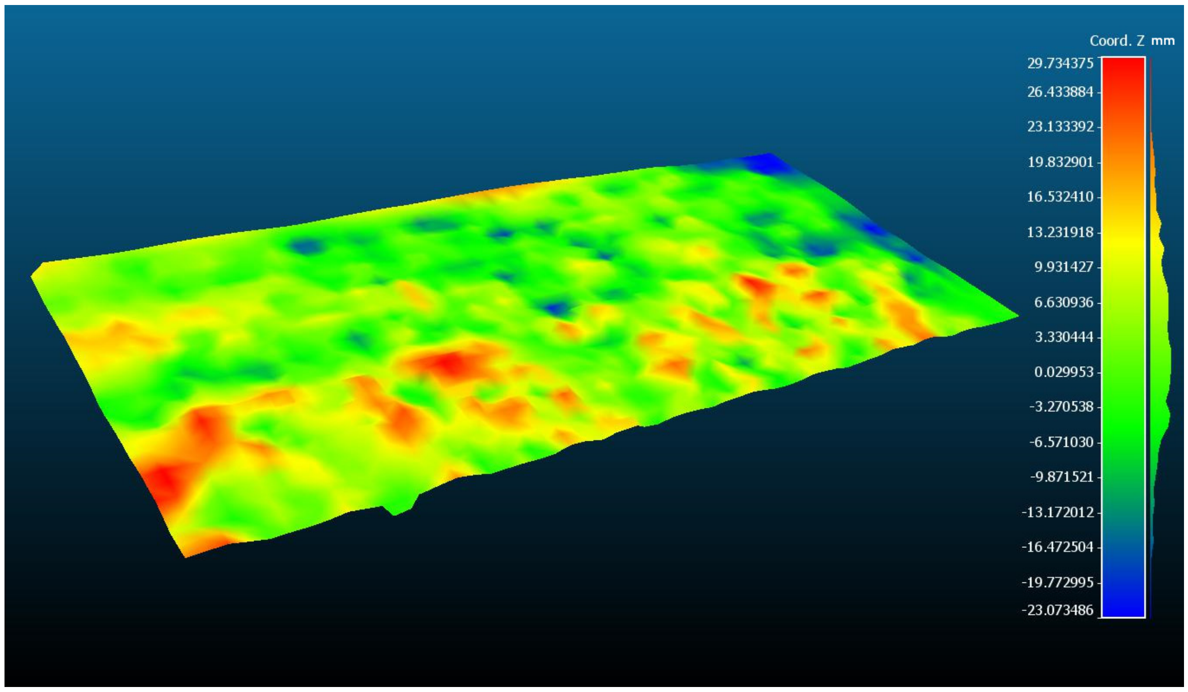


Figure 3.15 Reconstructed map of the high-reflective surface

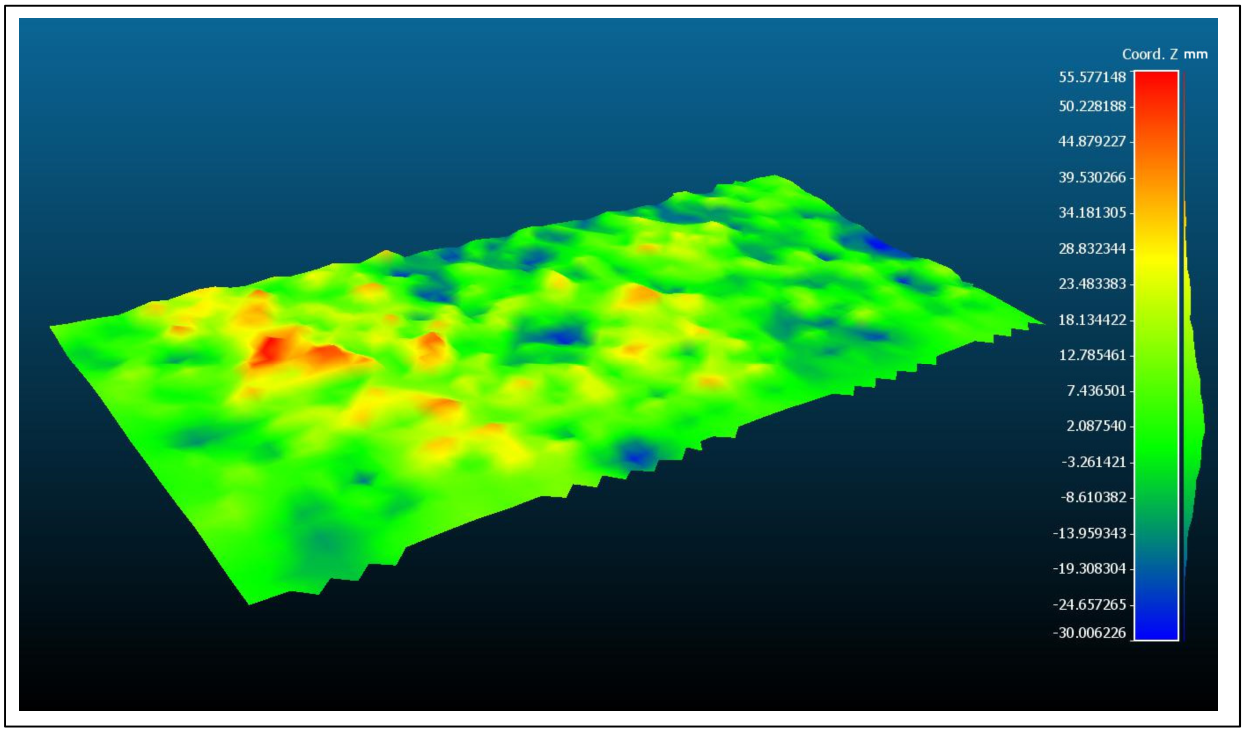


Figure 3.16 Reconstructed map of the mixed-reflective surface

3.1.1.4 Results Interpretation

The elevation of the reconstructed surface shows a discrepancy between the measurements and the real surface. Instead of having a plane surface, significant deformations appear in the reconstructed surface. The deformations are greater in the mixed reflectivity board. This is due to the low accuracy measurements resulting from the inherent characteristics of the LiDAR, the low resolution and the low reflectivity surface.

3.1.2 Outdoor Experiments

We conduct outdoor experiments to evaluate the LiDAR performance in capturing depth information in a real environment. We scan the road with the LiDAR mounted on the top of a car 2 m above the ground. The LiDAR is pointed toward the ground with an angle of 50 degrees and covers 2 m x 4 m surface. The LiDAR scanning resolution is 512 x 16 at 10 Hz. The car speed is up to 40 km per hour.

As mentioned in Section 2.3.5, the LiDAR returns four different types of information: ambient, reflectivity, signal and depth data. The first three information are 2D information and do not reflect the depth of the surface. To evaluate the quality of the first three information, we compare them to their synchronized RGB images. As expected, the LiDAR returns low resolution frames, i.e., 16 x 256 or 16 x 512 points depending on the LiDAR scanning speed. The LiDAR frames lack crucial details. Figures 3.17 and 3.18 show the ambient light, the reflectivity and the RGB frames. We notice that the LiDAR 2D information frame does not return any additional data to the RGB one; rather, the LiDAR frame lacks details such as colors, objects, and distresses. This is caused by the low LiDAR point density.

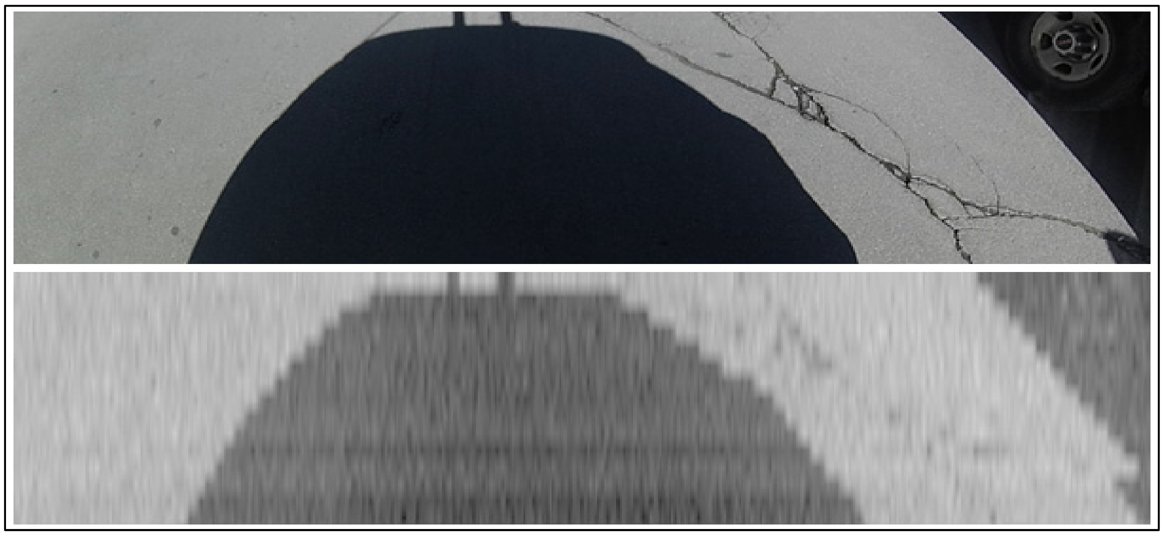


Figure 3.17 RGB frame with its equivalent ambient LiDAR data

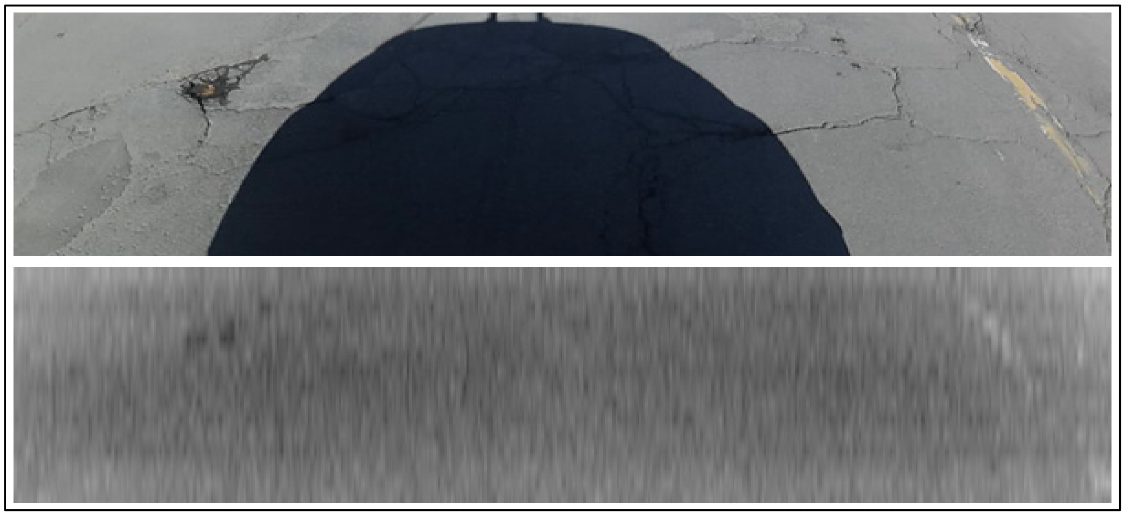


Figure 3.18 RGB frame with its equivalent reflectivity LiDAR data

3.1.2.1 Depth Information from the Reconstructed Surface

To assess the depth data, we conduct the same experiments done in Section 3.1.1.3 (Screened Poisson Surface Reconstruction). We reconstruct the surface and we measure the pavement depth. We sample multiple surfaces from the collected data. The samples belong to pavements with potholes with different sizes and depth and pavement with good conditions. The results of the reconstruction are shown in figures 3.20, 3.22, and 3.24, with their equivalent RGB, and LiDAR reflectivity frames, figures 3.19, 3.21, and 3.23.

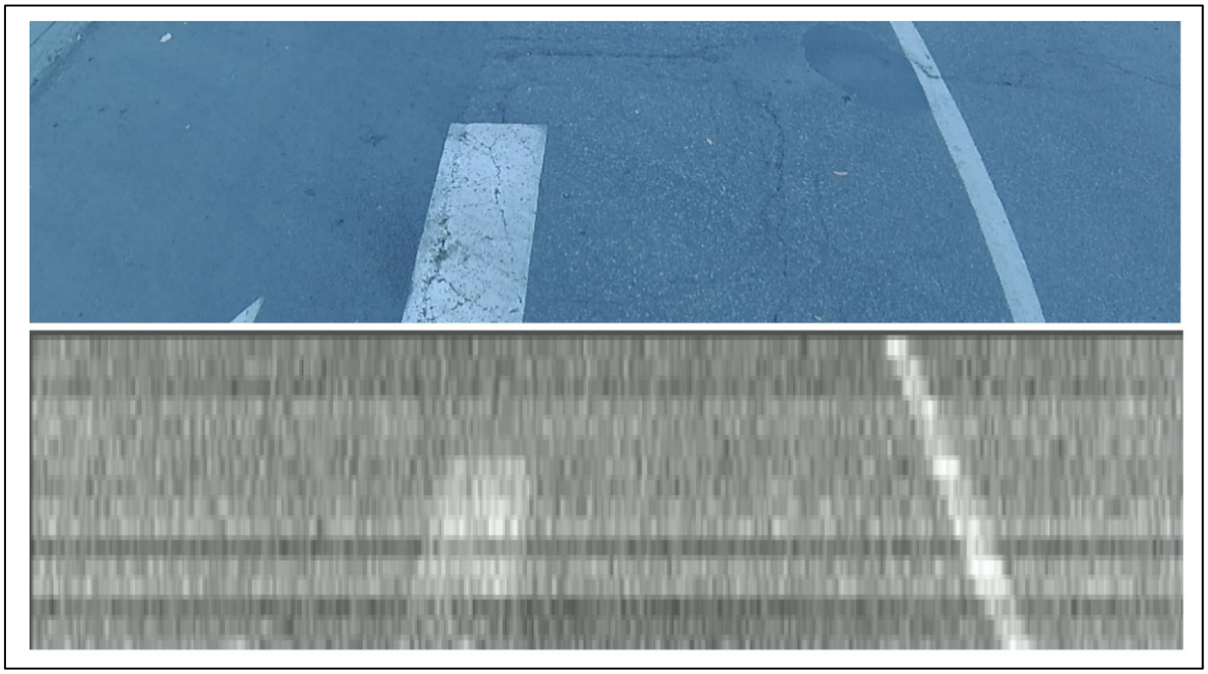


Figure 3.19 RGB frame (top) and its LiDAR equivalent (bottom);
pavement with no depth deformation

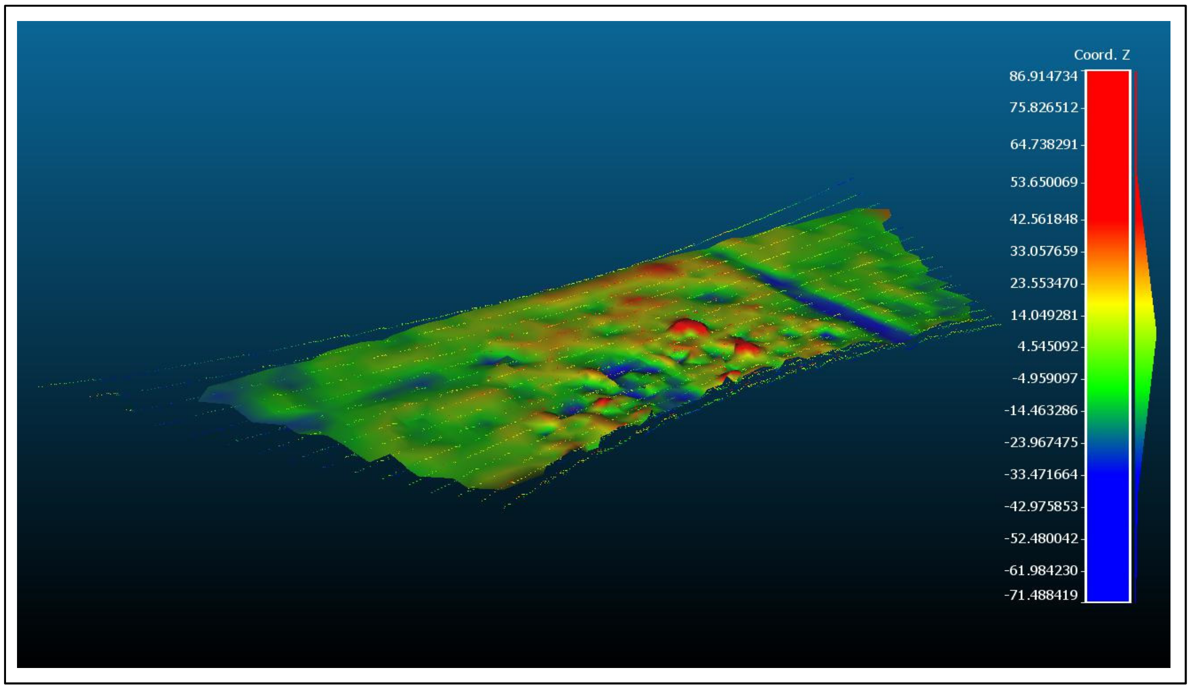


Figure 3.20 Pavement reconstruction with faulty depth deformation

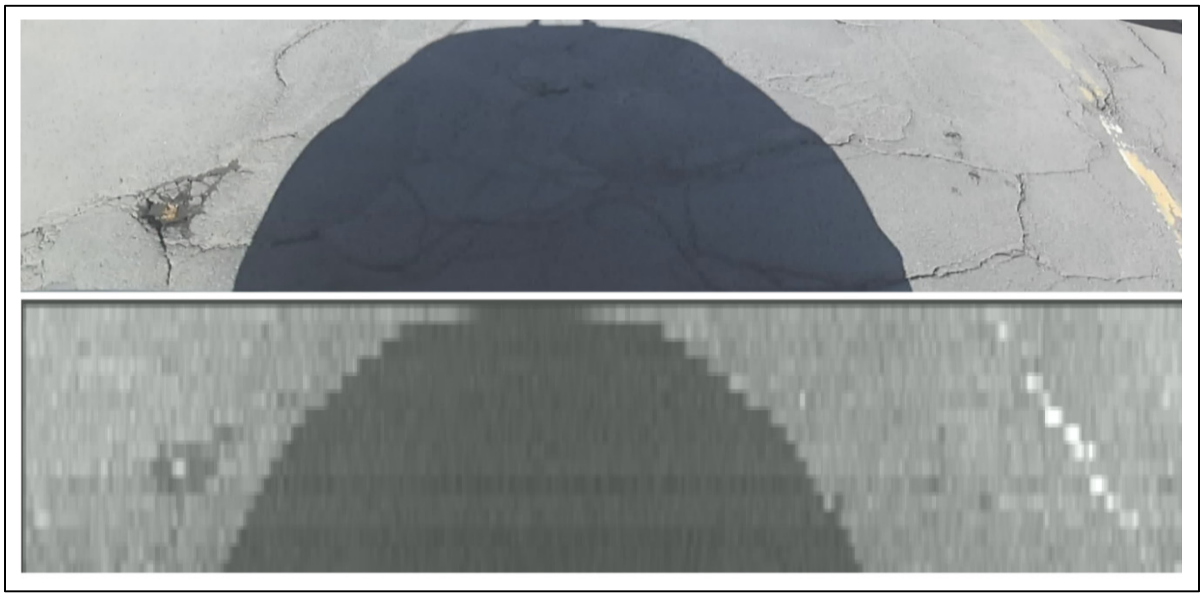


Figure 3.21 RGB frame (top) and its LiDAR equivalent (bottom); pavement with a low severity pothole

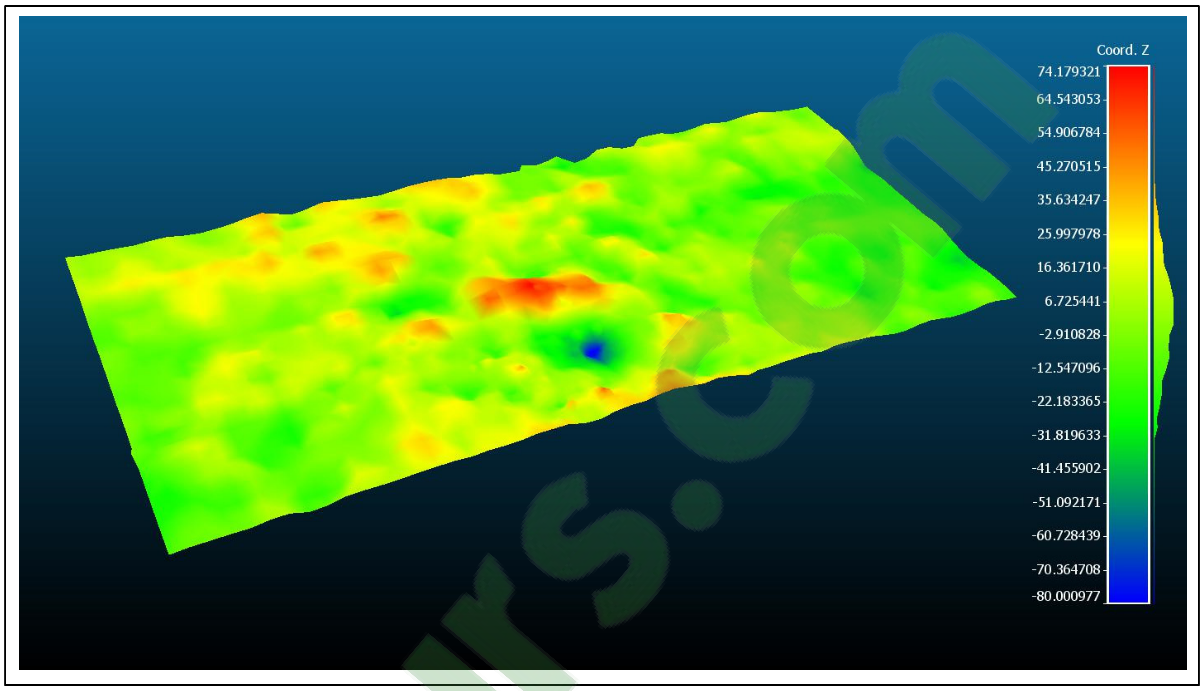


Figure 3.22 Pavement reconstruction with low severity pothole, the blue and red areas are faulty detections

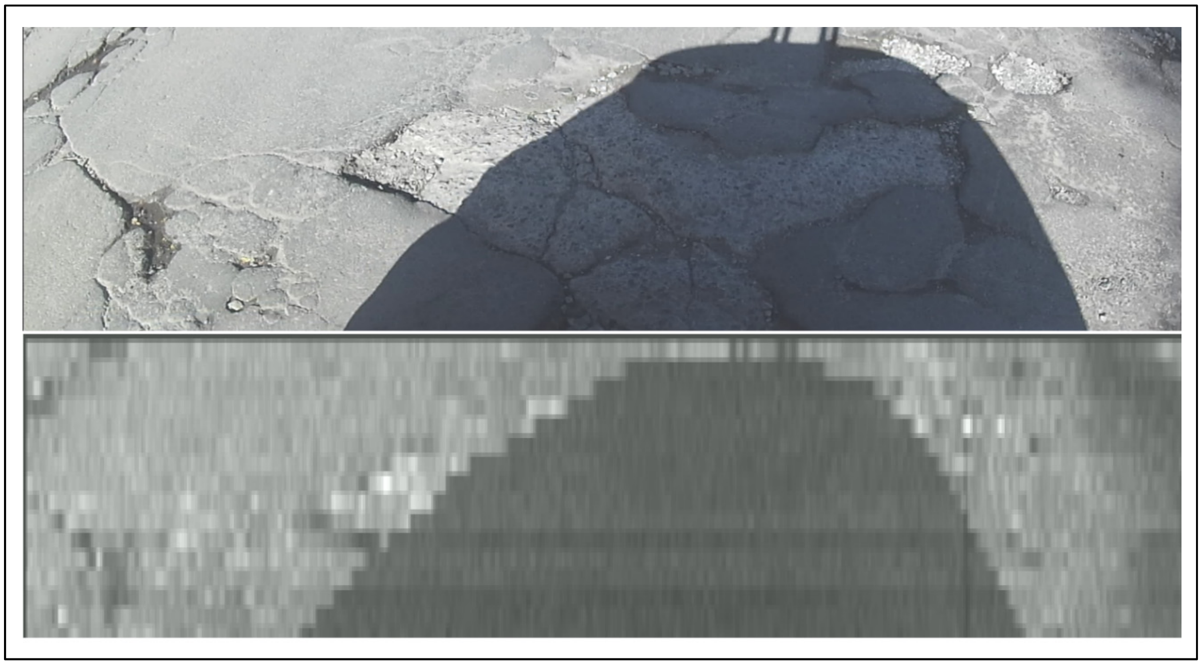


Figure 3.23 RGB frame (top) and its LiDAR equivalent (bottom); pavement with high severity pothole

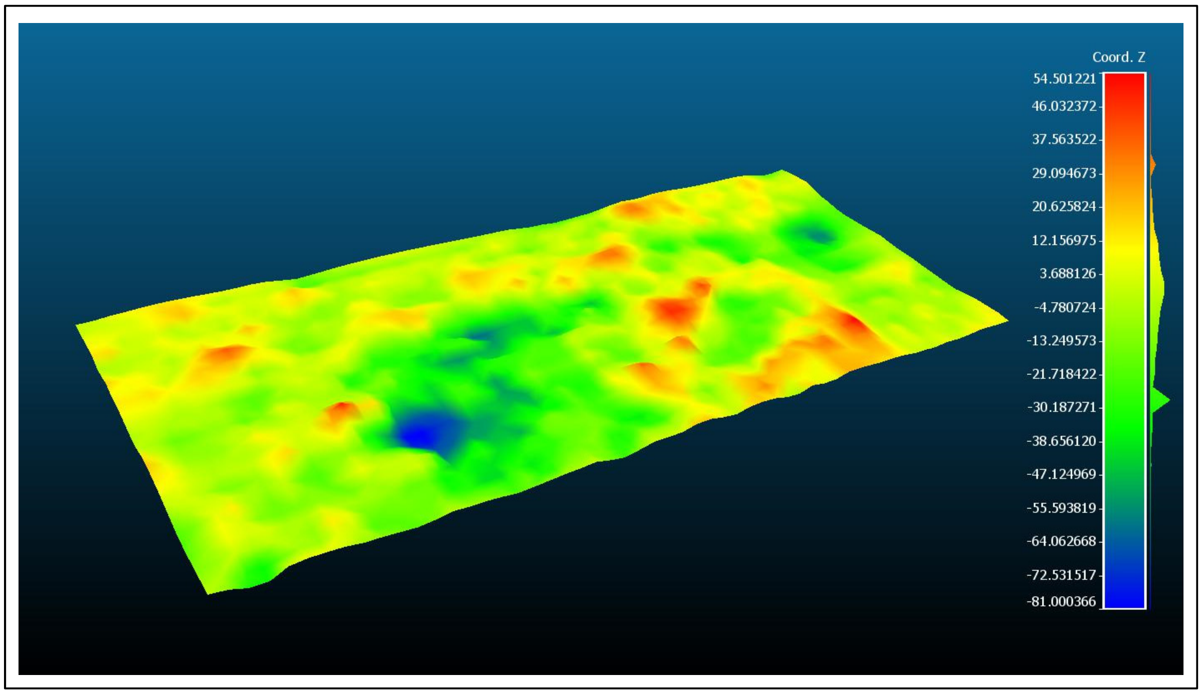


Figure 3.24 Pavement reconstruction with high severity pothole

3.1.2.2 Results Interpretation

The pavement reconstruction results show faulty deformations on the pavement surface. Frames with potholes have no indication of deformation where it is expected to be. This is due to the following reasons:

- the resolution affects the point density. That means a deformation (e.g. pothole) totaling 100 mm x 100 mm, may be only targeted by 16 or 32 points depending on where it falls during the scan and how many channels it is hitting by (a 100 mm x 100 mm object can be hit by one or two laser channels);
- the accuracy of the LiDAR is low, it varies between 10 mm on a high reflective surface with 10 mm standard deviation, and tends to degrade significantly on a low reflective surface. Low reflective surface has a significant impact on the measurements and surface reconstruction, most of the outlier points are located on low reflective surface.

While smoothing can help removing noises from the scan, it also tends to deteriorate the quality or the details in the measurements, i.e. depth information are lost.

3.1.3 Conclusion

The LiDAR data show a poor capacity in detecting pavement distresses such as potholes and cracks. Both in-laboratory and outdoor experiments show low quality data with low accuracy and low precision. Even though the accuracy and the resolution are acceptable for self-driving car applications, such as depth perception, it is not suitable to capture fine details nor depth information from the pavement distress.

3.2 RGB Camera

To collect pavement data, we install a low-cost action camera on a car. We record videos to arterial roads at a top speed of 40 km/h. Action cameras are widely used to capture action footage from the point of view of the shooter, they are compact and easy to install. They also offer an optimal solution between the price and the video quality. The action camera is set at 120 frames per second to reduce the blurring effect while driving at 40 km/h, and at a resolution of 2048 x 1080 pixel to capture fine details. The camera points toward the pavement with a 40-degree angle. This setup allows to capture a 4 m x 2 m road section with a moderate pixel density, i.e., 1 pixel covers approximately 1 mm. A deep Neural Network is then trained to detect and segment cracks from the collected data.

In the next section, we present the architecture of the machine learning model based on Region Proposal Neural Network, RPN, and Deep Neural Network, DNN.

3.2.1 Object Detection and Instance Segmentation with RPN

Object detection is the task of finding an object inside an image. Object detection returns the object bounding box (i.e. object localization) and its category (i.e. object classification). Several methods are developed to achieve localization and classification tasks. The traditional pipeline developed for region convolution neural network is based on three main steps: (1) region proposals, (2) feature extraction, and (3) classification. Further enhancement is added to the pipeline by He et al. (2017) in Mask R-CNN paper, thus, instead of returning a bounding box, the boundaries of the object are also detected; this detection is called instance segmentation.

3.2.1.1 Regions with Convolutional Neural Network (R-CNN)

Region proposal-based methods based on rich feature extraction, also known as Region Convolutional Neural Network (R-CNN), was first introduced by Girshick et al. (2014). R-CNN consists of three consecutive modules. The first module generates region proposals using selective search method (Uijlings et al., 2013). It first generates many candidate regions based on classic computer vision algorithms, then it uses greedy algorithm to merge related regions into larger one, then generates proposals (Uijlings et al., 2013); these proposals (2000 proposals per image) define the set of candidates region. The next module is the feature extractor which is a convolutional neural network that extracts rich feature hierarchies from the suggested region. The last module consists of an object category classifier. The architecture of R-CNN is presented in Figure 3.25. Although R-CNN is able to detect objects with a mean average precision (mAP) equal to 66.0%, it faces several drawbacks. R-CNN has a slow multi-stage training speed. Also, the test-time is very slow (47s per image on NVIDIA K40 GPU), thus, it cannot be implemented in real time. Moreover, the selective search adopted in the first module is a non-trainable algorithm, therefore, it generates faulty region proposals candidates.

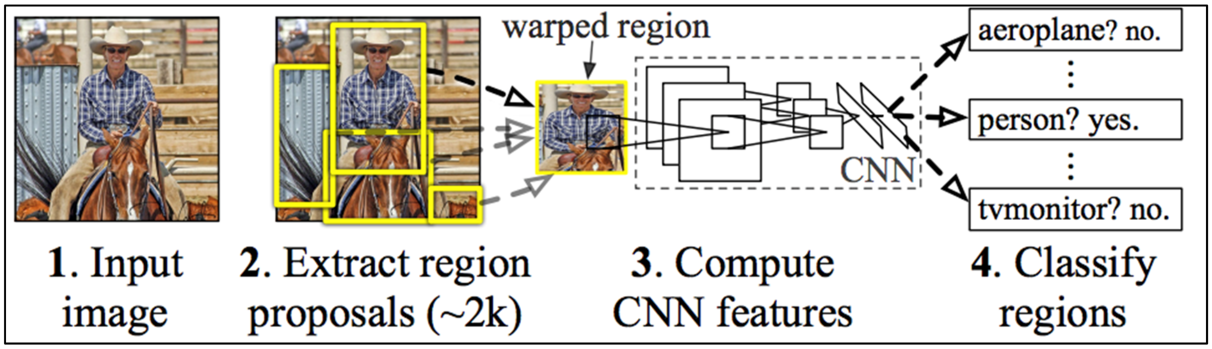


Figure 3.25 R-CNN model architecture
Taken from Girshick et al. (2014)

3.2.1.2 Fast R-CNN

Girshick et al. (2014), address the above issues in a later R-CNN release. They introduce enhancements to R-CNN to improve training and testing speed and increase the accuracy. The enhanced version of R-CNN is called Fast R-CNN (Girshick, 2015). The multi-stage pipeline training adopted in R-CNN was replaced by a single-stage multi-task loss where training can update the entire network. Fast R-CNN (Figure 3.26) takes the entire image as an input and feeds it directly to a CNN. The CNN generates a feature map for the entire image. The region of interest (RoI) is then identified from the generated feature map. Each RoI has four-tuple (r, c, h, w) that defines the RoI top-left corner (r, c) and its height and width (h, w). A RoI pooling layer is added after the generated feature map. The RoI pooling layer reshapes the RoI window to a fixed size, then it maps the reshaped RoI to fully connected layers (FCs) that return a RoI feature vector. The outputs of Fast R-CNN are the class of the proposed region and its bounding-box offsets. Fast R-CNN at test time achieves a speed of 0.32s per image (Girshick, 2015) ignoring the time spent on region proposals, since it does the convolution operation only once for each image instead of repeating this operation 2000 time per image like in R-CNN.

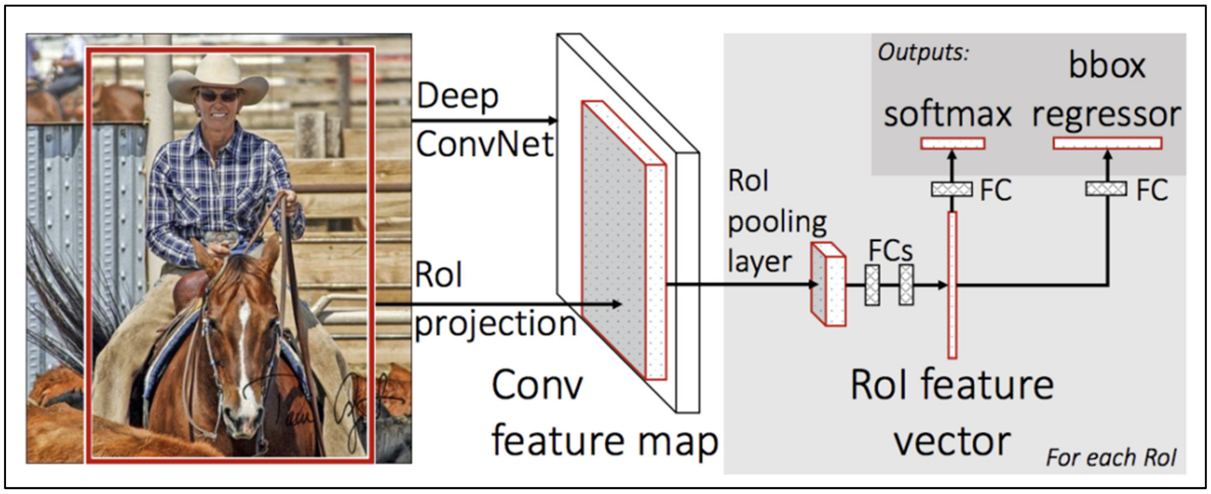


Figure 3.26 Fast R-CNN architecture
Taken from Girshick (2015)

3.2.1.3 Faster R-CNN

Faster R-CNN developed by Ren et al. (2015) introduces further enhancements to Fast R-CNN. The authors address the region proposal computational time issue. The previous method (Fast R-CNN) takes 0.32 seconds per image to detect the proposals; however, the proposal generation task takes 2 seconds. Thus, the detection with Fast R-CNN is bottlenecked with the regional proposal method. Therefore, Faster R-CNN paper proposes a new approach towards real-time object detection. The authors introduce a novel Region Proposal Network (RPN) that computes proposals' candidates with a DNN. By sharing convolutional layers between the

RPN and the detector network, the authors were able to reduce the time spent on generating proposals during the test.

Faster R-CNN is based on two modules: the RPN, a fully convolutional network, and the detector that uses the proposed RoI for object detection (Figure 3.27).

Region Proposal Network (RPN): A Region Proposal Network generates a set of proposals to the detector network, it takes an image as input and returns proposals. A proposal is a rectangular region defined by its center coordinates (x, y) and its width and height (w, h). Each proposal has a score called objectness score, it measures the probability of the object belonging to two sets of classes: background (i.e., non-object classes), and foreground (i.e., object classes).

A small network is added to the last shared convolutional layer. This network consists of a 3 x 3 sliding window that maps the feature map layer to a lower dimension. Furthermore, the authors add two parallel fully connected layers: the first one is a classifier, it generates the objectness score. The second layer is a regression layer, it generates the predicted proposals coordinates. The 3 x 3 sliding window on the feature map has a large receptive field, this means that the 3 x 3 window covers a large area on the input image, thus, it contains rich features. Objects in this area may have different scales, also they may have a different location inside the receptive field area. The authors address this issue by introducing the concept of anchors.

Anchors: The authors in Faster R-CNN paper (2015), define an anchor as a reference proposal positioned at the center of the sliding window (Ren et al., 2015). An anchor has different aspect ratios and different scales. The authors implemented 3 different aspect ratios (1:2, 1:1, 2:1) at 3 different scales (128^2, 256^2, 512^2). To illustrate that, a scale with a size 128^2 is set with 3 different aspect ratios: 128 x 256, 128 x 128, and 256 x 128. In total, each sliding window has k anchors; k is equal to the number of scales multiplied by the number of ratios (e.g., k is equal to 9 in the above case) – this method is called pyramid of anchors.

Each sliding window has 2k scores to classify each anchor as a foreground or a background object. Also, it has 4k predicted coordinates (x, y, w, h) to adjust the k anchors positions (xa, ya) and the dimensions (wa, ha) of the k anchors. An image with a feature map's size equals to W x H has a W x H x k anchors.

DNN for object detection: The detector network is the same as Fast R-CNN. It takes the features from the RPN, maps it to fully connected layers (FCs). An object classifier classifies the proposals according to their classes and a bounding-box regressor refines the bounding-boxes for each of the proposals. The bounding-box refinement is more accurate in the second stage since it is based on region proposals, i.e., the features fed to the fully connected layers belong to the proposed RoI and they are not limited to the sliding window (as is the case with the RPN), thus, more features are captured in the second stage.

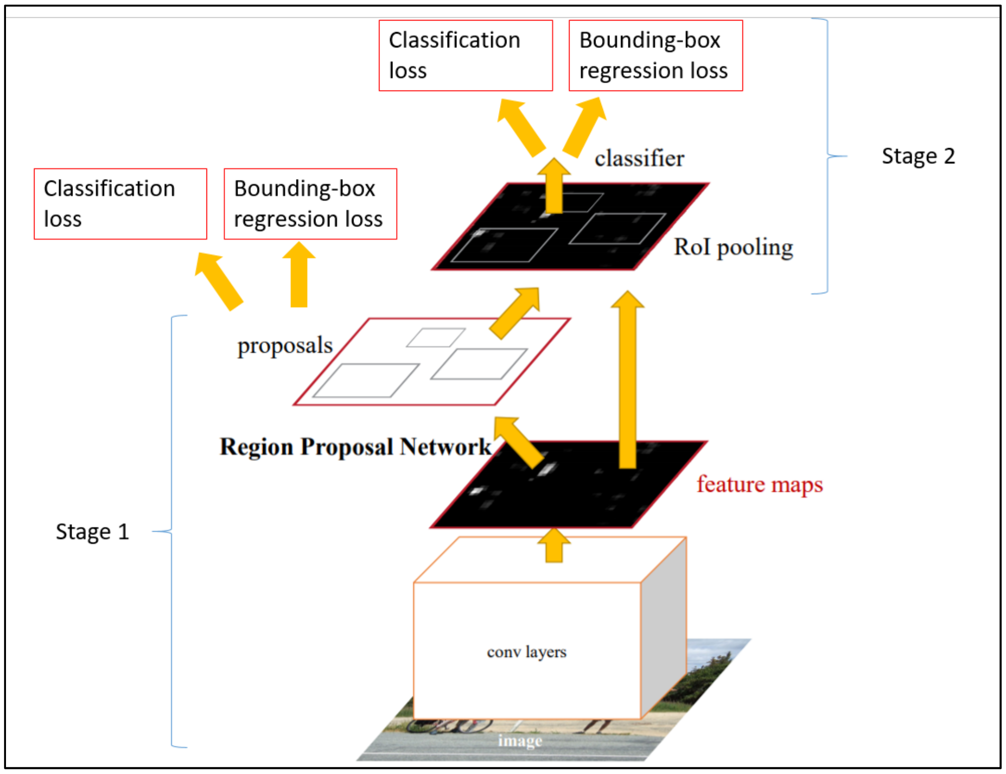


Figure 3.27 Faster R-CNN architecture
Adapted from Ren et al. (2015)

3.2.1.4 Mask R-CNN

Mask R-CNN (He et al., 2017), extends Faster R-CNN by adding a branch in parallel to the bounding-box classification branch. The new additional branch predicts the mask of a detected object. Mask R-CNN can be used for bounding-box object detection, instance segmentation, and person key-point detection. The developer of Mask R-CNN uses the Feature Pyramid Network (Lin et al., 2017), FPN to extract features.

FPN solves the problem of detecting small objects. Instead of using pyramid of images (i.e., same image with different scales), FPN replaces the feature extractor in Faster R-CNN. Figure 3.28 (Lin et al., 2017) shows pyramid architectures.

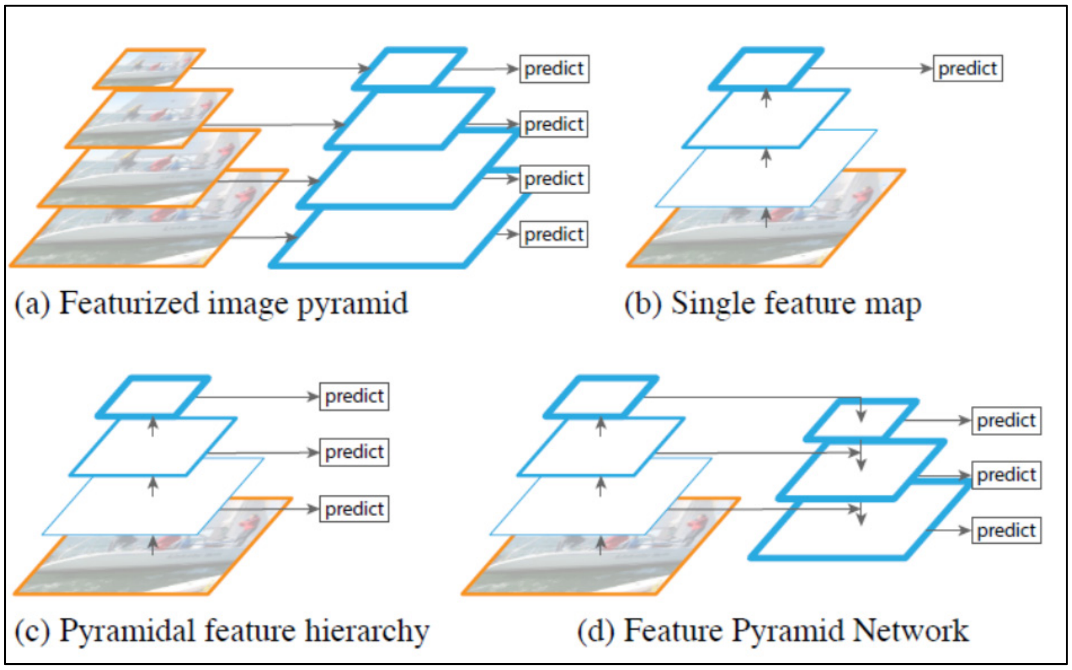


Figure 3.28 Prediction at different feature layers
Taken from Lin et al. (2017)

A featurized image pyramid Figure 3.28.a detects objects while changing the scale of the image. This method is slow as we are extracting feature maps from each scaled image then detecting objects. Figure 3.28.b is the simple ConvNet architecture, the detection occurs at the last feature map, this method is fast, though not accurate, and performs poorly on small objects, since upper feature maps have low spatial resolution than the low level features (Figure 3.46 – 3.49). The detection in Figure 3.28.c occurs at each level of the feature maps. Figure 3.28.d shows the architecture used in Mask R-CNN. This architecture maintains a high semantic feature maps along all scales. FPN connect the low-resolution strong semantic features with high-resolution semantically weak features in a top-down pathway with lateral connection (Lin et al., 2017) (Figure 3.30). By doing so, FPN preserves higher resolution layer with semantic rich layer. Figure 3.30 shows the FPN feature extraction process with ResNet-50 the DNN.

ResNet-50 is a Deep Neural Network developed by K. He et al. (2016). It is a 50 layers DNN. Before the introduction of residual blocks, DNN suffers from vanishing gradient (K. He et al., 2016), i.e., the DNN stop further training when the gradient is small; thus, the weights stop updating. To overcome this problem K. He et al. (2016) introduces the concept of residual blocks shown in Figure 3.29. This allows to pass the input x to the output with an identity function. The identity connection is called a skip connection with no weights to learn; thus, we preserve the gradient and allow it to reach earlier layers.

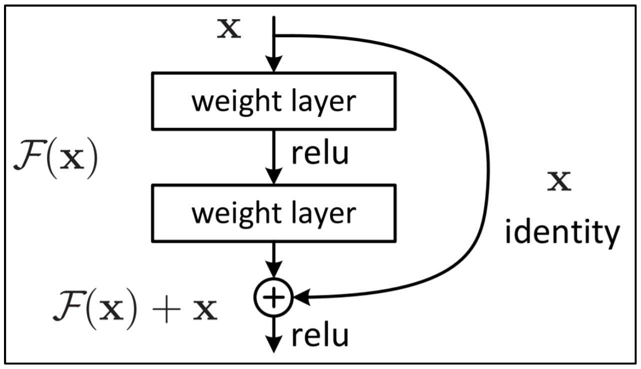

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3.29 Residual block
Taken from K. He et al. (2016)

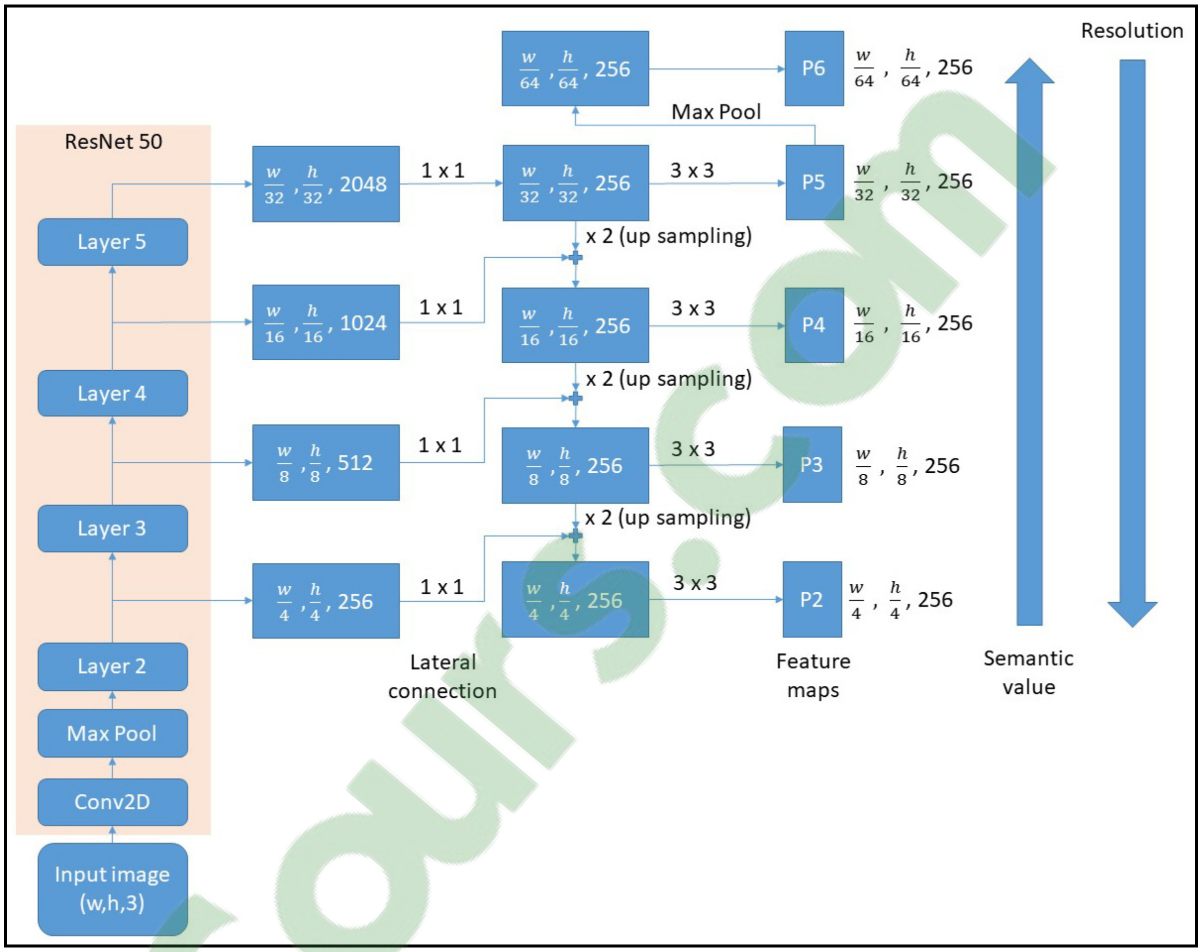

Resolution
ResNet 50
Layer 5
Layer 4
Layer 3
Layer 2
Max Pool
Conv2D
Input image (w,h,3)
$\frac{w}{64}, \frac{h}{64}, 256$
$\frac{w}{32}, \frac{h}{32}, 2048$
$\frac{w}{16}, \frac{h}{16}, 1024$
$\frac{w}{8}, \frac{h}{8}, 512$
$\frac{w}{4}, \frac{h}{4}, 256$
1 x 1
$\frac{w}{32}, \frac{h}{32}, 256$
$\frac{w}{16}, \frac{h}{16}, 256$
$\frac{w}{8}, \frac{h}{8}, 256$
$\frac{w}{4}, \frac{h}{4}, 256$
x 2 (up sampling)
3 x 3
Max Pool
P6
P5
P4
P3
P2
$\frac{w}{64}, \frac{h}{64}, 256$
$\frac{w}{32}, \frac{h}{32}, 256$
$\frac{w}{16}, \frac{h}{16}, 256$
$\frac{w}{8}, \frac{h}{8}, 256$
$\frac{w}{4}, \frac{h}{4}, 256$
Lateral connection
Feature maps
Semantic value

Figure 3.30 FPN diagram
Adapted from Fractal.ai (2019)

Figure 3.30 shows the generation of the feature maps within FPN. The input image is first processed by ResNet-50 (K. He et al., 2016). The output feature from layers 2, 3, 4, and 5 are connected laterally with a top-bottom pathway with a 1 x 1 convolutional operation. The output of the convolutional operation is added to the top-down pathway to generate a semantically rich feature maps (P2, P3, P4, P5, and P6).

The generated feature maps are then introduced into a Region Proposal Network (RPN). RPN takes the generated feature maps and convolute them with 3 x 3 filters followed by the rectified linear unit, ReLu, activation function (Equation 3.4) (Hahnloser et al., 2000). The output of the activation function is then introduced into two branches: a binary classifier to compute the objectness score, and a regressor layer to predict the bounding box coordinates. Figure 3.31 shows the diagram of the RPN network and equations 3.5, 3.6 and 3.7 show the equation of the loss function as implemented in (Abdulla, 2017).

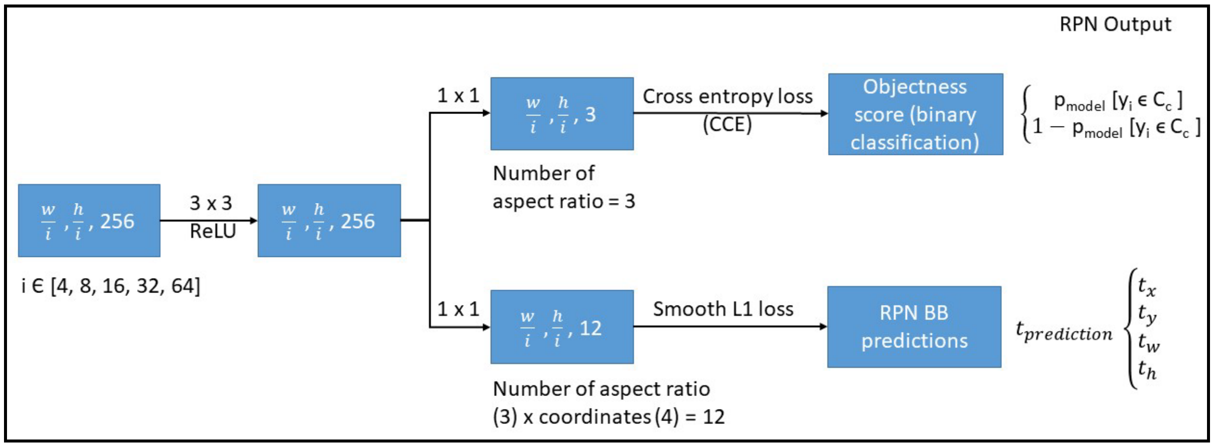

RPN Output
$\frac{w}{i}, \frac{h}{i}$, 256
3 x 3
ReLU
$\frac{w}{i}, \frac{h}{i}$, 256
i ∈ [4, 8, 16, 32, 64]
1 x 1
$\frac{w}{i}, \frac{h}{i}$, 3
Number of aspect ratio = 3
Cross entropy loss (CCE)
Objectness score (binary classification)
p_{model} [y_i ϵ C_c]
1 − p_{model} [y_i ϵ C_c]
1 x 1
$\frac{w}{i}, \frac{h}{i}$, 12
Number of aspect ratio (3) x coordinates (4) = 12
Smooth L1 loss
RPN BB predictions
$t_{prediction}$
t_x
t_y
t_w
t_h

Figure 3.31 RPN diagram showing the two classification and the bounding boxes branches
Adapted from Fractal.ai (2019)

Equation 3.5 is the Categorical Cross Entropy loss function, CCE, it computes the cross entropy loss between the true labels and the predictions (Chollet et al., 2015). N in Equation 3.5 is the number of samples in the batch (the batch size), i is the observation or one sample in the batch, C is the number of classes (in this case C is 2, background and foreground), and c is the category (i.e., foreground or background). 1 is the indicator function, i.e., it is equal to 1 if the ith observation belongs to the c category, 0 otherwise. p_{model} [y_i ϵ C_c] is the probability that the ith observation belongs to the c category. In RPN the CCE can be simplified to a Binary Cross Entropy, BCE, loss function (RPN classify objects as binary classes, i.e., foreground and background), Equation 3.6 (Chollet et al., 2015). $\hat{y}_i$ in Equation 3.6 is the probability of the ith observation belongs to one of the binary classes.

$$y = max(0, x) \tag{3.4}$$

$$CCE = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} 1_{y_i \epsilon C_c} * log(p_{model}[y_i \epsilon C_c]) \tag{3.5}$$

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}[y_i * \ log\widehat{y_i} + (1 - y_i) * log(1 - \widehat{y_i})] \tag{3.6}$$

Smooth L1 loss function adopted by (Abdulla, 2017) implementation is shown in Equation 3.7. x is the difference between the target and the prediction (i.e. $t_{target} - t_{predic}$). The bounding box target and prediction are 4 elements vectors defined in Equation 3.8 (Ren et al., 2015). In Equation 3.8 (He et al., 2017) t_{predic} is the predicted anchor-bounding box translation and t_{target} is the target anchor-ground truth translation. x, x_a, and x^* are the center coordinates of the predicted bounding box, the anchors and the ground truth bounding box respectively, same for y, w, and h (He et al., 2017).

$$L_{1\,;\,smooth} = \begin{cases} |x| \ if \ |x| > 1 \\ x^2 \ if \ |x| \leq 1 \end{cases} \tag{3.7}$$

$$t_{predic} = \begin{bmatrix} t_x = \dfrac{x - x_a}{w_a} \\ t_y = \dfrac{y - y_a}{h_a} \\ t_w = log\dfrac{w}{w_a} \\ t_h = log\dfrac{h}{h_a} \end{bmatrix}; \qquad t_{target} = \begin{bmatrix} t_x^* = \dfrac{x^* - x_a}{w_a} \\ t_y^* = \dfrac{y^* - y_a}{h_a} \\ t_w^* = log\dfrac{w^*}{w_a} \\ t_h^* = log\dfrac{h^*}{h_a} \end{bmatrix} \tag{3.8}$$

After generating the region of interest from the anchors with the RPN, we select the top 6000 regions according to their highest objectness score. Then, Non Maximum Suppression, NMS, algorithm (Algorithm 1) is applied to remove boxes overlapping with others, i.e. boxes with intersection over union, IoU, (Figure 3.32) larger than 70% (70% intersection over union). Algorithm 3.1 shows the NMS execution process.

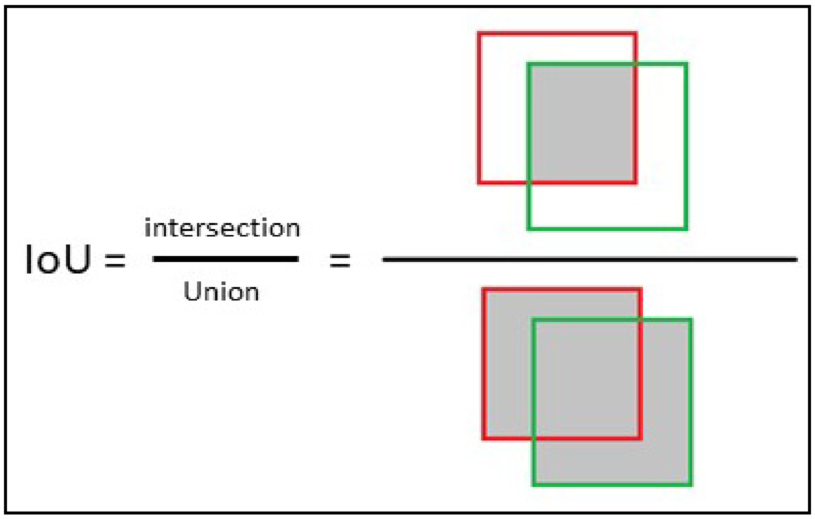


Figure 3.32 Intersection over Union

Algorithm 3.1 Non Maximum Suppression
Adapted from Bodla et al. (2017)

```
Non Maximum Suppression - adapted from (Bodla et al., 2017)

Input:       $B = \{b_1, .., b_K\}$, $S = \{s_1, .., s_K\}$, t
             B is the list of initial detection boxes
             S is the list containing corresponding detection scores
             t is the NMS threshold
Output:      D is the final list containing the post-NMS top N ROI

1  D = {}                       initialize D as an empty list
3  while B is not empty do      iterate in the initial detection boxes B list
4       m = argmax S            get the index of the largest detecting score in the S list
5       M = $b_m$               M is the box with the highest score
6       D = D U M               put M in the list of the final detection
7       B = B - M               remove M from the B list
8       for $b_i$ in B do       iterate inside the B list
9          if IoU (M, $b_i$) > t  compare the IoU between M and the rest of elements inside B
10             B = B - $b_i$    if the element $b_i$ is larger than t the threshold remove $b_i$ from B
11             S = S - $s_i$    and remove $s_i$ from S
12        end                   end the iteration once all the elements in B are compared to M
13    end                       end the iteration once B is empty
14    return D, S               return final detection list D, and its corresponding score list S
15  end                         end the NMS algorithm
```

The final proposals, or the NMS outputs, are then introduced to the box head which classify the proposals according to their classes and adjust their coordinates. Figure 3.33 shows the box head diagram.

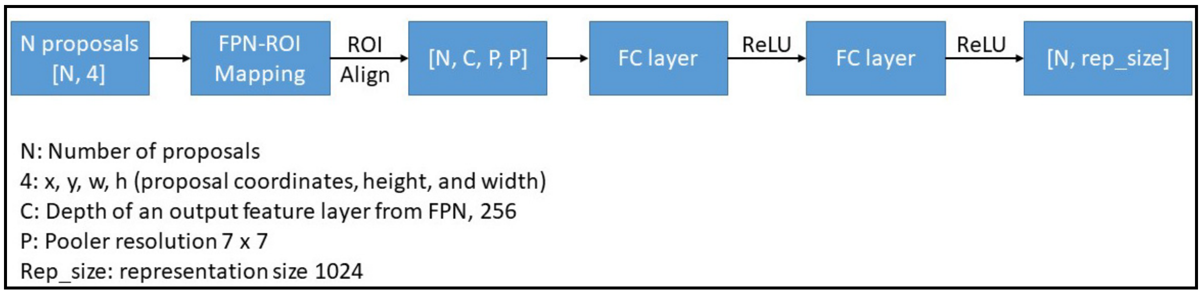

N proposals [N, 4]
FPN-ROI Mapping
ROI Align
[N, C, P, P]
FC layer
ReLU
FC layer
ReLU
[N, rep_size]
N: Number of proposals
4: x, y, w, h (proposal coordinates, height, and width)
C: Depth of an output feature layer from FPN, 256
P: Pooler resolution 7 x 7
Rep_size: representation size 1024

Figure 3.33 Box head feature extraction
Adapted from Fractal.ai (2019)

The N proposal generated by the RPN-NMS are mapped into a specific feature map generated by the FPN according to their width w, and height h. The Equation 3.9 extracted from (Lin et al., 2017) shows that the feature-level map is chosen according to the width (w) and height (h) of the RoI. 224 x 224 is the size of the images from the ImageNet dataset where ResNet is pre-trained on. K_0 is equal to 4, which means a RoI with size equal to 224 x 224 uses the fourth convolutional level output feature map to classify the detected object (Lin et al., 2017). A smaller RoI will result a low level feature map which is a higher resolution low semantic feature map (Figure 3.30). This justifies why FPN solves the problem of detecting small objects, which is suitable for crack detection application.

$$K = K_0 + \ log_2\left(\frac{\sqrt{w * h}}{\sqrt{224 * 224}}\right) \tag{3.9}$$

The RoI pooling layer defined in Fast R-CNN and used in Faster R-CNN leads to a slightly misalignment between the RoI on the features map and the original image, thus, a pixel wise detection (i.e., instance segmentation) performed directly on the RoI pooling layer is inaccurate. To solve this issue, the authors introduce RoIAlign, a method to adjust the RoIPool. This method is based on bilinear interpolation (Jaderberg et al., 2015). RoIAlign works by first dividing the RoI into bins, in our case we divide the RoI into 7 x 7 grid (the size of the pooling layer or the pooler resolution). From each bin we sample 4 points, according to (He et al., 2017) the number and the position of the sampling points don’t affect the final results. In contrast to Mask R-CNN facebook implementation (Girshick et al., 2018), Matterport implementation (Abdulla, 2017) uses one point inside each bin instead of four points.

To better understand the RoIAlign method, we visualize the approach of pooling and aligning the ROI with the feature map.

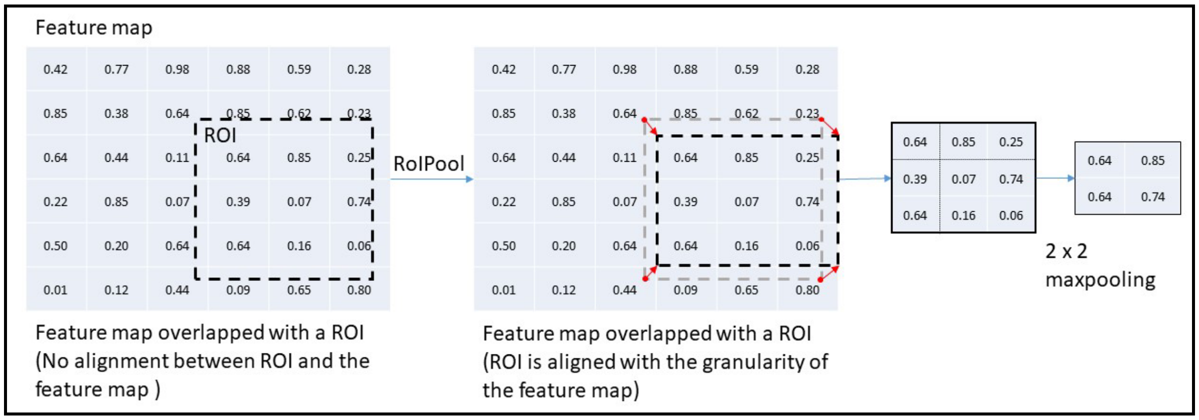

Feature map
ROI
RoIPool
Feature map overlapped with a ROI (No alignment between ROI and the feature map)
Feature map overlapped with a ROI (ROI is aligned with the granularity of the feature map)
2 x 2 maxpooling

Figure 3.34 2 x 2 ROI max pooling

In Figure 3.34 we show how the ROI pooling works. The misalignments between the feature map and the ROI is due to: (1) the rescaling while generating the feature maps from the input image, and (2) the ROI coordinates are with respect to the input image, i.e., the change in scale between the input image and its feature map leads to this misalignment. The pooling layer ignores the exact dimensions of the ROI and round its boundaries to match the feature map granularity. Although RoIPool works on object detection, it leads to faulty results on the pixel wise detection.

Figure 3.35 shows the solution presented in (He et al., 2017). Instead of using the RoIPool, He et al. use the RoIAlign with bilinear interpolation.

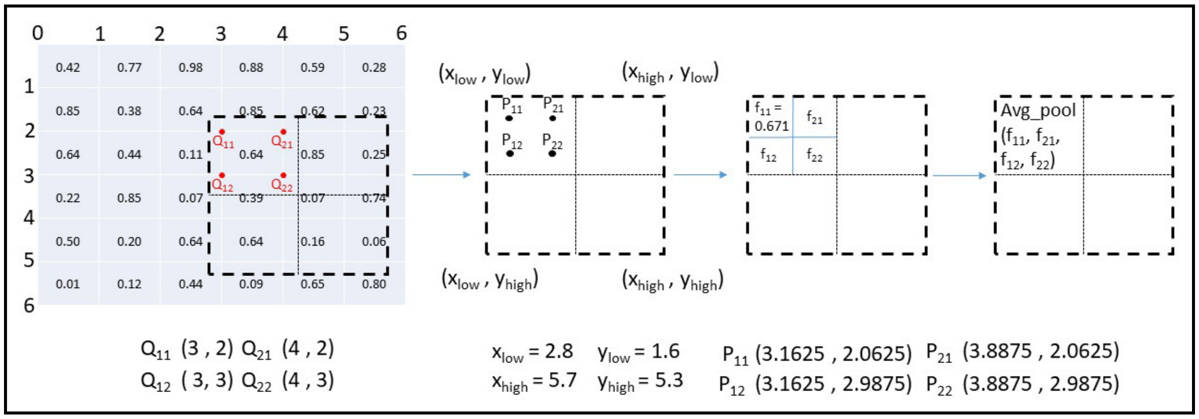


Figure 3.35 Bilinear interpolation in RoIAlign

RoIAlign in Figure 3.35 maintain the coordinate of the proposal's bounding box (x_{low} , y_{low} , x_{high} , y_{high}). We split the RoI into four bins, and in each bin we sample four points (P_1, P_2, P_3, P_4). The coordinates of the samples are computed according to Equation 3.10 extracted from (Abdulla, 2017).

$$x = x_{low} + ((i + 0.5) * \frac{x_{high} - x_{low}}{num\_samples} \tag{3.10}$$

$$i = [0, 1]$$

$$y = y_{low} + ((j + 0.5) * \frac{y_{high} - y_{low}}{num\_samples}$$

$$j = [0, 1]$$

$$num\_samples = 4$$

To compute the value corresponding to each sampled point, we apply a bilinear interpolation following Equation 3.11. (x_1, x_2, y_1, y_2) in Equation 3.11 are the coordinates of the cells surrounding the sample point (in our example they are the coordinates of the Q points, Figure 3.35). (x_i, y_j) are the $P_{i,j}$ coordinates i.e., the sampled point. $f_{i,j}(x_i, y_j)$ is the interpolated value at Pi,j point, and $f(x_i, y_j)$ is the original value at Qij point.

$$f_{i,j}(x_i, y_j) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 - x_i & x_i - x_1 \end{bmatrix} \begin{bmatrix} f(x_1, y_1) & f(x_1, y_2) \\ f(x_2, y_1) & f(x_2, y_2) \end{bmatrix} \begin{bmatrix} y_2 - y_j \\ y_j - y_1 \end{bmatrix} \quad (3.11)$$

Following the bilinear interpolation we average pool or max pool the interpolated values from the RoI. The average pool and the max pool equations are shown in equations 3.12 and 3.13.

$$maxpool = max\,\{f_{11}, f_{12}, f_{21}, f_{22}\} \quad (3.12)$$

$$averagepool = \frac{f_{11} + f_{12} + f_{21} + f_{22}}{4} \quad (3.13)$$

The output of the RoIAlign is reshaped and introduced to two fully connected layers, fc layer. Figure 3.33 shows the output of the reshaped RoIAlign connected to two consecutive fc layer to extract 1024 feature. The 1024 feature vector is then passed into a classifier to predict the class and a regressor to adjust the bounding box coordinates (Figure 3.36).

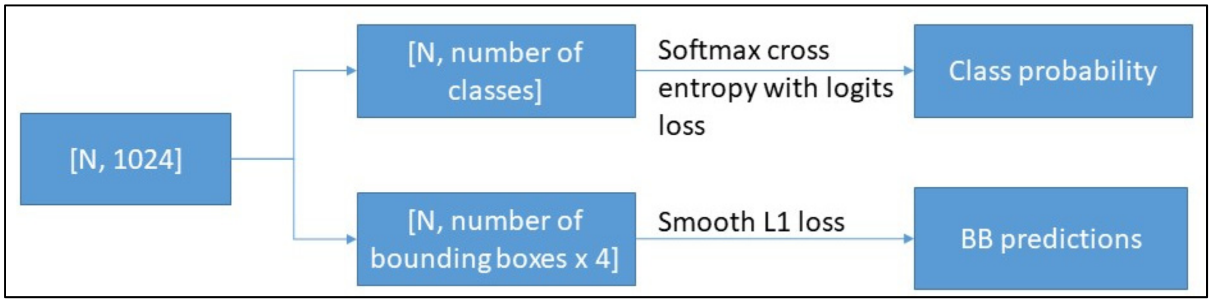


Figure 3.36 The classification head with the loss functions

The final detected bounding boxes are sent to the mask head, the final stage of detection. The masks are extracted by pixel wise detection from the convolutional features. Unlike the prediction of classes or the bounding box coordinates where the feature map is warped to a vector (1024 vector), the mask head preserves the spatial structure of masks by using a Fully Convolutional Network (FCN) (Long et al., 2015). FCN uses a per-pixel multinomial logistic loss, in our case it's a binomial logistic loss (one class for each detection), Equation 3.4. Figure 3.37 shows Mask R-CNN block diagram, it shows the data flow between its different stages.

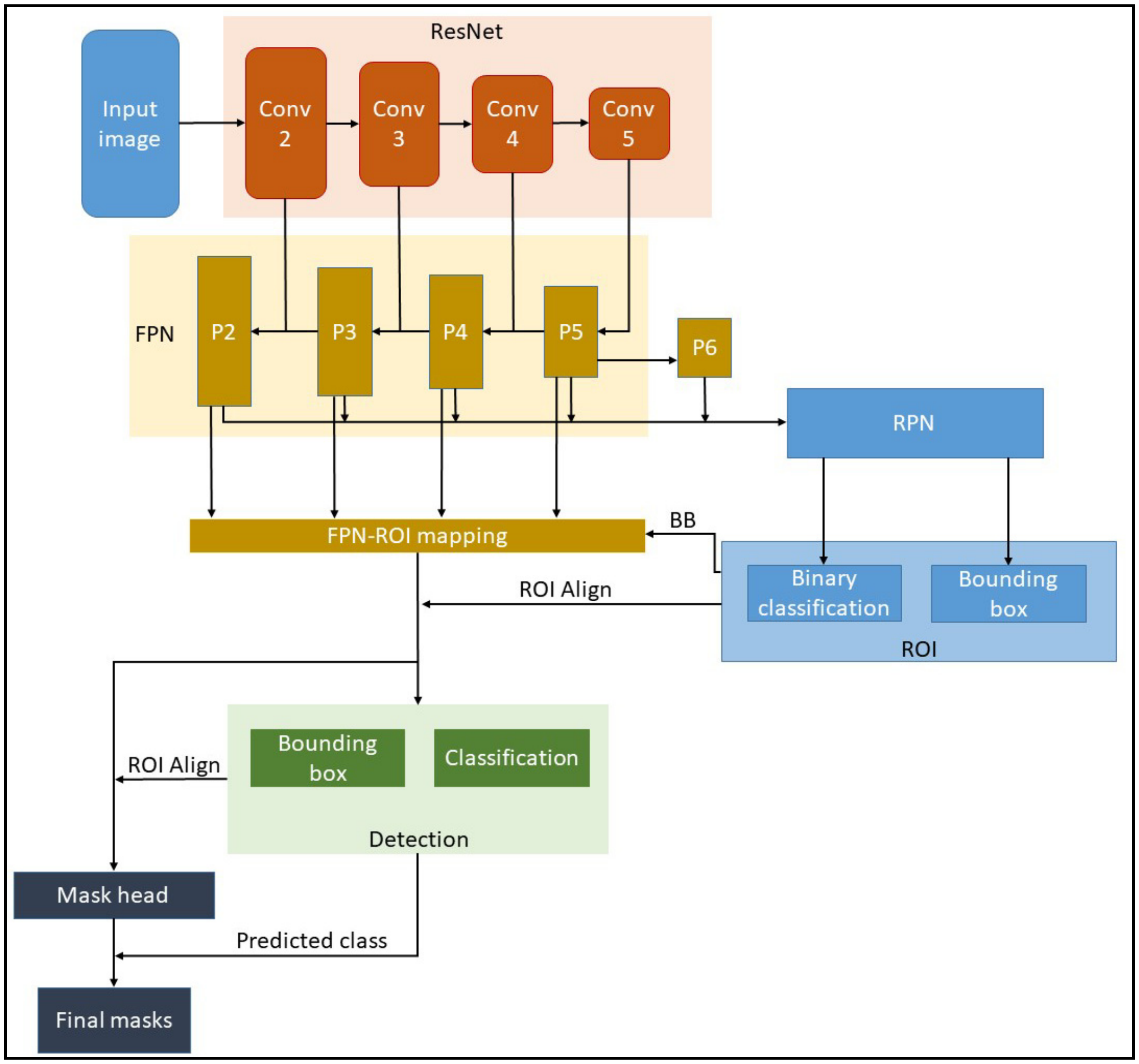


Figure 3.37 Mask R-CNN block diagram

3.2.2 Mask R-CNN in Detecting Cracks: Experiments and Results

The conventional pipeline used for supervised DNN is adopted to evaluate Mask R-CNN in detecting cracks. The pipeline, Figure 3.38 consists of collecting and annotating the dataset, then, training and evaluating the DNN.

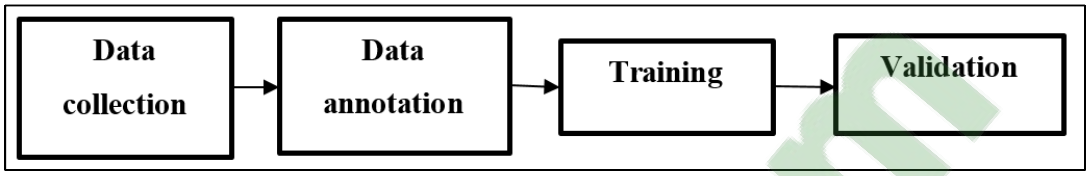


Figure 3.38 The conventional pipeline used for supervised machine learning algorithms

We create an original way for annotating our dataset to address the common problems in conventional computer vision-based crack detection methods (process detailed in the following section). The trained model can therefore detect connected cracks and cracks with different patterns. Moreover, we address the problem of detecting cracks from noisy images with different levels of luminosity by including different scenarios to the training dataset, such as images with shadows and surrounded by random objects. We validate our trained model on the validation dataset. We split the validation dataset to four subsets according to the level of noises and crack shapes in each image. We compute the pixel wise segmentation precision and recall for the entire dataset.

Data Annotation: We manually extract frames with cracks from the recorded video, and then annotate them following the common objects in context (coco) dataset format (Lin et al., 2014) Drawing crack's masks in accurate fashion is an important task: it allows a precise detection where the segmentation of the detected crack precisely follows its edge. By doing so, we ensure that all the detected pixels are relevant to the crack. This leads to a reliable segmentation and allows to precisely measure the width of the crack. We develop an annotation tool to draw high definition masks on top of the cracks. This tool is based on photo editing applications such as GIMP (GNU Image Manipulation Program (GIMP, 2019)). We follow an original way in the annotation to make the training easier and the detection reliable. Since cracks have a large variation in shapes and patterns, taking the whole crack as a single object may be an inefficient way to train an object detector such as Mask R-CNN. Therefore, we split each crack into different fractions, i.e. the union of different small cracks creates a full crack. Figure 3.39

shows an annotation example from the dataset. This annotation approach has several benefits. First, it allows a unified crack pattern along all the dataset; thus, the shape of the crack is no longer a problem for the detection. Second, a fraction occupies a larger space in the bounding box that surrounds it and, as a result, more relevant crack features are learned. Third, the number of cracks in each image increases up to 50 times (depending on the shape and the length of the crack), thus, the number of objects that Mask R-CNN takes to train is larger.

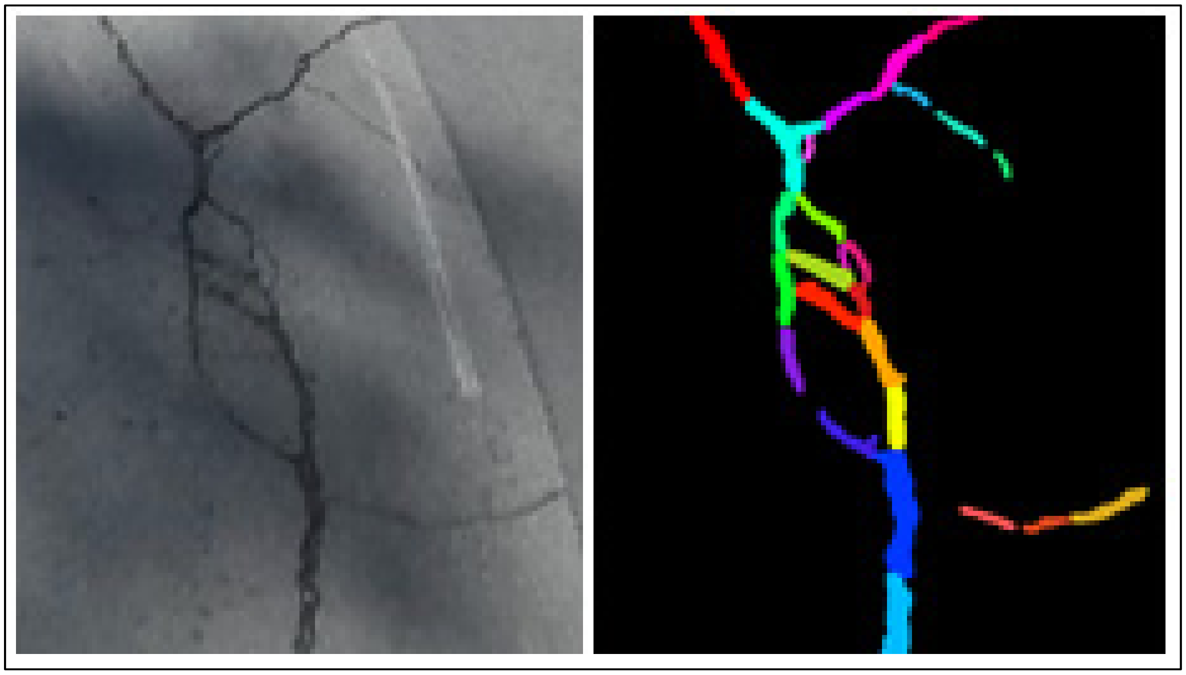


Figure 3.39 RGB image (left) and its corresponding ground truth (right). Different colours refer to different fractions of the crack, each colour in the GT image is considered as a separate object

We include different scenarios in the annotation process such as cracks in different luminosity levels, surrounded by random objects, and located on different types of pavement material. Figure 3.40 shows a set of different scenarios in the dataset. The total number of annotated images is 252, with 5204 fractions.

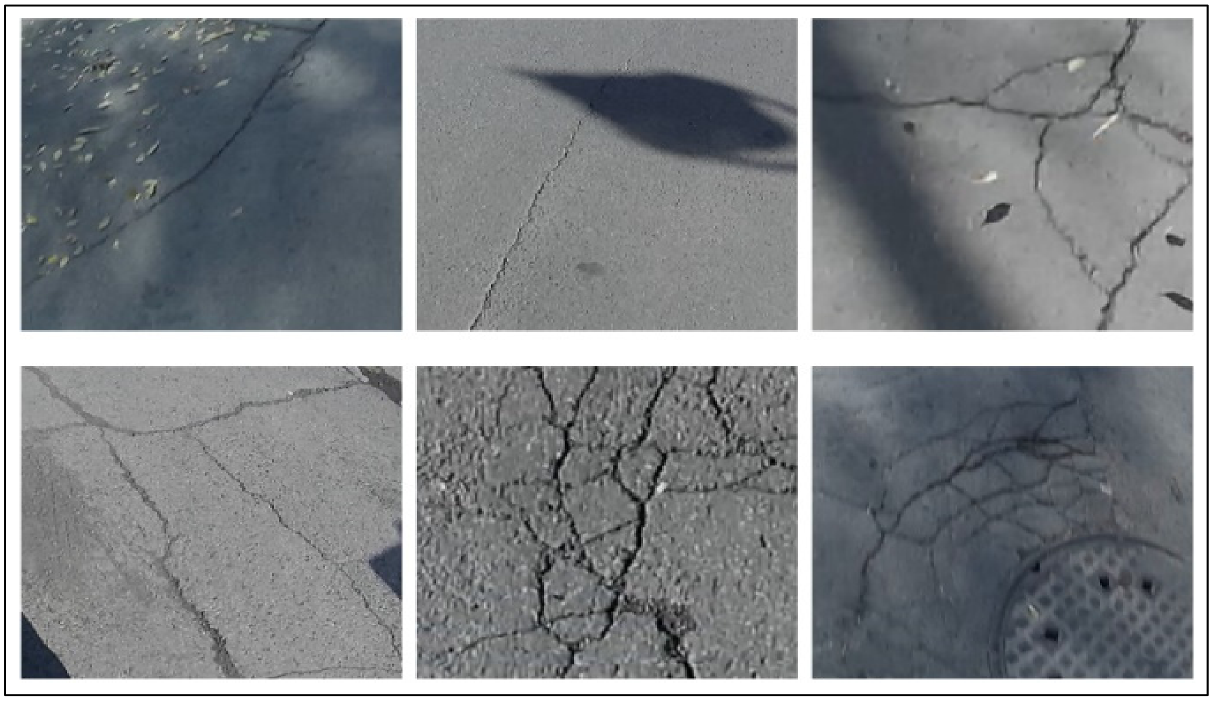


Figure 3.40 Cracks in different lighting conditions, with random objects and different pavement materials

3.2.2.1 Training and Validating the Results on Mask R-CNN

Mask R-CNN is one of different network architecture for object detection and instance segmentation. It belongs to a family of region-based CNN (Girshick, 2015; Girshick et al., 2014; Ren et al., 2015). It is built on top of Faster R-CNN (Ren et al., 2015) (i.e., for object detection), it extends Faster R-CNN by adding a branch in parallel to the bounding-box classification branch. The new additional branch predicts the mask on a detected object. In our application, Mask R-CNN is used for object detection and instance segmentation. The architecture of Mask R-CNN is based on a region proposal network RPN, Fast R-CNN (Girshick, 2015), and the mask generation head. Figure 3.41 illustrates a simple architecture of Mask R-CNN.

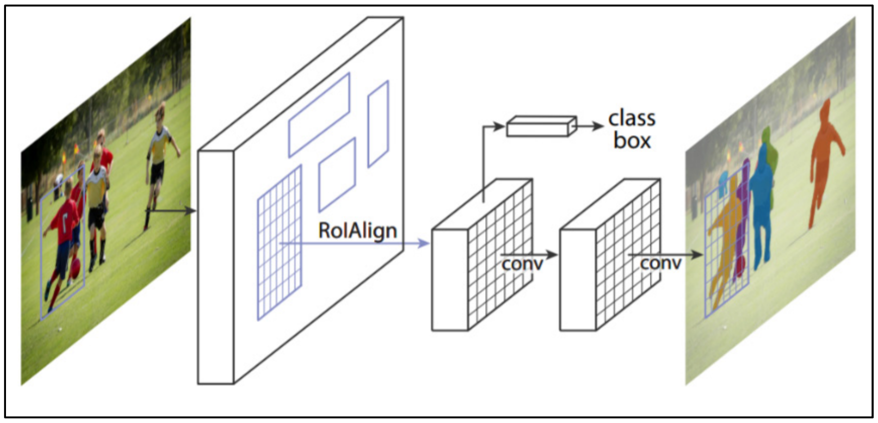


Figure 3.41 Architecture of Mask R-CNN
Taken from He et al. (2017)

The RPN takes an image as input and generates a set of proposals. A proposal is defined by its center coordinates (x, y) and its width and height (w, h). Similar to Faster R-CNN, each proposal has a score called "objectness score" that measures the probability of the object belonging to two sets of classes: background (non-crack class), and foreground (crack class). After generating the region proposal, a classifier categorizes the detected objects. A regressor adjusts their coordinates, and the mask head generates a pixel-wise segmentation for the detected object.

The same object in an image may have different scales and location inside the receptive field area. The authors of Mask R-CNN address this challenge by introducing the concept of anchors, a reference proposal positioned at the center of the sliding window on the feature map (He et al., 2017). An anchor has different aspect ratios and different scales. Carefully choosing the size of anchors may affect the training and the detection results. Large anchors cause missed detection (false negative) and small anchors cause faulty detection (false positive).

What makes Mask R-CNN a good choice for crack detection is that it integrates the feature pyramid hierarchical technique that preserves low level features such as edges. Feature Pyramid Network (FPN) (Lin et al., 2017) is an important component in detecting objects at different scales as it allows a good feature representation for each convolutional level.

Moreover, the adopted approach in annotation allows the proposal network to generate mini bounding boxes around a crack. The mini proposals significantly reduce the amount of background pixels compared to crack pixels allowing an accurate segmentation. Figure 3.42 shows a set of mini proposals covering a crack.

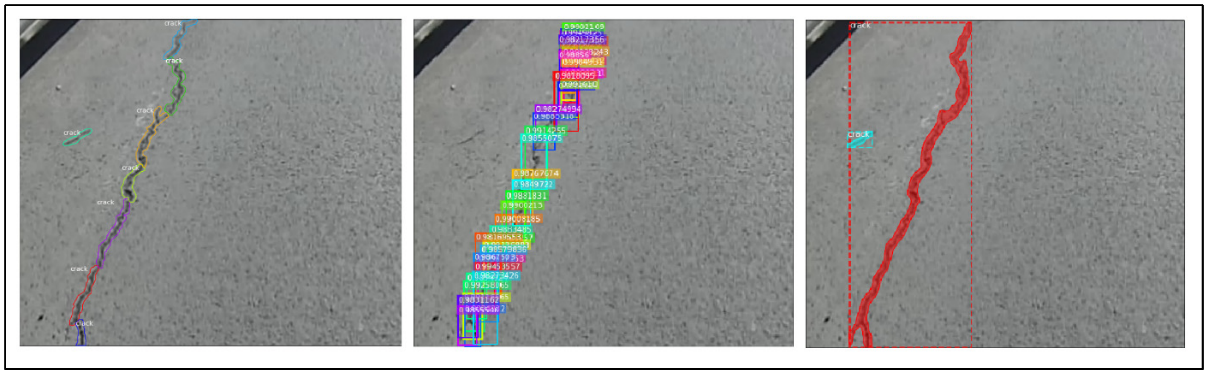


Figure 3.42 From left to right: RGB image with the GT; the top 40 mini proposals generated by ROI network; and the segmentation

Training: We adopt the Mask R-CNN Matterport implementation (Abdulla, 2017). We split our dataset to training (182 images, 3977 cracks' fractions) and validation (70 images, 1227 crack's fractions) subsets. The original images extracted from the videos have a 1920 x 1080 pixels resolution. We crop them to get multiple small images. This will reduce the GPU memory allocation while training and will create more background objects in each image. The network extracts from each image a subset of region proposals. Splitting one image to multiple images increases the amount of background objects. Thus, the network is more reliable in detecting non-crack objects. We carefully choose the configuration of Mask R-CNN. Cracks are usually thin and long objects, and therefore, we choose anchors that fit them. We adopt five different sizes for the anchor: 8, 16, 32, 64, and 128 pixels; and three aspect ratios: 1:2, 1:1, and 2:1. We use ResNet-50 (K. He et al., 2016) pre-trained on coco dataset, as the DNN backbone for feature extraction. We train only the heads' parameters of the network (i.e. the regressors, the classifiers and the mask head), this method is known by transfer learning.

While training, we add modifications to the images using image augmentation methods (Jung et al., 2020). We apply rotation, changing in the scale, contrast normalization, and we change the intensity of the images. By doing so, the number of the training examples is artificially increased and therefore, the trained model is more capable to generalize. We stop the training when the validation Mask R-CNN mask loss starts increasing, i.e. at 350 epochs.

In inference mode, the original model returns fractions of the whole crack. These fractions tend to overlap. Therefore, we add a function to Mask R-CNN that merges all the detected fractions into one object. The merge function takes the union of the overlapped cracks and returns a single object. Figure 3.43 shows an example of the segmentation before and after the merge function is applied.

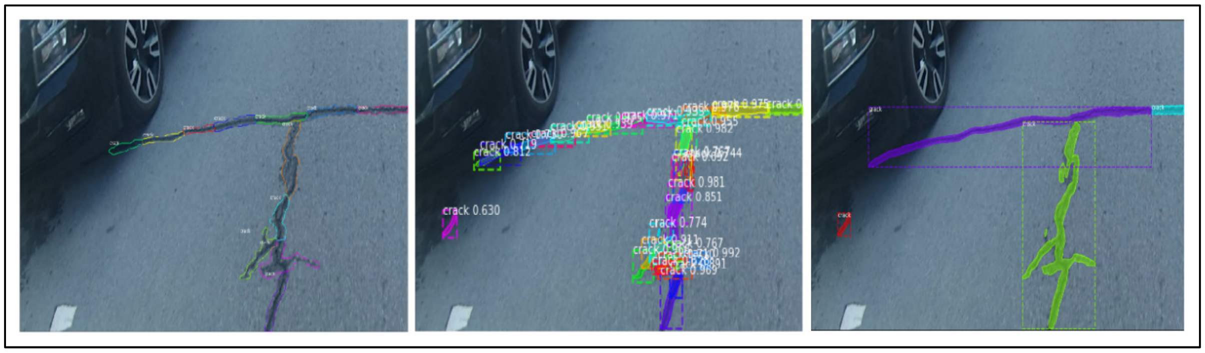


Figure 3.43 From left to right: RGB image with the GT; crack segmentation before applying the merge function; and after applying the merge function

To better understand the different steps of Mask R-CNN detection and segmentation, we show in the following the 3 main stages of the detection. In stage 1 the RPN runs a binary classifier on anchors that covers the image, and classify those anchors as background and foreground objects, moreover RPN also refine the coordinates of the proposed bounding box to better fit the detected object, Figure 3.44 shows the top 150 final proposals with their confidence scores.

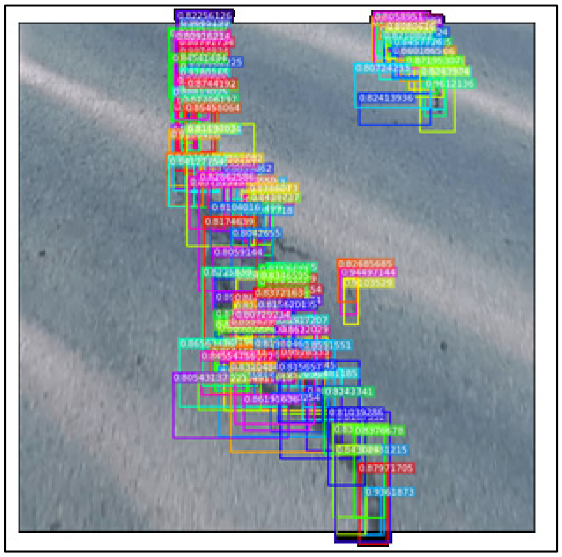


Figure 3.44 Top 150 proposals with their confidence scores

In stage two (Figure 3.45), the final proposals are classified according to their class and their bounding boxes are refined for a second time.

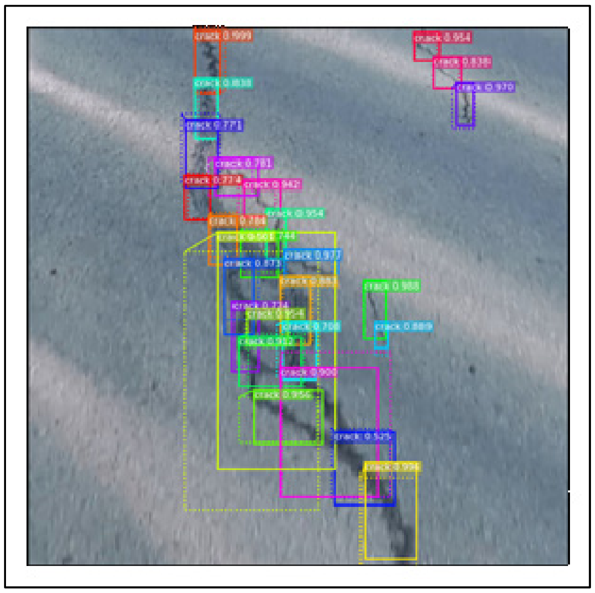


Figure 3.45 Final detection, before refinement (dotted lines), after refinement (solid lines)

Stage three takes the detection from stage two and runs a pixel wise detection to generate masks for each instance and then merge all the overlapped masks (Figure 3.46).

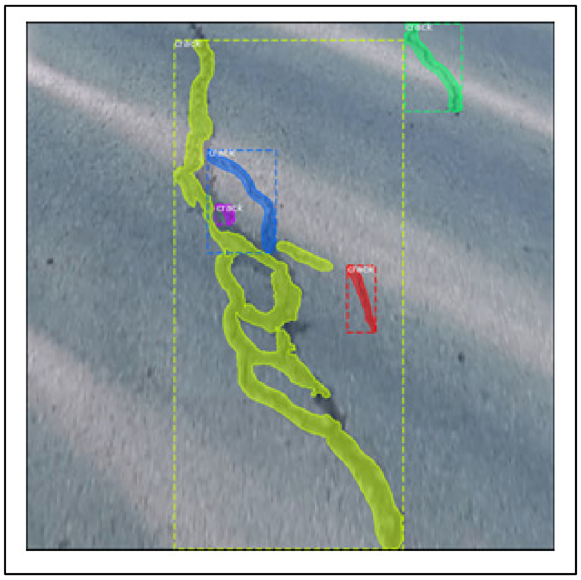


Figure 3.46 The final mask detection

Feature maps visualization: We use ResNet-50 (K. He et al., 2016) as the DNN backbone in Mask R-CNN. ResNet-50 is known for its high performance in object classification. ResNet-50 has five convolutional stage C1, C2, C3, C4, and C5. In figures 3.47 to 3.51 we show the feature maps generated by the ResNet-50 network at the second, third, fourth, and fifth convolutional stage for the same example shown in Figure 3.44. We can notice the details and the resolution in earlier convolutional stage (C2 and C3) in figures 3.46 and 3.48. However, the details fade in C4 and C5 feature maps. All the pixel wise detection occurs on the earlier stages. This is mainly induced by the size of the detected bounding box. Taking a crack as a whole object will force the network to choose a later feature maps (Equation 3.9) which are low resolution and crack features are blended with the semantic features. However, the low resolution layers have an important role in extracting semantic features such as background features.

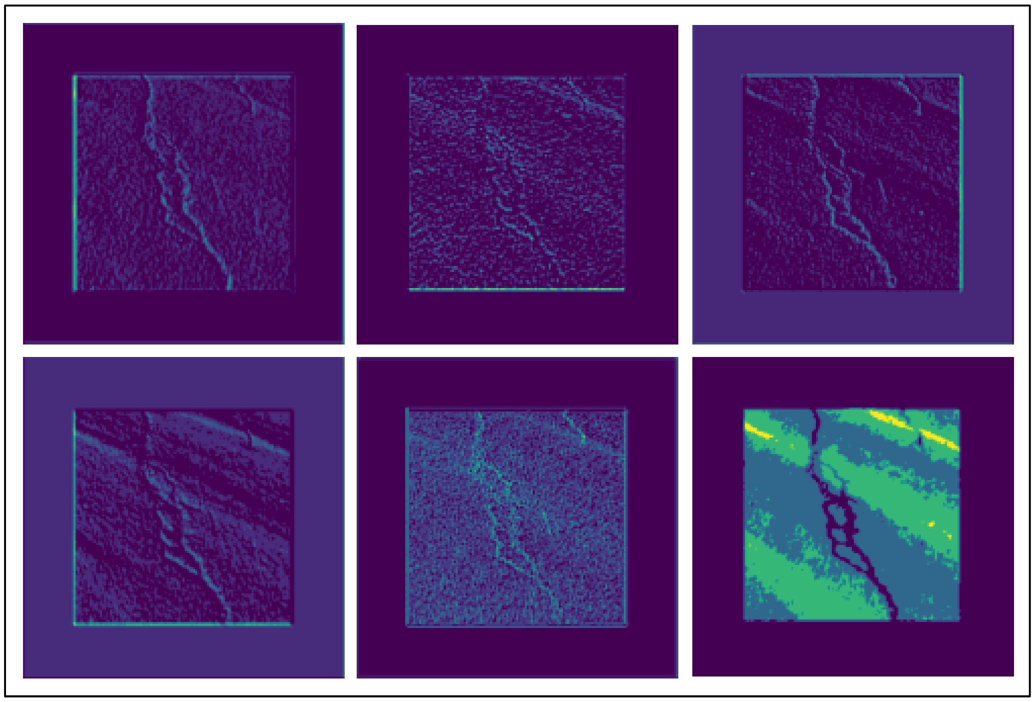


Figure 3.47 A selection of the C2 feature maps output

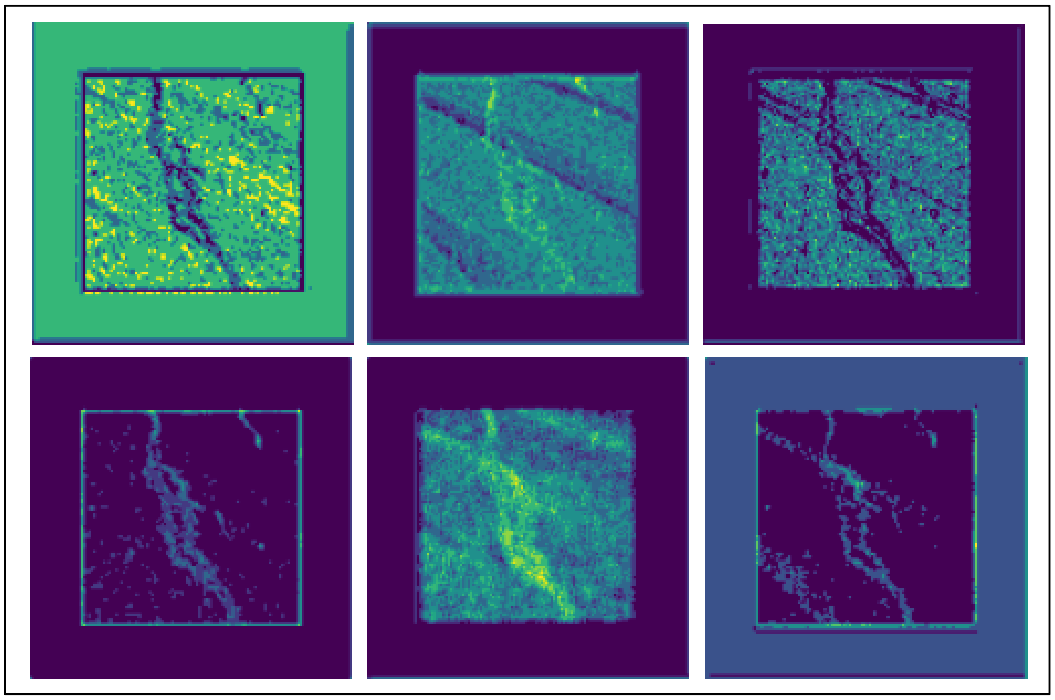


Figure 3.48 A selection of the C3 feature maps output

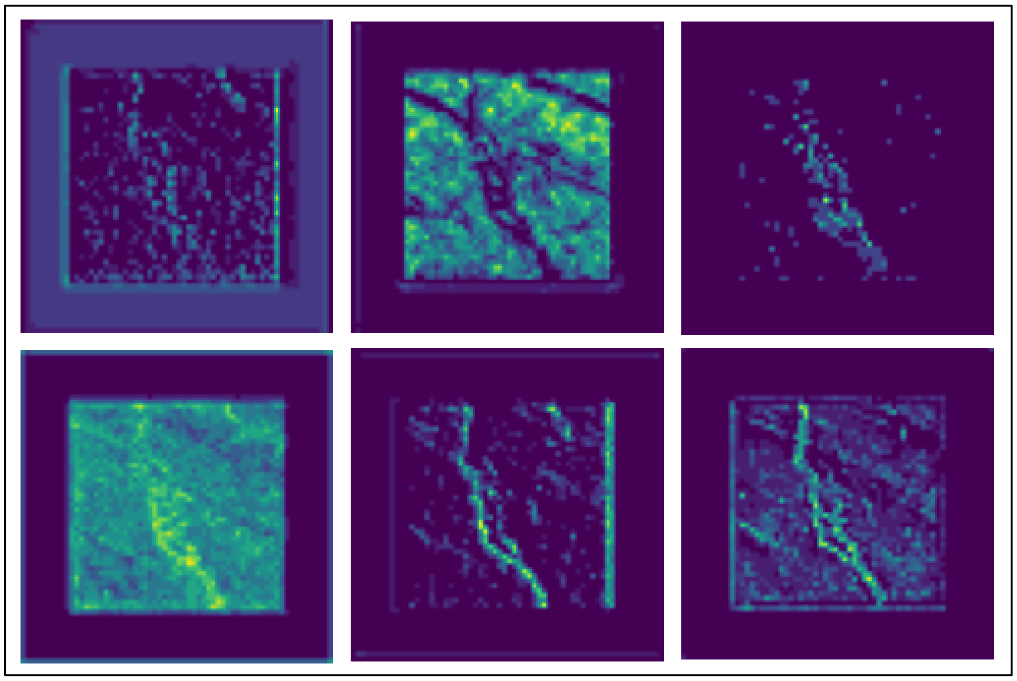


Figure 3.49 A selection of the C4 feature maps output

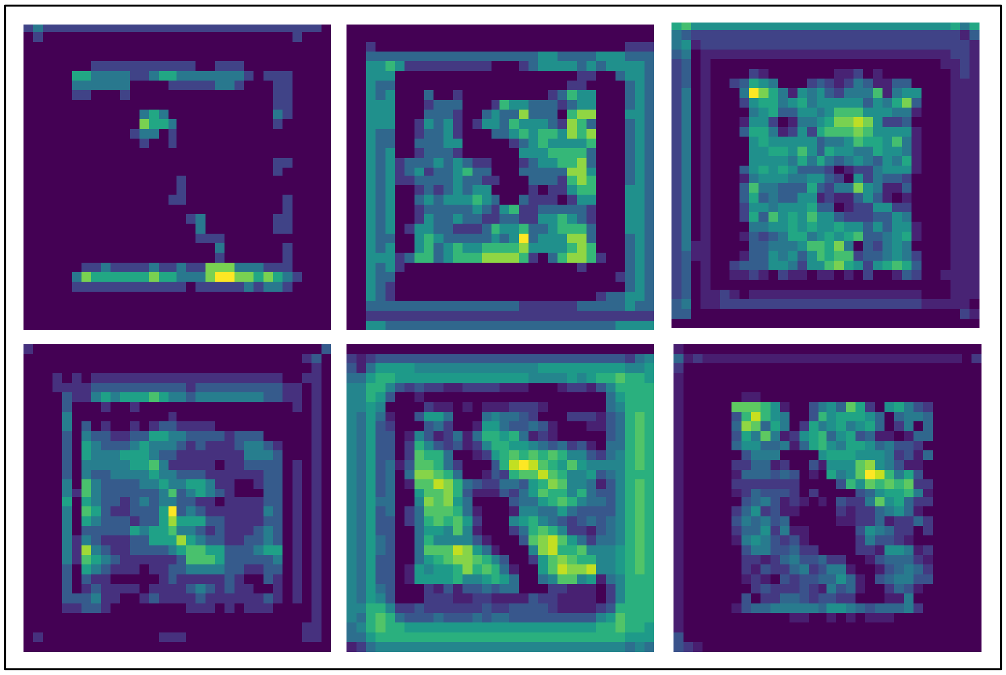


Figure 3.50 A selection of the C5 feature maps output

The bright pixels in figures 3.47 to 3.50 are the activation pixels, we can notice that once the pixels that belong to a crack are activated, the background pixels are deactivated and vice versa. In earlier stages, C2 and C3, (figures 3.47 and 3.48) the horizontal, vertical, diagonal edges of the crack are detected, as well as the brightness of the pavement. In later convolutional stages (figures 3.49 and 3.50) more semantic features are detected such as the pavement as an entire object.

Validation: To evaluate the performance of the trained model, we compute its precision, recall, and the average intersection over union (AIU), following (3.14), (3.15), and (3.17) equations respectively. True positives (TP) are the number of pixels detected by the model and match the Ground Truth (GT). False positives (FP) are the number of pixels detected by the model and different from the GT. False negative (FN) are the pixels that belong to the ground truth but not detected. Results show that the model is able to detect cracks with 77.67% precision and 79.19% recall (78.42% F1 score, Equation 3.16) and with 64.6% AIU.

$$Precision = \frac{\sum_i TP_i}{\sum_i (TP_i + FP_i)} \tag{3.14}$$
$$i = image\ id \in [\,0;\ 70]$$

$$Recall = \frac{\sum_i TP_i}{\sum_i GT_i} \tag{3.15}$$
$$i = image\ id \in [\,0;\ 70]$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.16}$$

$$AIU = \frac{\sum_i TP_i}{\sum_i (TP_i + FP_i + \ FN_i)} \tag{3.17}$$

The trained model shows its capability to detect cracks with different severity levels, shapes, and types. It also shows its capacity to differentiate between cracks and background noises within different and high contrast luminosity levels. The trained model returns better results on longitudinal, transversal, diagonal cracks, and cracks with ramifications. However, while the trained model can perfectly detect cracks with complex shapes such as alligator cracks, it has a low recall of 61.6% while segmenting complex shapes such as alligator cracks Figure 3.46. This drop can be explained by the following reasons: (1) the constraint set on the detection threshold for the entire validation dataset (e.g, some of the alligator crack fractions have a confidence score lower than the threshold set for the detection); and (2) the lack of alligator cracks in the training dataset. Figure 3.51, shows an example of the detection of an alligator crack. Results show that some of the detected fractions have a low confidence score (e.g., 0.5); however, the detected bounding boxes cover the entire crack.

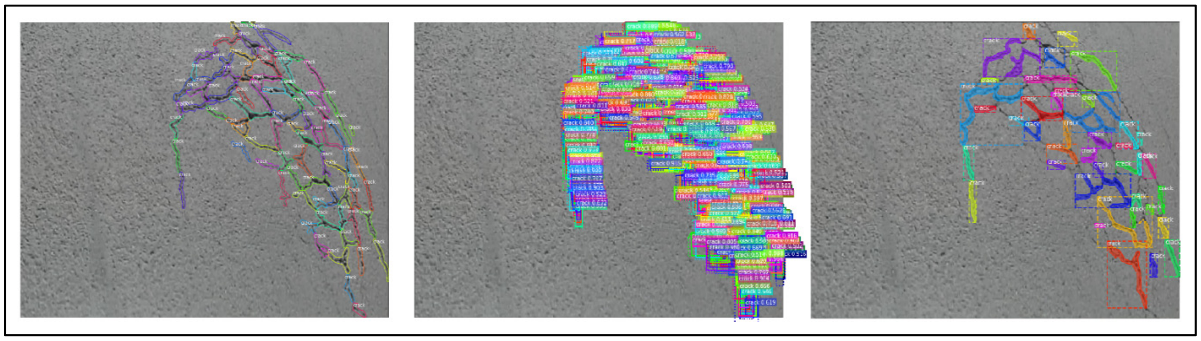


Figure 3.51 Alligator crack segmentation. Left: Ground truth segmentation. Middle: bounding box detection. Right: mask segmentation

To show the capacity of the model in detecting cracks, we split the validation dataset into four categories according to the level of noise in each image (i.e. low, moderate, high noise and one category for alligator cracks). The noise level in an image depends on how frequent the non-crack objects occur (e.g., shadows, gravel, moisture, oil drops, leaves, road lines, etc.). Some noises such as shadows, dark marks from oil, moisture, etc., may be challenging to be detected as non-crack objects. These noises have common features with cracks such as pixel intensity and contrast with the background. However, Mask R-CNN is able to learn not only the crack features, but also the background features; thus, the final model is able to distinguish different cracks and noises. Lower precision is reported on images with high levels of noise where false positives are detected. We also notice that the trained model precisely segment the crack which is critical to assess the severity level of the crack. Table 3.1 shows the precision, the recall and the average intersection over union on the four categories.

Table 3.1 Crack segmentation precision, recall, and AIU in different scenarios

	Low Noise	**Moderate Noise**	**High Noise**	**Alligator Cracks**	**All**
Number of images	15	31	19	5	70
Precision (%)	76.86	77.28	75.73	83.43	77.67
Recall (%)	84.90	84.29	81.58	61.62	79.18
AIU (%)	67.76	67.26	66	54.22	64.6

For reference, figures 3.52, 3.53, and 3.54 below show the detection under different scenarios.

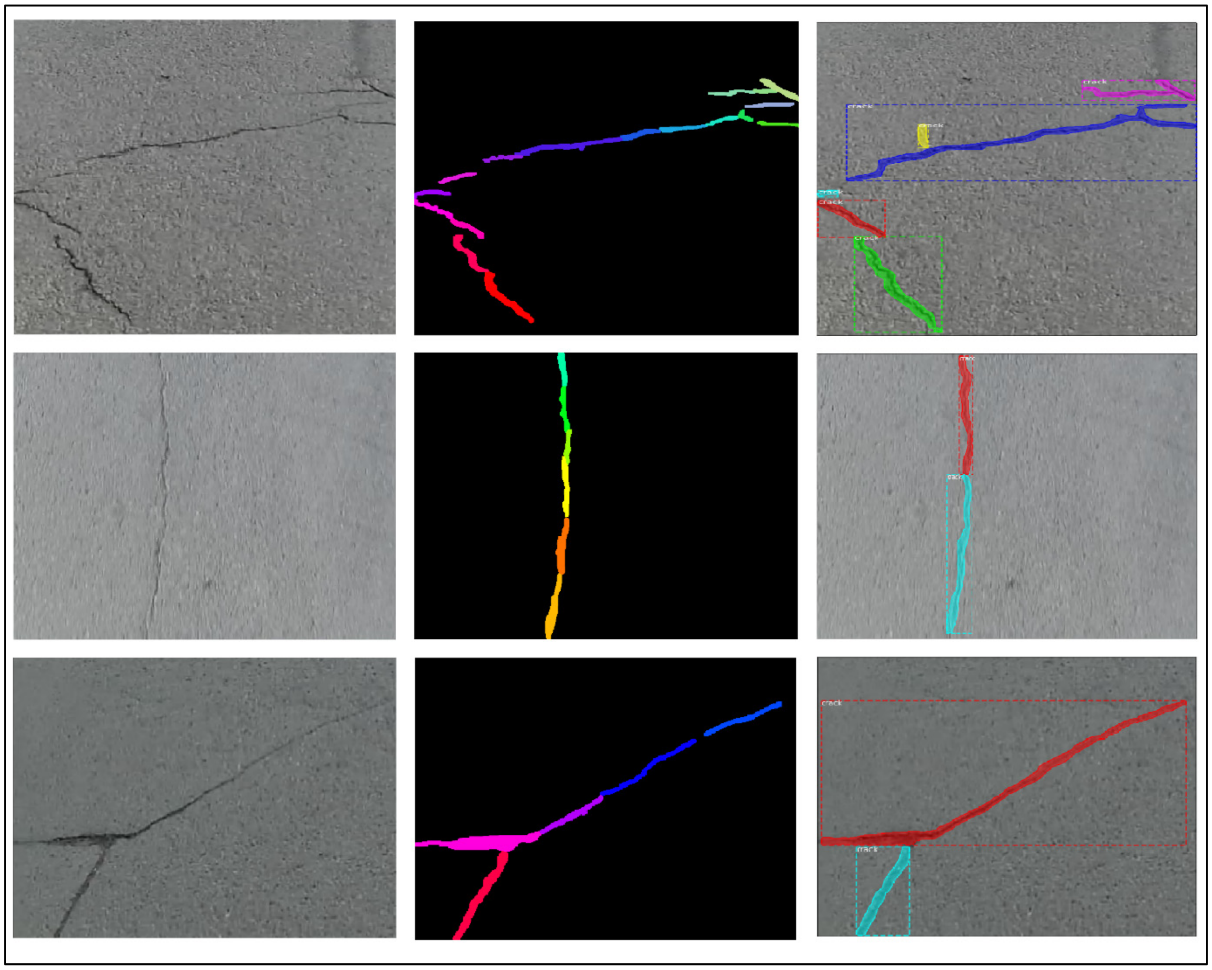


Figure 3.52 Low noise crack segmentation. Left to right columns: RGB image; GT; mask segmentation

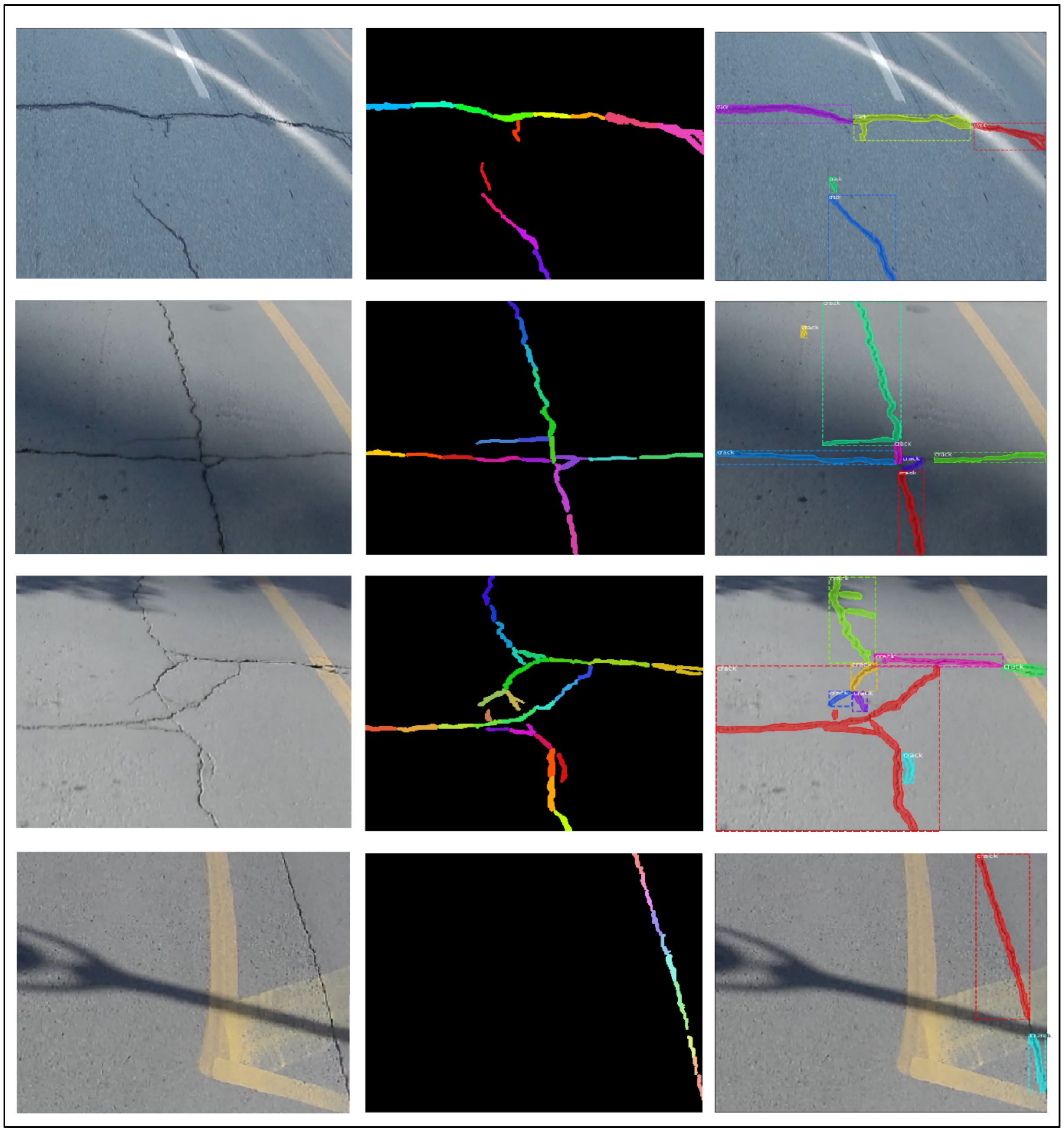


Figure 3.53 Moderate noise crack segmentation. Left to right columns: RGB image; GT; mask segmentation

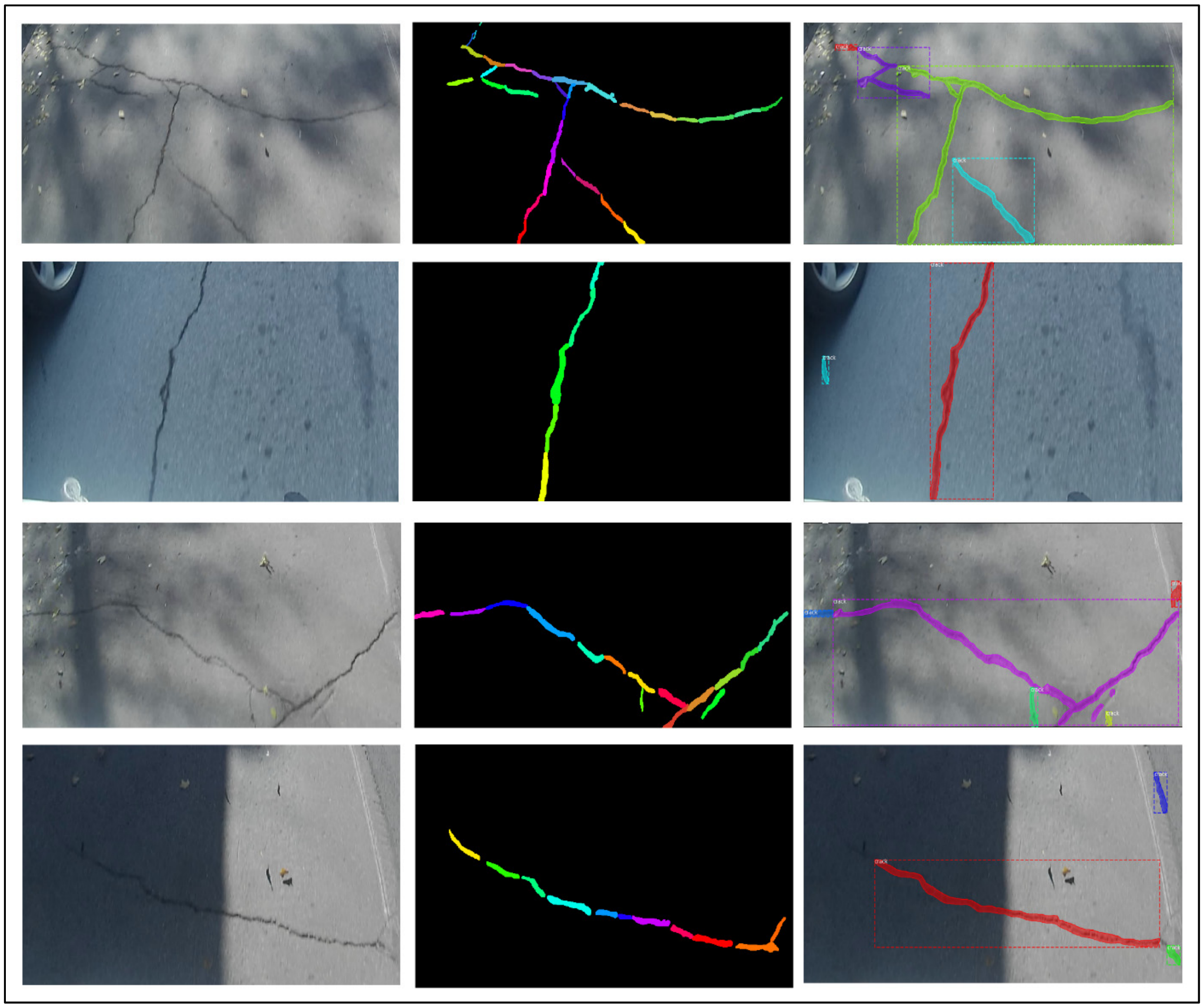


Figure 3.54 High noise crack segmentation. Left to right columns: RGB image; GT; mask segmentation

3.2.3 Conclusion

We compare our results to results from (Yang et al., 2019). Our approach surpasses all the scores reported in (Yang et al., 2019) with a 15.6% improvement on the AIU and 14% improvement on the F1 score. We should also note that the results are reported on two different datasets. However, by comparing our dataset with the Crack500 dataset, we can notice that

our dataset is richer in scenarios, crack shapes, and the images are more challenging. The results show the relevance of our approach in crack segmentation and generalization.
The results prove that the annotation technique and Mask R-CNN that we adopted, can achieve higher precision and recall compared to other state-of-the-art techniques. The results also show that training Mask R-CNN on fractions has a remarkable impact on crack segmentation. Using this approach has the following advantages:

- the annotation technique boosts the number of cracks and non-crack objects provided for the training;
- the annotation technique ensures more relevant crack's pixels inside each bounding box.
- the annotation technique allows a unified pattern across different crack types;
- the integration of the annotation technique with ROI network in Mask R-CNN allows crack fraction's proposals to be passed to the classification phase. Most of the non-crack pixels are rejected during the ROI phase.

Comparing the speed of detection, FPHBN achieves a 0.25s for a 640 x 540p tested on 12G Geforce GTX TITAN X NVIDIA's GPU while Mask R-CNN achieves a 0.35s for 1024 x 1024p image tested on 6G Quadro RTX 3000 NVIDIA's GPU.

CONCLUSION AND RECOMMENDATION

In this project, we test self-driving car sensors in detecting pavement surface distresses. We study the LiDAR in detecting potholes and cameras in detecting and segmenting different types of cracks. We conduct several experiments on the LiDAR to test its capacity in detecting potholes and assess their depths.

In chapter one, we show the different types of pavements and their corresponding pavement distress. We present the developed methods based on manual and autonomous inspection i.e., 2D image processing, 3D reconstruction to detect road deteriorations. The literature review shows the limitations of these methods. Although some of these methods are highly accurate in detecting surface deformations, they are highly expensive to be deployed frequently to scan and assess pavement deteriorations.

In chapter two, we introduce our methods based on conventional self-driving cars. We introduce the current sensing technology deployed on autonomous cars. We show the motivation behind our work. And we present the hardware setup for our experiments and the data collection methods.

In chapter three, we introduce the experiments conducted on the collected datasets. We first test the LIDAR data in an indoor and outdoor environments to test its accuracy and precision. Results show that the current 3D spinning LiDAR technology deployed on self-driving cars lacks precision especially on low reflective surface. It also lacks resolution to detect potholes and assess their characteristics. Surface reconstruction from the outdoor and the indoor scanned point clouds shows a deformation in the elevation of the pavement surface. A more accurate LiDAR technology could solve those limitations.

We also study RGB images and Region-based Deep Neural Network in detecting cracks. We adopt an original method in annotating the collected data, and we train Mask R-CNN for crack detection. Moreover, we address the problem of detecting cracks that have different shapes in different scenarios, while driving at a speed up to 40 km/h. The adopted approach shows its relevance in detecting cracks in noisy environment, under different luminosity levels, and cracks with complex shapes. The annotation method also gives a remarkable advantage in detecting cracks regardless of their shapes, severity levels and orientations.

More enhancement can be introduced while using the same method. For instance, we strongly recommend a higher-resolution video (i.e. 4K videos instead of Full High Definition, HD, videos) since cracks can be very thin. We also recommend a higher frame rate to reduce the blurring effect at high speed. Moreover, we suggest to use a larger and richer training dataset consisting in different scenarios to improve the results of the mask segmentation. We also suggest to train and validate the Mask R-CNN model on different datasets and compare the results with the results of other approaches. We also suggest to use K-fold cross validation to evaluate the performance of the adopted approach.

We also note that the severity level of the crack can be computed as the mask R-CNN returns a pixel-wise segmentation of the detected cracks. We also suggest to add more distress categories into the training dataset to enhance the results and generalize the model. The geographical coordinates can be also added to locate each crack; thus, a road map can be generated with the severity level of each individual crack. We also recommend the use of debluring algorithms to reduce the blurring effect at high speed, thus, a high-speed scans can be performed.

Furthermore, combining the 3D data with the 2D images helps in extracting depth features from the pavement surface. However, elevation measurements from the 3D LiDAR should be accurate to properly extract relevant features. With accurate 3D measurements, we can add the 3D data to the R-CNN adopted in this research to extract depth features. By doing so, we expect further enhancement in the precision and the recall of the trained model.

Finally, a thorough market study can be conducted to check if the developed tool for crack detection is feasible to be deployed on a large scale.

ANNEX I

ACQUIRING THE LIDAR DATA

To acquire data from the LiDAR we develop a software that reads the UDP packets and save them in a text file. The generated text file is 4 times smaller than the file generated by the software developed by the Ouster Company. No data are lost during the process of writing the UDP packets. After generating the text file, which is in hexadecimal format, we convert it to a readable format, the file format is named MyFormat. ANNEX II shows a sample of the acquired data in the readable format.

The code developed also provides a tool to check if there is any packet loss during the read and write process. While using the code provided by the Ouster Company we notice that many packets are dropped due to the UDP protocol. However we fixed this issue by reducing the amount of the written data, thus, the network congestion is significantly reduced.

To visualize the LiDAR data we develop a code that reads data from MyFormat file and generates a video visualization of the signal, reflectivity, ambient and range data.

We also create a synchronization application between the RGB videos and the LiDAR frames. This application allows us to synchronize the data between the two sensors according to a specific time frame that can be manually introduced. Although the synchronization was successful, a pixel to pixel overlapping between the two sensors was partially successful. This is due to several reasons:

- the resolution difference between the two sensors;
- the camera lens artefacts especially on the edges and corners of each RGB frame;
- the position of the LiDAR and the camera are not perfectly overlapped;
- the scanning mechanism on the LiDAR. In contrast to the camera where each frame is captured instantaneously, LiDAR points are captured while it spins, thus, while the vehicle

is moving. This leads to a misalignment between the RGB frames and the LiDAR frames, (e.g., a LiDAR frame takes 0.1s to be captured, a RGB frame takes 0.004s);

The code to acquire and process the LiDAR data is publicly published on GitHub: https://github.com/NizarTarabay/Extract_packets_os16.

In the following we show the main parts of the code and their functionality:

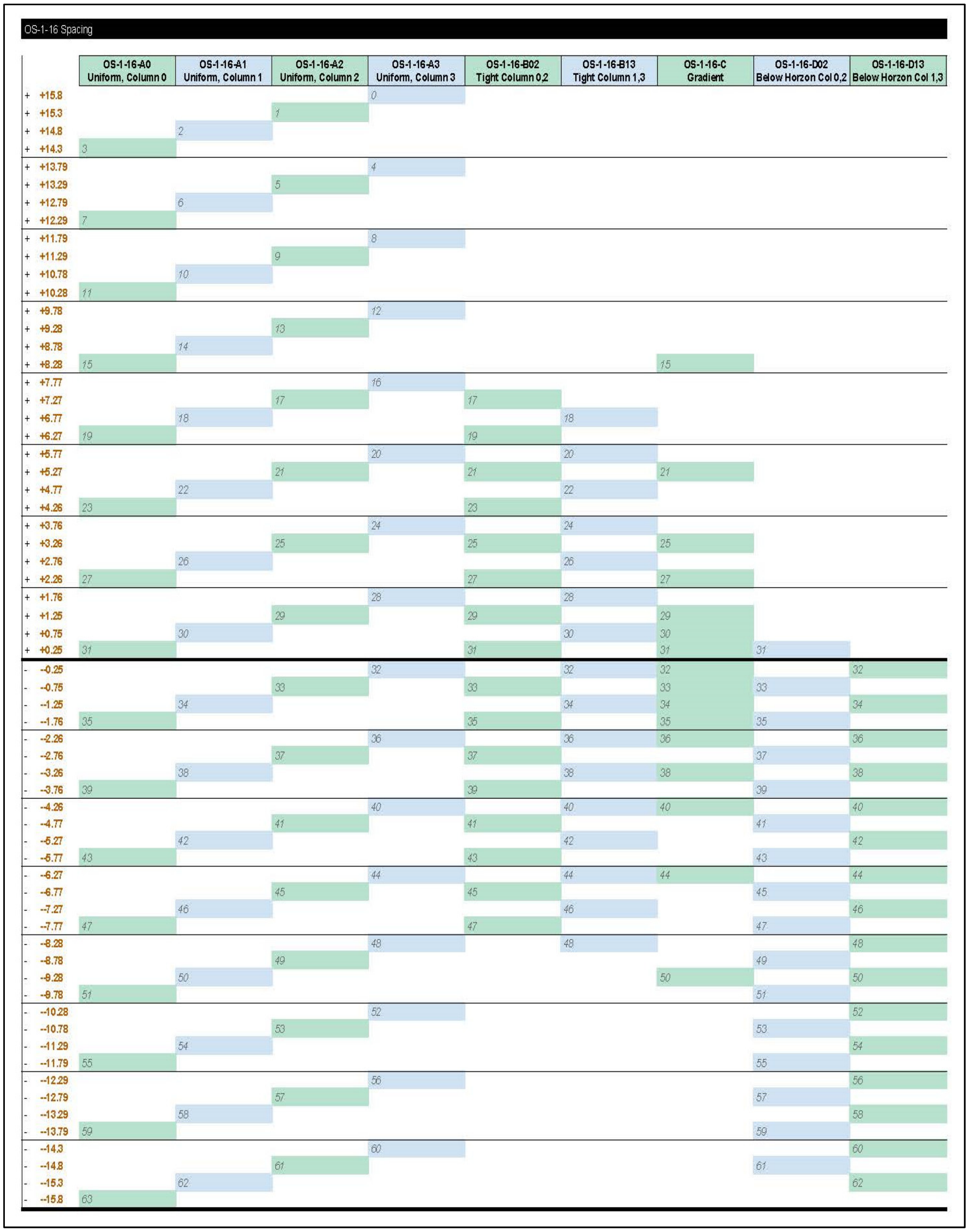
- Write_lidar_packets_bytes.py: run this file after the LiDAR is connected to the computer. This will save the UDP packets into a text file;
- Translate_lidar_packets.py: run this file to convert the saved packets into a readable format (ANNEX II);
- Helpers.py: A library containing all the functions to operate on the readable format file, i.e. extract LiDAR signals, generate point clouds, check for dropped frames;
- Frame_drop_checker.py: run this file to check if frames or data are dropped during the process of recording data;
- Display_lidar_packets.py: displays the data acquired by the LiDAR as a video format;
- Video_processing/videos_next_to_each_other.py: gives the possibility to synchronise and to display the RGB and the LiDAR videos.

Ouster provides a full description to connect the LiDAR to the computer at: https://github.com/Ouster-lidar/Ouster_example

ANNEX II

LIDAR CHANNEL ANGLES

Figure-A II-1 shows the LiDAR channels angle. “Os-1-16 spacing” is a document provided with Ouster LiDAR, it gives information on the distribution of the LiDAR channels in different types of Os-LiDAR. In our case we are using the OS-1-16-A1 Uniform, column 1 distribution. The numbers in the first column are the channels’ angle. The rest of the columns are for different LiDAR configurations.



OS-1-16 Spacing

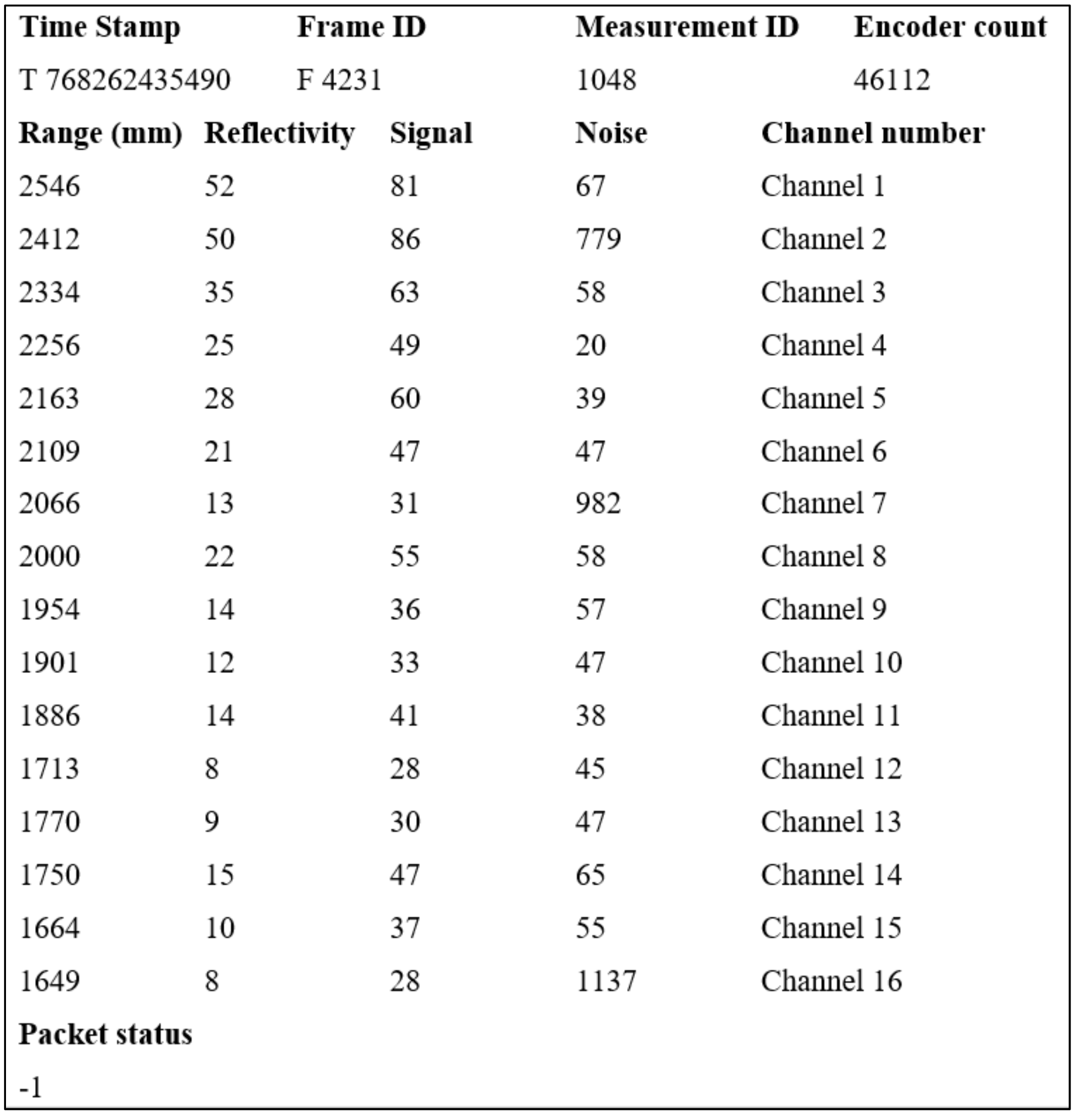
		OS-1-16-A0 Uniform, Column 0	OS-1-16-A1 Uniform, Column 1	OS-1-16-A2 Uniform, Column 2	OS-1-16-A3 Uniform, Column 3	OS-1-16-B02 Tight Column 0,2	OS-1-16-B13 Tight Column 1,3	OS-1-16-C Gradient	OS-1-16-D02 Below Horzon Col 0,2	OS-1-16-D13 Below Horzon Col 1,3
+	+15.8				0					
+	+15.3			1						
+	+14.8		2							
+	+14.3	3								
+	+13.79				4					
+	+13.29			5						
+	+12.79		6							
+	+12.29	7								
+	+11.79				8					
+	+11.29			9						
+	+10.78		10							
+	+10.28	11								
+	+9.78				12					
+	+9.28			13						
+	+8.78		14							
+	+8.28	15						15		
+	+7.77				16					
+	+7.27			17		17				
+	+6.77		18				18			
+	+6.27	19				19				
+	+5.77				20		20			
+	+5.27			21		21		21		
+	+4.77		22				22			
+	+4.26	23				23				
+	+3.76				24		24			
+	+3.26			25		25		25		
+	+2.76		26				26			
+	+2.26	27				27		27		
+	+1.76				28		28			
+	+1.25			29		29		29		
+	+0.75		30				30	30		
+	+0.25	31				31		31	31	
-	-0.25				32		32	32		32
-	-0.75			33		33		33	33	
-	-1.25		34				34	34		34
-	-1.76	35				35		35	35	
-	-2.26				36		36	36		36
-	-2.76			37		37			37	
-	-3.26		38				38	38		38
-	-3.76	39				39			39	
-	-4.26				40		40	40		40
-	-4.77			41		41			41	
-	-5.27		42				42			42
-	-5.77	43				43			43	
-	-6.27				44		44	44		44
-	-6.77			45		45			45	
-	-7.27		46				46			46
-	-7.77	47				47			47	
-	-8.28				48		48			48
-	-8.78			49					49	
-	-9.28		50					50		50
-	-9.78	51							51	
-	-10.28				52					52
-	-10.78			53					53	
-	-11.29		54							54
-	-11.79	55							55	
-	-12.29				56					56
-	-12.79			57					57	
-	-13.29		58							58
-	-13.79	59							59	
-	-14.3				60					60
-	-14.8			61					61	
-	-15.3		62							62
-	-15.8	63								

Figure-A II-1 “OS-16 spacing” shows the 16 LiDAR channels distribution

ANNEX III

THE READABLE FORMAT FILE

Figure-A III-1 shows how the LiDAR measurements are saved. This file format, with the Ouster-16 LiDAR user manual are very important for further use of the code presented in Annex I.



Time Stamp	**Frame ID**		**Measurement ID**	**Encoder count**
T 768262435490	F 4231		1048	46112
Range (mm)	**Reflectivity**	**Signal**	**Noise**	**Channel number**
2546	52	81	67	Channel 1
2412	50	86	779	Channel 2
2334	35	63	58	Channel 3
2256	25	49	20	Channel 4
2163	28	60	39	Channel 5
2109	21	47	47	Channel 6
2066	13	31	982	Channel 7
2000	22	55	58	Channel 8
1954	14	36	57	Channel 9
1901	12	33	47	Channel 10
1886	14	41	38	Channel 11
1713	8	28	45	Channel 12
1770	9	30	47	Channel 13
1750	15	47	65	Channel 14
1664	10	37	55	Channel 15
1649	8	28	1137	Channel 16
Packet status				
-1				

Figure-A III-1 The readable format file

ANNEX IV

MASK R-CNN CONFIGURATION

Table-A IV-1 presents the configuration of Mask R-CNN adopted in our experiment:

Table-A IV-1 The configuration of Mask R-CNN

Configuration	**Value**
BACKBONE	resnet50
BACKBONE_STRIDES	[4, 8, 16, 32, 64]
BATCH_SIZE	1
DETECTION_MAX_INSTANCES	100
DETECTION_MIN_CONFIDENCE	0.7
DETECTION_NMS_THRESHOLD	0.3
IMAGE_CHANNEL_COUNT	3
IMAGE_MAX_DIM	1024
IMAGE_MIN_DIM	512
IMAGE_RESIZE_MODE	Square
IMAGE_SHAPE	[1024 1024 3]
LEARNING_MOMENTUM	0.9
LEARNING_RATE	0.001
MASK_POOL_SIZE	14
MASK_SHAPE	[28, 28]
MAX_GT_INSTANCES	50
NUM_CLASSES	2
POOL_SIZE	7
POST_NMS_ROIS_INFERENCE	3000
POST_NMS_ROIS_TRAINING	4000

PRE_NMS_LIMIT	6000
ROI_POSITIVE_RATIO	0.33
RPN_ANCHOR_RATIOS	[0.5, 1, 2]
RPN_ANCHOR_SCALES	(8, 16, 32, 64)
RPN_ANCHOR_STRIDE	1
RPN_TRAIN_ANCHORS_PER_IMAGE	256
VALIDATION_STEPS	76
WEIGHT_DECAY	0.0001

ANNEX V

PREPARING THE DATASET, TRAINING AND EVALUATING MASK R_CNN

To prepare a new dataset please follow the direction provided on coco_annotation_tool repository published publicly at:
https://github.com/NizarTarabay/coco_annotation_tool

We use Matterport-Mask R-CNN implementation (Abdulla, 2017). This implementation is available under the MIT license, published on GitHub:
https://github.com/matterport/Mask_RCNN.

To run the training and the inference on our dataset and replicate the above results we attach to this report files containing a *jupyter notebook* to run the training and the inference, and the training/validation dataset with their documentation.

LIST OF BIBLIOGRAPHICAL REFERENCES

Abbas, S. M., & Muhammad, A. (2012). Outdoor RGB-D SLAM Performance in Slow Mine Detection. At *ROBOTIK 2012; 7th German Conference on Robotics* (pp. 1-6).

Abdulla, W. (2017). Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow: Github. Retrieved from https://github.com/matterport/Mask_RCNN

Akagic, A., Buza, E., Omanovic, S., & Karabegovic, A. (2018). Pavement crack detection using Otsu thresholding for image segmentation. At *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1092-1097). doi: 10.23919/MIPRO.2018.8400199

Alfarrarjeh, A., Trivedi, D., Kim, S. H., & Shahabi, C. (2018). A deep learning approach for road damage detection from smartphone images. At *2018 IEEE International Conference on Big Data (Big Data)* (pp. 5201-5204). IEEE.

Amhaz, R., Chambon, S., Idier, J., & Baltazart, V. (2016). Automatic Crack Detection on Two-Dimensional Pavement Images: An Algorithm Based on Minimal Path Selection. *IEEE Transactions on Intelligent Transportation Systems, 17*(10), 2718-2729. doi: 10.1109/TITS.2015.2477675

Attard, L., Debono, C. J., Valentino, G., Castro, M. D., Masi, A., & Scibile, L. (2019). Automatic Crack Detection using Mask R-CNN. At *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)* (pp. 152-157). doi: 10.1109/ISPA.2019.8868619

Berger, M., Tagliasacchi, A., Seversky, L., Alliez, P., Guennebaud, G., Levine, J., . . . Silva, C. (2016). A Survey of Surface Reconstruction from Point Clouds. *Computer Graphics Forum*, n/a-n/a. doi: 10.1111/cgf.12802

Bodla, N., Singh, B., Chellappa, R., & Davis, L. (2017). Improving Object Detection With One Line of Code.

Cava, M. d. (2016). Ford, Baidu bet $150M on Velodyne laser radar. Retrieved from https://www.usatoday.com/story/tech/news/2016/08/16/ford-baidu-bet-150m-velodyne-laser-radar/88813028/

Chollet, Francois, & others, a. (2015). Keras. keras.io. Retrieved from https://keras.io

CloudCompare. (2018). CloudCompare: Retrieved from http://www.cloudcompare.org/.

Coenen, T. B. J., & Golroo, A. (2017). A review on automated pavement distress detection methods. *Cogent Engineering, 4*(1). doi: 10.1080/23311916.2017.1374822. Retrieved from http://doi.org/10.1080/23311916.2017.1374822

D6433-18, A. (2018). *Standard Practice for Roads and Parking Lots Pavement Condition Index Surveys*. Compass.

Deng, J., Dong, W., Socher, R., Li, L., Kai, L., & Li, F.-F. (2009). ImageNet: A large-scale hierarchical image database. At *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248-255). doi: 10.1109/CVPR.2009.5206848

Eisenbach, M., Stricker, R., Seichter, D., Amende, K., Debes, K., Sesselmann, M., . . . Gross, H. (2017). How to get pavement distress detection ready for deep learning? A systematic approach. At *2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 2039-2047). doi: 10.1109/IJCNN.2017.7966101

Fractal.ai. (2019). Mask R-CNN Unmasked. Retrieved on 29, June from https://medium.com/@fractaldle/mask-r-cnn-unmasked-c029aa2f1296

GIMP. (2019). GIMP: https://www.gimp.org.

Girshick, R. (2015). Fast R-CNN. At *2015 IEEE International Conference on Computer Vision (ICCV)* (pp. 1440-1448). doi: 10.1109/ICCV.2015.169

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. At *2014 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 580-587). doi: 10.1109/CVPR.2014.81

Girshick, R., Radosavovic, I., Gkioxari, G., Dollar, P., & He, K. (2018). Detectron: Github. Retrieved from https://github.com/facebookresearch/detectron

Gopalakrishnan, K., Khaitan, S. K., Choudhary, A., & 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.

Guan, H., Li, J., Cao, S., & Yu, Y. (2016). Use of mobile LiDAR in road information inventory: a review. *International Journal of Image and Data Fusion, 7*(3), 219-242. doi: 10.1080/19479832.2016.1188860. Retrieved from https://doi.org/10.1080/19479832.2016.1188860

Hahnloser, R. H. R., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., & Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature, 405*, 947. doi: 10.1038/35016072. Retrieved from https://ui.adsabs.harvard.edu/abs/2000Natur.405..947H

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. At *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 2980-2988). doi: 10.1109/ICCV.2017.322

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. At *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770-778). doi: 10.1109/CVPR.2016.90

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

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. At *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).

Huang, W., & Jing, Z. (2007). Evaluation of focus measures in multi-focus image fusion. *Pattern Recognition Letters, 28*(4), 493-500. doi: https://doi.org/10.1016/j.patrec.2006.09.005. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167865506002352

Jaderberg, M., Simonyan, K., Zisserman, A., & Kavukcuoglu, K. (2015). *Spatial transformer networks* presented at Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, Montreal, Canada.

Joubert, D., Tyatyantsi, A., Mphahlehle, J., & Manchidi, V. (2011, 23-25 November 2011). *Pothole Tagging System* presented at Council for Scientific and Industrial Research, 4th Robotics and Mechatronics Conference of South Africa (RobMech 2011), CSIR International Conference Centre, Pretoria. Retrieved from http://hdl.handle.net/10204/5384

Jung, A., Wada, K., Crall, J., Tanaka, S., Graving, J., Reinders, C., . . . others. (2020). imgaug: github. Retrieved from https://github.com/aleju/imgaug

Kazhdan, M., & Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Trans. Graph., 32*(3), Article 29. doi: 10.1145/2487228.2487237. Retrieved from https://doi.org/10.1145/2487228.2487237

Kim, T., & Ryu, S.-K. (2014). Review and analysis of pothole detection methods. *Journal of Emerging Trends in Computing and Information Sciences, 5*(8), 603-608.

Koch, C., Georgieva, K., Kasireddy, V., Akinci, B., & Fieguth, P. (2015). A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Advanced Engineering Informatics, 29*(2), 196-210. doi: https://doi.org/10.1016/j.aei.2015.01.008. Retrieved from http://www.sciencedirect.com/science/article/pii/S1474034615000208

Kuhl, A., Wöhler, C., Krüger, L., d'Angelo, P., & Gross, H.-M. (2006). *Monocular 3D Scene Reconstruction at Absolute Scales by Combination of Geometric and Real-Aperture Methods* (Vol. 4174). doi: 10.1007/11861898_61

Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D., & Matas, J. (2018). Deblurgan: Blind motion deblurring using conditional adversarial networks. At *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8183-8192).

Laurent, J., Hébert, J. F., Lefebvre, D., & Savard, Y. (2012). Using 3D Laser Profiling Sensors for the Automated Measurement of Road Surface Conditions. At (pp. 157-167). Springer Netherlands.

Laurent, J., Petitclerc, B., Samson, E., & Inc., P. S. (2018). High resolution multi-lane road surface mapping using 3D laser. At *International Federation of Surveyors FIG Congress*.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. At *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117-2125).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., . . . Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. At D. Fleet, T. Pajdla, B. Schiele & T. Tuytelaars (Éds.), *Computer Vision – ECCV 2014* (pp. 740-755). Springer International Publishing.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. At *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3431-3440).

Mathavan, S., Kamal, K., & Rahman, M. (2015). A Review of Three-Dimensional Imaging Technologies for Pavement Distress Detection and Measurements. *IEEE Transactions on Intelligent Transportation Systems, 16*(5), 2353-2362. doi: 10.1109/TITS.2015.2428655. Retrieved from http://dx.doi.org/10.1109/TITS.2015.2428655

Miller, J. S., & Bellinger, W. Y. (2014). *Distress identification manual for the long-term pavement performance program*. United States. Federal Highway Administration. Office of Infrastructure Research and Development.

Moazzam, I., Kamal, K., Mathavan, S., Usman, S., & Rahman, M. (2013). Metrology and visualization of potholes using the microsoft kinect sensor. At *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)* (pp. 1284-1291). doi: 10.1109/ITSC.2013.6728408

Nayar, S. K., & Nakagawa, Y. (1994). Shape from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 16*(8), 824-831. doi: 10.1109/34.308479

Oliveira, H., & Correia, P. L. (2014). CrackIT — An image processing toolbox for crack detection and characterization. At *2014 IEEE International Conference on Image Processing (ICIP)* (pp. 798-802). doi: 10.1109/ICIP.2014.7025160

Ouster. (2018). *Ouster Hardware User Guide*.

Polaczyk, P., Huang, B., Shu, X., & Gong, H. (2019). Investigation into Locking Point of Asphalt Mixtures Utilizing Superpave and Marshall Compactors. *Journal of Materials in Civil Engineering, 31*(9), 04019188. doi: doi:10.1061/(ASCE)MT.1943-5533.0002839. Retrieved from https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29MT.1943-5533.0002839

Ragnoli, A., De Blasiis, M. R., & Di Benedetto, A. (2018). Pavement distress detection methods: A review. *Infrastructures, 3*(4), 58.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). *Faster R-CNN: towards real-time object detection with region proposal networks* presented at Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, Montreal, Canada.

ROMDAS. (2016). ROMDAS System Overview. In ROMDAS (Éd.). New Zealand.

Santana, E., & Hotz, G. (2016). Learning a driving simulator. *arXiv preprint arXiv:1608.01230*.

Sautya, M. (2018). Advantages and Disadvantages Of Cement Concrete Road (Rigid Pavements). Retrieved on 1 may, 2020 from https://civilnoteppt.com/advantages-and-disadvantages-of-cement-concrete-road-rigid-pavements/

Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z. (2016). Automatic Road Crack Detection Using Random Structured Forests. *IEEE Transactions on Intelligent Transportation Systems, 17*(12), 3434-3445. doi: 10.1109/TITS.2016.2552248

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

Subbarao, M., & Surya, G. (1994). Depth from defocus: A spatial domain approach. *International Journal of Computer Vision, 13*(3), 271-294. doi: 10.1007/BF02028349. Retrieved from https://doi.org/10.1007/BF02028349

Sun, Y., Duthaler, S., & Nelson, B. J. (2005). Autofocusing algorithm selection in computer microscopy. At *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 70-76). IEEE.

Szymański, P., Pikos, M., & Nowotarski, P. (2017). Concrete road surface with the use of cement concrete - selected results. *Procedia Engineering, 208*, 166-173. doi: https://doi.org/10.1016/j.proeng.2017.11.035. Retrieved from http://www.sciencedirect.com/science/article/pii/S1877705817360289

Tyson, S., & Tayabji, S. D. (2012). *Continuously Reinforced Concrete Pavement Performance and Best Practices*. United States. Federal Highway Administration.

Uijlings, J., Sande, K., Gevers, T., & Smeulders, A. (2013). Selective Search for Object Recognition. *International Journal of Computer Vision, 104*, 154-171. doi: 10.1007/s11263-013-0620-5

Xie, S., & Tu, Z. (2015). Holistically-nested edge detection. At *Proceedings of the IEEE international conference on computer vision* (pp. 1395-1403).

Yang, F., Zhang, L., Yu, S., Prokhorov, D., Mei, X., & Ling, H. (2019). Feature pyramid and hierarchical boosting network for pavement crack detection. *IEEE Transactions on Intelligent Transportation Systems*.

Zhang, L., Yang, F., Daniel Zhang, Y., & Zhu, Y. J. (2016). Road crack detection using deep convolutional neural network. At *2016 IEEE International Conference on Image Processing (ICIP)* (pp. 3708-3712). doi: 10.1109/ICIP.2016.7533052

Zhang, Y., Chen, C., Wu, Q., Lu, Q., Zhang, S., Zhang, G., & Yang, Y. (2018). A Kinect-Based Approach for 3D Pavement Surface Reconstruction and Cracking Recognition. *IEEE Transactions on Intelligent Transportation Systems*, 1-12. doi: 10.1109/TITS.2018.2791476

Zou, Q., Cao, Y., Li, Q., Mao, Q., & Wang, S. (2012). CrackTree: Automatic crack detection from pavement images. *Pattern Recognition Letters, 33*(3), 227-238. doi: https://doi.org/10.1016/j.patrec.2011.11.004. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167865511003795