

**DETECTION WITH K-NEAREST NEIGHBOUR ALGORITHM
FOR CRACKED CONCRETE**



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**DETECTION WITH K-NEAREST NEIGHBOUR ALGORITHM FOR CRACKED
CONCRETE**

MUHAMMAD ZHARFAN BIN MAHADZIR



Faculty of Mechanical Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2022

DECLARATION

I declare that this project report entitled “Detection With K-Nearest Neighbour Algorithm for Cracked Concrete” is the result of my own work except as cited in the references

Signature :.....

Name : MUHAMMAD ZHARFAN BIN MAHADZIR

Date :.....



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

APPROVAL

I hereby declare that I have read this project report and in my opinion this report is sufficient in terms of scope and quality for the award of the degree of Bachelor of Mechanical Engineering.

Signature :

Supervisor's Name : MR. ZAIRULAZHA BIN ZAINAL

Date :



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DEDICATION

To my beloved parent and family.



ABSTRACT

Nowadays, the construction sector plays a vital role in country development. Many high and unique buildings were made to show how the economic status and progress of the country developed. Thus, solid and high efficiency materials such as concrete are needed to construct modern and robust buildings. Even though concrete has high material durability compared to other materials, that specialty cannot deny that the concrete has one main problem if not well maintained. The problem is the cracked wall surface structure. Generally, three types of cracks can be found at the concrete surface, there is a minor crack, moderate crack, and severe crack. This will make the building safety low. The current method nowadays to perform structural health monitoring (SHM) for the concrete wall is by using manual inspection that is visual inspection. The reason why SHM is performed is to detect concrete surfaces and monitor the current structure condition. Several problems have been carried out in this study when the manual inspection is applied to inspect the presence of cracks toward the concrete surface. The first problem is that manual inspection consumes more time to inspect the presence of cracks at the surface since the inspection is done visually and can be done by an experienced inspector only. Next is, hazardous environment to perform inspection increase the safety risk toward the inspector since the presence of crack indicates that the building is not in good condition. Lastly, a limited number of experienced inspectors has become one of the problems for current crack building condition inspection. Thus, the main objective of this study is to produce a technique that can detect the present crack at the concrete surface by using the K-Nearest Neighbor (KNN) algorithm, which can reduce the crack inspection time, reduce the number of inspectors at the hazardous area and lastly required fewer experiences manpower for crack inspection. This technique can detect the presence of crack by using crack concrete surface images. The highest crack classification accuracy obtained in this work was 94.40% correct class classification. The suitable number of greyscale intensity level that had been used is 0.4 for features extraction and the K value is 3. The best number of training datasets that had been used to archive 94.40% accuracy level is 800 crack datasets and 100 non-crack datasets that use 90.00% training and 10.00% testing from the datasets. In conclusion, the crack detection method was able to detect the crack dataset with high correct classification accuracy.

ABSTRAK

Pada masa kini, sektor pembinaan memainkan peranan penting dalam pembangunan sesebuah negara. Banyak bangunan tinggi dan unik dibuat untuk menunjukkan bagaimana taraf ekonomi dan kemajuan negara tersebut berkembang. Oleh itu, bahan kukuh dan tinggi kecekapan seperti konkrit diperlukan untuk membina bangunan moden dan kukuh. Walaupun konkrit mempunyai ketahanan bahan yang tinggi berbanding bahan lain, keistimewaan itu tidak dapat menafikan bahawa konkrit mempunyai satu masalah utama jika tidak diselenggara dengan baik. Masalahnya ialah struktur permukaan dinding yang retak. Secara amnya, tiga jenis rekahan boleh didapati pada permukaan konkrit iaitu retak kecil, retak sederhana dan retak teruk. Ini akan menjadikan keselamatan bangunan tersebut rendah. Kaedah pada masa kini yang digunakan untuk melaksanakan pemantauan kesihatan struktur (SHM) bagi dinding konkrit adalah dengan menggunakan pemeriksaan manual iaitu pemeriksaan visual. Tujuan SHM dilakukan adalah untuk mengesan keretakan pada permukaan konkrit dan memantau keadaan semasa struktur konkrit. Beberapa masalah telah ditemui dalam kajian ini apabila pemeriksaan manual digunakan untuk memeriksa kehadiran retakan pada permukaan konkrit. Masalah pertama ialah pemeriksaan manual memerlukan lebih banyak masa untuk memeriksa kehadiran retakan pada permukaan konkrit memandangkan pemeriksaan dilakukan secara visual dan boleh dilakukan oleh pemeriksa yang berpengalaman sahaja. Seterusnya, persekitaran berbahaya untuk melakukan pemeriksaan meningkatkan risiko keselamatan terhadap pemeriksa kerana kehadiran keretakan menunjukkan bangunan itu tidak dalam keadaan baik. Akhir sekali, bilangan pemeriksa berpengalaman yang terhad telah menjadi salah satu masalah untuk memeriksa kehadiran retakan pada permukaan. Justeru, objektif utama kajian ini adalah untuk menghasilkan satu teknik yang dapat mengesan rekahan yang ada pada permukaan konkrit dengan menggunakan algoritma K-Nearest Neighbor (KNN), yang dapat mengurangkan masa pemeriksaan retak, mengurangkan bilangan pemeriksa pada kawasan berbahaya dan akhirnya memerlukan lebih sedikit tenaga kerja pengalaman untuk pemeriksaan retak. Teknik ini boleh mengesan kehadiran retak dengan menggunakan imej permukaan konkrit yang retak. Ketepatan pengelasan retak tertinggi yang diperoleh dalam kajian ini ialah 94.40% pengelasan kelas betul. Bilangan tahap keamatan skala kelabu yang sesuai yang telah digunakan ialah 0.4 untuk pengekstrakan ciri dan nilai K ialah 3. Bilangan set data untuk latihan yang terbaik telah digunakan untuk mendapatkan 94.40% tahap ketepatan pengelasan ialah 800 set data retak dan 100 set data tidak retak yang menggunakan 90.00% latihan dan 10.00% ujian daripada set data. Kesimpulannya, kaedah ini dapat mengesan dataset retak dengan ketepatan pengelasan betul yang tinggi.

ACKNOWLEDGEMENTS

First and foremost, Alhamdulillah, praise be to Allah, the Almighty, for His bounties in allowing me to complete my research journey successfully.

With deep appreciation, I extend an unreserved thank you to my project supervisor, Mr Zairulazha Bin Zainal, for allowing me to carry out this research with all of his assistance and supervision. During Pandemic COVID-19, I appreciate your encouragement and enthusiasm, as well as your knowledge and expertise. The study process was made easier thanks to the assistance provided.

I am highly appreciative to my loving family, my parents and my siblings, who provided me with an incredible amount of support and encouragement as I worked to accomplish this project. Special thanks to my family members, Sabirun Bin Md Saad and Hanif Bin Hussein, for supporting me in financing while completing this work.

Finally, a huge thank you to my faculty of mechanical engineering colleagues, particularly Siti Sarah Ain Binti Mohamad Sofi, for her intelligent suggestions and guidance while performing the image categorization test.

TABLE OF CONTENTS

	PAGE
DECLARATION	i
APPROVAL	ii
DEDICATION	iii
ABSTRACT	iv
ABSTRAK	v
ACKNOWLEDGEMENTS	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	x
LIST OF TABLES	xiv
LIST OF ABBEREVATIONS	xvi
CHAPTER	
1. INTRODUCTION	1
1.1 Background Study	2
1.2 Problem Statement	4
1.3 Objectives	5
1.4 Scope of Project	6
2. LITERATURE REVIEW	7
2.1 Introduction	7
2.2 Factor of Cracking Concrete	8
2.3 Crack Classification	9
2.4 Types of Images for Images Processing	12
2.5 Architecture Process	14
2.6 Image Processing Method	16
2.6.1 Feature Extraction	18
2.7 Classification Algorithm	21
2.7.1 K Nearest Neighbour (KNN) Algorithm	21
2.8 Classification Analysis	27
2.8.1 Confusion Matrix	27

3. METHODOLOGY	31
3.1 Introduction	31
3.2 General Methodology	32
3.3 Image Preparation	37
3.3.1 Images Quantity	38
3.3.2 Images Specification	42
3.3.3 Type of Crack	42
3.4 Image Processing	45
3.4.1 Greyscale Image Conversion	45
3.4.2 Image Binarizing Process	46
3.5 Feature Extraction	47
3.6 Images Classification with K Nearest Neighbour (KNN) Algorithm	52
3.6.1 Dataset for Training and Testing Quantity	52
3.6.2 K Value for KNN Crack Detection	54
3.6.3 KNN Testing Visualisation Result	54
3.6.4 KNN Visualisation Description	56
3.7 Result Evaluation	58
3.7.1 Confusion Matrix	58
3.8 Random Images Testing	62
4. RESULT AND DISCUSSION	63
4.1 Introduction	63
4.2 Images Processing and Features Extraction	63
4.3 K-Nearest Neighbor (KNN) Image Classification Result	68
4.3.1 The Testing Result For 80% Training And 20% Testing	69
4.3.2 The Testing Result For 90% Training And 10% Testing	75
4.3.3 Crack Detection Testing by Using Different Value Training Dataset	82
4.3.4 K- Nearest Neighbor (KNN) Visualization Discussion	89
4.4 Classification Accuracy with Different K- Nearest Neighbor (KNN) Value Analysis	92
4.5 Testing with Different Number of Training Datasets Analysis	94
4.6 Confusion Matrix Testing Result	97

4.7	Random Images Testing Result	100
5.	CONCLUSION AND RECOMMENDATION	108
5.1	Conclusion	108
5.2	Recommendation for Future Work	109
	REFERENCES	110
	APPENDICES	115



LIST OF FIGURES

FIGURE	TITLE	PAGE
2.1	Shrinkage cracking sample toward concrete structure from the source (Larosche, 2009).	9
2.2	Minor crack sample from the source (Kaish et al., 2018).	10
2.3	Moderate crack sample from the source (Kaish et al., 2018).	11
2.4	Severe crack sample from the source (Kaish et al., 2018).	12
2.5	Binary image sample from the source (Amlan and Devdas, 2000).	13
2.6	Greyscale image sample from the source (Varsha et al., 2019).	13
2.7	RGB image sample source from (Maurits, 2020).	14
2.8	Crack image feature extraction source from (Varsha et.al 2019).	18
2.9	Feature extraction by using Sobel method from sources (Ali & Clausi, 2001).	20
2.10	Feature extraction by using Canny method from sources (Ali & Clausi, 2001).	20
2.11	KNN algorithm classification concept from the source (Imandoust & Bolandraftar, (2013).	21
2.12	Classification accuracy different value of K source from (Chih et al., 2014).	24
2.13	KNN classification accuracy and time taken image processing from source (Zhang et al.2018).	26
2.14	Effect of K value toward running cost from source (Zhang et al.2018).	26
2.15	Confusion Matrix plot from the source (Jason, 2020).	28
2.16	Accuracy formula for confusion matrix from the source (Ajay et al., 2020).	30

3.1	Flowchart for general methodology.	34
3.2	Flow work of the images processing and crack detection test process.	36
3.3	Sample images of crack concrete surface.	37
3.4	Sample images of crack concrete surface.	37
3.5	Sample images of non- crack concrete surface.	38
3.6	Sample images of non- crack concrete surface.	38
3.7	Crack image dataset folder.	39
3.8	Training and testing crack image datasets.	40
3.9	Training and testing non-crack image datasets.	41
3.10	Minor crack image.	42
3.11	Moderate crack image.	43
3.12	Severe crack image.	43
3.13	Non-crack image has hole on surface.	44
3.14	Non-crack image has rough surface.	44
3.15	Non-crack image has small object on it.	44
3.16	KNN visualisation graph.	55
3.17	Table of 2x2 confusion matrix from the source (Shin, 2020).	58
3.18	Formula for precision, recall and accuracy source from (DataQ, 2013).	59
3.19	Table of Confusion Matrix.	60
3.20	Random crack and non-crack dataset for classification test.	62
4.1	0.2 Greyscale intensity level result.	65
4.2	0.4 Greyscale intensity level result.	66
4.3	0.6 Greyscale intensity level result.	66
4.4	0.4 Greyscale intensity level result for non-crack image.	67
4.5	Crack length measurement for training dataset. 68	68
4.6	Confusion Matrix accuracy data result for K value 3.	69
4.7	KNN classification visualization result for K value 3.	70
4.8	Confusion Matrix accuracy data result for K value 5.	71
4.9	KNN classification visualization result for K value 5.	72
4.10	Confusion Matrix accuracy data result for K value 7.	73
4.11	KNN classification visualization result for K value 7.	74

4.12	Confusion Matrix accuracy data result for K value 3.	76
4.13	KNN classification visualization result for K value 3.	77
4.14	Confusion Matrix accuracy data result for K value 5.	78
4.15	KNN classification visualization result for K value 5.	79
4.16	Confusion Matrix accuracy data result for K value 7.	80
4.17	KNN classification visualization result for K value 7.	81
4.18	Confusion Matrix accuracy data result for high crack image dataset.	84
4.19	KNN classification visualization result for high crack image dataset.	85
4.20	Confusion Matrix accuracy data result for low crack image dataset.	87
4.21	KNN classification visualization result for low crack image dataset.	88
4.22	KNN classification visualisation for balance training dataset.	89
4.23	KNN classification visualisation for crack training dataset is high.	90
4.24	Graph of accuracy against the number of K value.	92
4.25	KNN clustering concept source from (Band, 2020).	93
4.26	Graph of accuracy comparison for different number of datasets for training process.	95
4.27	Confusion Matrix accuracy result for K value 3.	98
4.28	Formula for precision, recall and accuracy calculation source from (DataQ, 2013).	99
4.29	Correct severe crack dataset class prediction.	101
4.30	Correct minor crack dataset class prediction.	101
4.31	Correct moderate crack dataset class prediction.	101
4.32	Correct blur crack dataset class prediction.	102
4.33	Correct moderate crack dataset class prediction.	102
4.34	Correct blur crack dataset class prediction.	102
4.35	Correct obstacle crack dataset class prediction.	103
4.36	Correct moderate crack dataset class prediction.	103
4.37	Correct severe crack dataset class prediction.	103
4.38	Correct obstacle crack dataset class prediction.	104

4.39	Correct class prediction for obstacle non-crack dataset.	104
4.40	Correct class prediction for non-crack dataset.	104
4.41	Correct class prediction for obstacle non-crack dataset.	105
4.42	Correct class prediction for rough surface non-crack dataset.	105
4.43	Correct class prediction for smooth surface non-crack dataset.	105
4.44	Correct class prediction for smooth surface non-crack dataset.	106
4.45	Correct class prediction for rough surface non-crack dataset.	106
4.46	Correct class prediction for smooth surface non-crack dataset.	106
4.47	Correct class prediction for rough surface non-crack dataset.	107
4.48	Correct class prediction for obstacle non-crack dataset.	107



LIST OF TABLES

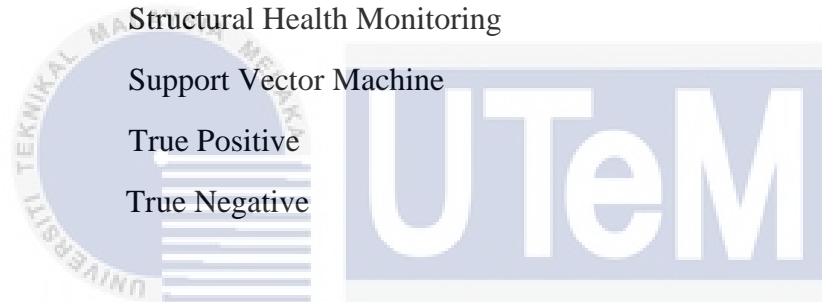
TABLE	TITTLE	PAGE
2.1	Average performance of KNN classification from the source (Jinho et al., 2012).	22
2.2	Accuracy result for medical images classification from the source (Rakesh & Khachane, 2012).	23
3.1	Images specification.	42
3.2	MATLAB command for image processing.	45
3.3	Crack greyscale image output after undergoes colour conversion process.	46
3.4	MATLAB Command for image binarizing.	47
3.5	MATLAB command for feature extraction.	49
3.6	The crack image binarizing and features extraction process output.	49
3.7	Crack length measurement for horizontal and vertical position.	51
3.8	Table for 80% training and 20% testing dataset.	53
3.9	Table for 90% training and 10% testing dataset.	53
3.10	MATLAB command for KNN algorithm.	54
3.11	MATLAB command for KNN visualisation.	57
3.12	MATLAB command for Confusion Matrix data tabulation.	60
3.13	Confusion Matrix calculation formula.	61
4.1	KNN classification accuracy data result for K value 3.	69
4.2	KNN classification accuracy data result for K value 5.	71
4.3	KNN classification accuracy data result for K value 7.	73
4.4	KNN classification accuracy data result for K value 3.	75
4.5	KNN classification accuracy data result for K value 5.	78
4.6	Table 4.6: KNN classification accuracy data result for K value 7.	80

4.7	Classification result for 800 crack dataset and 100 non-crack dataset.	83
4.8	Classification result for 100 crack dataset and 800 non-crack dataset.	86
4.9	Accuracy result when number of crack dataset is higher for training process.	94
4.10	Accuracy result data when too high crack dataset is used for training process.	96
4.11	Data of precision, recall and accuracy.	99



LIST OF ABBEREVATIONS

FP	False Positive
FN	False Negative
KNN	K- Nearest Neighbor
RGB	Red, Green and Blue
SHM	Structural Health Monitoring
SVM	Support Vector Machine
TP	True Positive
TN	True Negative



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHAPTER 1

INTRODUCTION

In this era of modernisation, the construction sector plays an essential role in country development. The necessary element that completed the construction sector is the concrete element which is the concrete structural part. Due to the rapid growth of superhigh buildings, larger sizes, and longer spans, the best concrete performance is now needed. The importance of the concrete construction necessitates an initial structural health monitoring examination toward the concrete surface. Concrete, according to Amlan and Devdas (2000), is a composite material made up mainly of a binding medium, such as a mixture of Portland cement and water, and embedded particles or pieces of aggregate, generally a mix of fine and coarse aggregate.

In comparison to others construction mediums such as natural stone or steel that had previously been used, concrete was employed in construction because it satisfied a broad range of performance criteria. Since the use of concrete in the construction industry is critical, inspecting the concrete's physical condition has become a top priority in order to ensure the best structure efficiency. The physical surface state of the concrete is used to determine whether or not cracks are present on the surface. If the sign was present, it meant that the structure needed to do a repairing process. The maintenance of concrete structures necessitates the presence of concrete surface cracks. Furthermore, seemingly insignificant cracks can develop and eventually trigger multiple structural failures.

The stability of a structure is an important consideration not only in the construction phase but also in the conservation phase. Visual inspection, which lacks detailed examination, is the current approach for inspecting the surface quality of concrete nowadays. As a result, a more effective crack classification method is required in order to determine whether any cracks have formed on the surface utilising computerised techniques. Thus, this report will present the technique for cracked concrete image detection and classification using the k-nearest neighbour algorithm method.

1.1 Background Study

In comparison to other materials, concrete now makes up the majority of infrastructure. One of the techniques used to determine the structural health of these structures is to look at a crack on the concrete's surface. Since the condition of a solid structure is generally quickly and directly determined by inspecting the surface crack, the inspection should be performed regularly to ensure sturdiness and protection over its life cycle. A concrete crack is a flaw in the surface of the concrete that shows up as a crack. One of the most common and significant forms of asphalt concrete crack is cracking. Cracking distress is generally categorised into three types: longitudinal, alligator, and transverse cracking.

Concrete structural maintenance is described as work done to maintain or prolong the service life of concrete before major restoration or full reconstruction is completed. It is graded as routine or preventive based on its purpose. Cracks are a significant source of concern for a structure's stability, longevity and serviceability. The reason for this is because when cracks develop and spread, they diminish the effective loading area, increasing stress and ultimately causing the concrete or other structures to collapse.

Cracking is unavoidable in all sorts of constructions, including concrete walls, since reinforced concrete constructions are still restricted and buildings degrade with time. Cracks in the concrete surface will allow another material to enter the structure, compromising the structure's integrity and efficiency. Surface cracks are vital signs of structural damage and stability in almost all forms of structures. Visual inspection of the structural elements is critical for detecting cracks and assessing the physical and functional conditions. However, many developing countries using manual crack detection methods to inspect building's structures by referring to Nhat.(2018). As a consequence, crack measurements and gathering or processing pertinent data would take longer and need more work.

In addition, manual inspection necessitates subjective judgements by inspectors and in terms of cost and reliability of the analysis, manual visual examination is inefficient. The appearance of cracks on the concrete surface indicates severe problems inside the structures. As a result, structure health monitoring (SHM) systems are needed to ensure the primary stability and execution of the structure for an extended period in order to avoid ruinous disappointment in the early stages. SHM is a technique for identifying cracks in building systems as well as a methodology for crack detection.

Therefore, a more efficient semi-automated fracture categorization analysis method is required. Hence the application of the k-nearest neighbour algorithm was explored in order to evaluate the surface condition of the cracked concrete such as cracked width and length. The K-Nearest Neighbour (KNN) algorithm was proposed in this report to identify the presence of cracks on the surface focused on building concrete structures. According to Zhang.(2016), this classifier method was chosen because it is straightforward, requires little data, and has been documented to be highly efficient and successful in solving classification problems.

1.2 Problem Statement

This project focus on crack detection and classification method toward the surface condition images of the concrete building structure. Since the safety of concrete condition become the priority toward the building structure, it is necessary to do a safety inspection toward the concrete structure for maintenance purpose. The current method for concrete maintenance inspection nowadays is manual inspection. Manual inspection consuming more time to analyse the concrete surface condition since the procedure is done by observation toward the concrete surface manually and need to be done by an experienced inspector.

Besides, the area that has a crack at the concrete surface might affect the building structure condition. This will make the inspection environment hazardous for an inspector to do measurements of cracks and record the data at the inspection site. Thus, we need a method that reduces the interaction of inspector and cracks area to classify the surface is crack or not.

Next, to do structural health inspection need many inspectors since most buildings nowadays have significant structures. Thus, limited numbers of experienced inspectors become one of the problems to perform the manual inspection. When the number of inspectors less compared to the inspection area, this will be affected the subjective judgment of inspectors since they cannot spend more time for analysis at a specific site.

Based on the problem statement, it is vital to do an alternative way for concrete crack detection and classification in order to reduce time consumption during the concrete surface inspection, to decrease the hazardous risk toward the inspector during the analysis process and finally to overcome the limited number of experienced inspectors to do maintenance inspection.

1.3 Objective

The objective in this study would be:

1. To study the suitable method for crack image processing process that can enhanced the present of crack clearly.
2. To develop a technique that detect and classify cracked and non-crack concrete image using K-Nearest Neighbour (KNN) algorithm method.
3. To evaluate the result of classifier and determine the crack image classification accuracy.



1.4 Scope of Project

The main scope of this research is to develop a technique that can detect crack at the concrete surface from images and categorised the image which is crack or non-crack categories through the implementation of the K-Nearest Neighbour (KNN) Algorithm method. The sources of crack concrete images were obtained from SDNET2018 only and the images sample are from building concrete wall structure. The project scopes are as follow.

1. Prepared and study the image data set for non-crack concrete and crack from SDNET 2018 for the proposed method.
2. Enhanced the images by using image processing program.
3. Develop the K-Nearest Neighbour (KNN) algorithm for crack detection and classification test by using MATLAB software. The detection process uses 80% training and 20% testing, and 90% training and 10% testing of datasets.
4. Analyse the accuracy of crack image classification result for different value of K which is 3,5,7 by using confusion matrix.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The second chapter organises the references in terms of theory and specifics concerning the project's scope. The primary goal of conducting a literature review is to gather information and compare previous crack detection and classification studies. This literature review is carried out by reading a variety of books, journals, published knowledge, and written materials pertaining to the type of detection and classification method. This literature review will be concentrating on the crack detection and classification method for concrete surface structure. The article review also will highlight the factor that causes the crack to occur and types of the crack. Other than that, the type of images that suitable for the image processing method also will be described in this chapter such as the image format and the quality. Next, the architecture step for crack detection based on image processing is presented here. The image processing technique that can be used to process the image in order to classify the presentation of crack will be discussed. The image classification algorithm which is K Nearest Neighbour (KNN) algorithm also will be present in this part. This algorithm will be used to classify the presence of crack toward the images. Lastly, the confusion matrix will be used to evaluate the accuracy of the categorized images. A confusion matrix also known as an error matrix is a quantitative approach to describing image categorization accuracy.

2.2 Factor of Cracking Concrete

Several causes, including thermal expansion, overloading, constraint, and chemical reactions, may cause cracking in concrete, according to Laroshe (2009). Besides, the external forces also will produce cracks, which range in size from extremely small interior microcracks to very massive external fissures. Cracking may occur immediately after casting, when the concrete is still malleable or after it has hardened. Internal or external factors can cause cracking in any instance. In both fresh and hardened concrete, cracking is frequently accompanied by volume change and moisture loss. Fresh concrete shrinkage is known as "plastic shrinkage," but hardened concrete shrinkage is known as "drying shrinkage." Concrete cracking is caused by a variety of variables, one of which being plastic shrinkage. "Plastic shrinkage" cracking is a term used to describe cracking caused by a volumetric change in the plastic state.

The rapid loss of water from fresh concrete is the most typical cause of plastic shrinkage. The circumstance that will cause a fracture is the evaporation of surface water from the surface of new concrete or suction of the sub-base or formwork immediately underneath the concrete. Rapid drying is the next aspect that contributes to the presence of cracks in concrete structures. The danger of cracking will be significantly increased if the slab dries quickly. A significant quantity of water is required for the chemical process that permits concrete to transition from a liquid or plastic state to a solid form. Hydration is a chemical process that happens days or weeks after the concrete has been poured. The reaction process will have a low efficiency if the drying step does not occur regularly. As a result, the concrete will begin to crack.

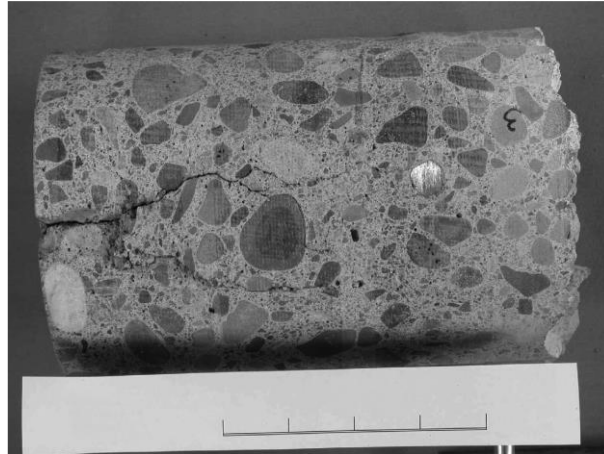


Figure 2.1: : Shrinkage cracking sample toward concrete structure from the source (Larosche, 2009) .

2.3 Crack Classification

According to Varsha (2019), there are two different types of concrete cracks: active and dormant. In active cracks, the direction, width and depth of the crack change with time, whereas in dormant cracks the direction, width, and depth remain constant. Both active and dormant fractures give access for moisture entry, which can lead to future harm if left unaddressed. Longitudinal cracks, transverse cracks, various fractures, crocodile cracks, and reflection cracks are some of the active cracks. Dormant cracks are incredibly fine in nature and they heal on their own over time. For example, micro cracks and thin cracks in the dormant crack. Crack classification is a method of applying machine learning algorithms to identify a particular crack type. Crack detection is the process of detecting or identifying the existence of a crack, while crack classification is the categorization of the fracture based on the feature extracted from the crack area.

Below is the type of crack sample that commonly occur at concrete structure.

1. Minor crack.

Minor cracks are very small or thin cracks that can be divided into three types: thin, microscopic, and line-like. Concrete bridges, underwater dams, polymers, autos, and planes are all susceptible to this type of crack. This crack is tiny in size and discontinuous.



Figure 2.2: Minor crack sample from the source (Kaish et al., 2018).

2. Moderate crack.

Moderate cracks are not as serious as severe cracks, thus remedial action is required. This fracture is commonly found in underwater dams and concrete structures, and it can be classified as medium, sealed, or severe. According to Pengfeishi (2016), the moderate crack sizes are larger than minor crack and hence discontinuities can be easy to analyse with image processing.



Figure 2.3: Moderate crack sample from the source (Kaish et al., 2018).

3. Severe crack.

Severe cracks are extremely large and dangerous, need prompt remedial action. This type of fracture can be found in a variety of places, including underwater dams, subway tunnels, bridges, pavements, concrete, and civil structures. Large fracture sizes and plainly identifiable cracks characterise severe cracks. Wenyu Zhang (2014) used morphological operations, thresholding operations, and KNN to detect and classify sub-way tunnel cracks. The accuracy percentage is 90% in this case. Severe cracks can be recognised even in the face of complex backgrounds such as noise, according to a study.

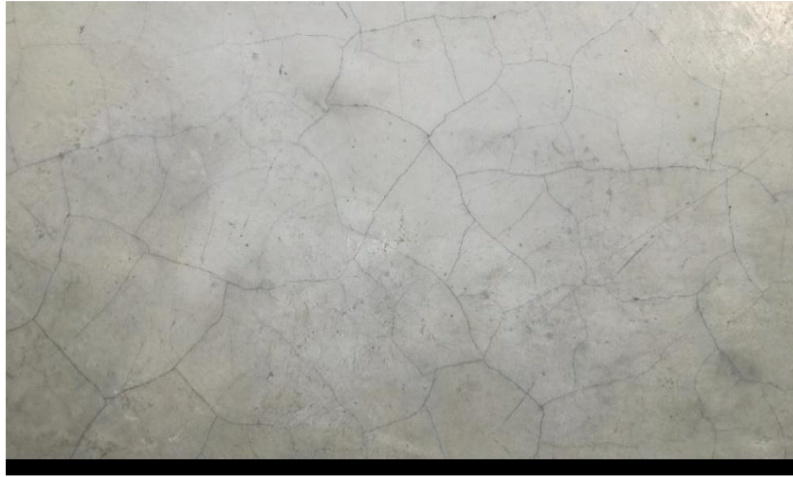


Figure 2.4: Severe crack sample from the source (Kaish et al., 2018).

2.4 Types of Images for Images Processing

A function which in two-dimension form, $F(x, y)$, were applied to create a figure, where x and y are spatial coordinates. The intensity of an image at any pair of coordinates (x, y) equals the amplitude of 'F' at that point. When the x , y and amplitude values of 'F' are finite, we call it a digital image. To put it another way, an image is a two-dimensional array made up of rows and columns. A digital image is made from of a small pieces numbers, each of which has a distinct value at a distinct location. Picture elements, image elements, and pixels are all terms used to describe these components. A pixel is the most common unit of measurement for the elements of a digital image. By referring to Varsha et al. (2019), the are three main image types that have been used for the image processing process. The image types are binary, greyscale and colour.

Binary pictures are the simplest kind of image, with just two possible values: black and white, or 0 and 1. A binary image is called a 1-bit image because each pixel is represented by just one binary integer. These kinds of images are typically utilised in situations where simply a general shape or contour is required. To create binary images from

grey-scale pictures, a threshold operation is performed, in which every pixel over the threshold value is transformed white ('1'), while those below it is rendered black ('0').

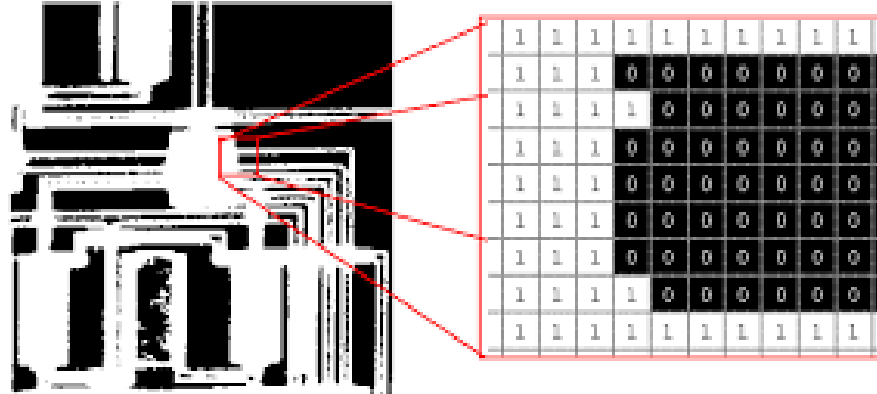


Figure 2.5: Binary image sample from the source (Amlan and Devdas, 2000).

The greyscale image comes next. Monochrome (one-color) images are the same as grey-scale images. They only have grey-level information and no colour data. The number of bits utilised for each pixel determines the number of different grey levels available. A typical grey-scale picture has 8 bits per pixel of data, which allows for 256 distinct grey levels. Medical imaging and astronomy both use pictures with 12 or 16 bits per pixel. When a small area of an image is expanded more bigger to detect details, these extra grey levels become useful.

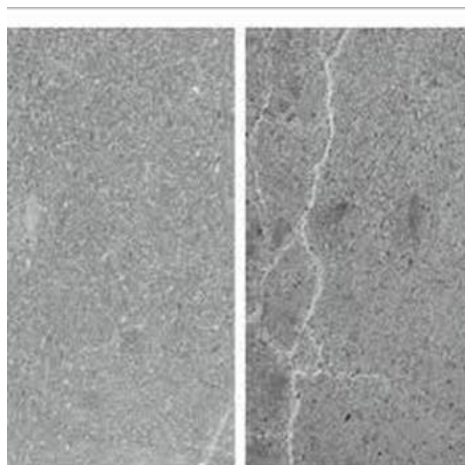


Figure 2.6: Greyscale image sample from the source (Varsha et al., 2019) .

Finally, there is a colour type image. Three-band monochrome image data may be used to represent colour images, with each band representing a different colour. The true information in the digital image data is the grey-level information in each spectral band. Common colour images are represented by the colours red, green, and blue (RGB images). If the 8-bit monochrome standard was utilised as a reference, the comparable colour image would contain 24-bits/pixel (8-bits for each of the three colour bands red, green and blue).



Figure 2.7: RGB image sample source from (Maurits, 2020).

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2.5 Architecture Process

The fundamental architecture for crack detection via image processing is presented in this section. The primary benefit of image-based crack detection, according to Arun and Sumathi (2017), is that it provides more accurate findings than previous manual approaches. The image size determines the complexity of the fracture detecting technique. The image sample, which has a resolution of more than 10 megapixels, may be used to capture detailed photos of concrete surfaces.

The main architectural stages for crack identification based on image processing are shown below.

1. Image Acquisition.

This procedure might be as straightforward as being given an image which is a digital image. The majority of the process involves scaling and colour conversion from rgb to grey or vice versa.

2. Image Enhancement.

This is the straightforward and attractive approach in the world of image processing. It's also subjective, and it's used to extract some of an image's hidden elements.

3. Image Restoration.

The approach aims to make an image more attractive, but it is objective since it is based on a statistical or probabilistic model of image deterioration.

4. Colour Image Processing.

The colour models for pseudo-colour and full-colour image processing may be used in digital image processing.

5. Morphological Processing.

The process is focused with image component extraction methods that may be used to represent and describe shapes.

6. Segmentation Procedure.

Part of the procedure involves partitioning a picture into its fundamental components or objects. Autonomous segmentation is the most demanding element of image processing.

Furthermore, Yiyang et al. (2019) proposed a digital image processing-based crack detection algorithm. Pre-processing, image segmentation, and feature extraction were used to gather information about the crack image. The Threshold method of segmentation was used after smoothing the acceptable input image. To analyse the image, the roundness index's size and perimeter were calculated. Using the comparison method, the presence of the crack in the image was then determined.

2.6 Image Processing Method

Pre-processing, detection, and classification are the three main processes in crack detection. According to the literature, smoothing and filtering methods were utilised during the pre-processing phase, and detection was carried out using various approaches, including the Otsu method, statistical approach, threshold technique, segmentation technique, and classification using deep learning algorithms.

Ahmed et al. (2017) uses a three-step process in their research. They aimed to locate the crack's surface. The image was first converted to grayscale, and then the fractures were detected using Sobel's filter. The images were then divided into the foreground and background categories. After categorising the noise, Sobel's filtering was used to remove it. Following that, the Otsu technique was applied to detect cracks. They used a real dataset with an accuracy of above 85%.

For identifying fractures in concrete structure images, Talab et al. (2015) suggested a unique image processing approach. This approach consists of three phases. To identify fractures, convert the image to grayscale using the picture's edge, and then develop the image using Sobel's filter. After that, an appropriate threshold binary image of the pixel is used to classify the foreground and background images. After the images were categorised, Sobel's

filtering was employed to eliminate any leftover noise. After the image was heavily filtered, cracks were found using Otsu's approach.

Koch and Brilakis (2011) indicated that colour information, namely RGB values, is not required while executing the image segmentation procedure regarding fault detection in pavement surface images. However, in this study's first selection, the RGB value will be the significant threshold value. The crack detection model and the pothole detection model are vastly different. This procedure aims to convert colour images to grayscale. Koch and Brilakis (2011) also implemented a 5 by 5-pixel median filter to decrease noise generated by hardware during image or video capture. In this research, the median filter approach is applied in the second step of the crack identification model. To remove the influence point or patterns, an 11 by 11 filter is utilised instead of a 5 by 5 filter.

By referring Varsha et.al (2019), to increase the present of crack for crack features extraction, there are several methods that the writer proposes. First, the RGB images need to be converted into greyscale image. The greyscale image was converted into binarize image after the greyscale intensity is adjusted into specified amount. For the features extraction, Canny edges feature extraction is used to make the present of crack become clearer in order to be used for crack detection process. Figure 2.8 is the output result from the study.

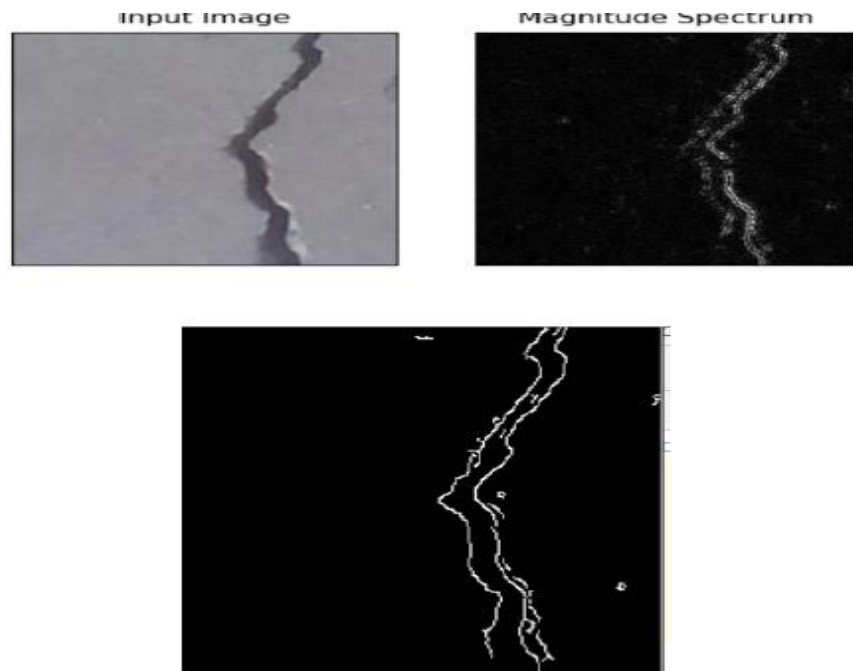


Figure 2.8: Crack image feature extraction source from (Varsha et.al 2019).

2.6.1 Feature Extraction

Feature extraction is used in machine learning and image processing to create derived values (features) that are meant to be useful and non-redundant. As a result, the algorithm image processing can extract the image information more correctly and, in certain situations, lead to improved human interpretations. The process of analyzing image important information such as pixel number is known as feature extraction. The feature extraction process also reduces the number of resources needed to describe a huge amount of data. One of the common issues for image processing is the data analysis when the information of the images is too high. A large number of image information necessitates a lot of memory and processing capacity, and it can also lead a classification algorithm to overfit to training examples and fail to generalize to new samples. Feature extraction is a broad phrase that refers to strategies for creating combinations of variables to get around these issues while still accurately representing the data.

Based on the literature review for the image processing method, most of the feature's extraction implemented to extract the crack information from the image after performing the image processing is Sobel and Canny features extraction. An article from Ali & Clausi (2001) is referred to analyze the effectiveness of these two feature extraction methods. This article tests both methods by extracting the presence of crack for wall structure obtained by using the remote sensing method. The result of the features extraction for this method is analysed based on the correct presence of crack structure and the number of noises in the images. Both images undergo the same images processing which is one thresholding process. For the result, the presence of crack using the Canny method is more precise than Sobel.

Next, the Sobel images feature extraction result shows that the images have higher noises than the Canny method. Thus, the writer concludes that the Canny method is more accurate than Sobel in terms of feature extraction, which makes cracks clearer and less noises image output. This because the Canny method have a greater number of thresholding process than Sobel which is 2. The thresholding process is the process that replaces and improve the image pixel color to black or white pixel if the image intensity is less than some fixed constant value. From the article, the presence of noises in the Canny method result is low compared to Sobel because, in the Canny method, Gaussian filter is used to improve and smooth the binarize image by adjusting the contrast and brightness of the image. Thus, the noises of the images will be reduced since the contrast adjusting process will make the image background become blur. While for Sobel, the method contains only one threshold process that is sensitive toward the image's noises.

This method required a better images processing method to perform the feature extraction process and the do not use Gaussian filter to smooth the process image. For the crack detection project, the suitable feature extraction that had been applied in this project is by using the Canny method by referring to the previous comparison data. This is because

crack images containing many noises and crack criteria need a highly efficient feature extraction method to enhance the presence of a crack in the images. Figure 2.9 and 2.10 are the comparison result for both methods.

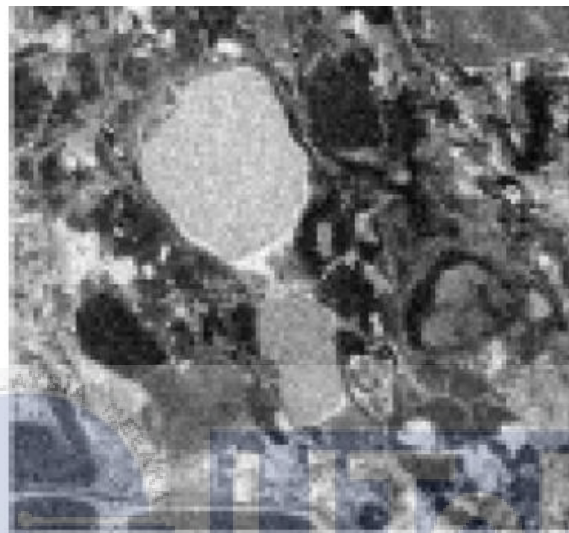


Figure 2.9: Feature extraction by using Sobel method from sources (Ali & Clausi, 2001).



Figure 2.10: Feature extraction by using Canny method from sources (Ali & Clausi, 2001).

2.7 Classification Algorithm

2.7.1 K Nearest Neighbour (KNN) Algorithm

The KNN algorithm, according to Alkhatib et al. (2013), is a machine learning approach that is considered a lazy learning algorithm with a low computation cost and is very straightforward to implement. As a consequence, instead of using a function or model to interpret the data, the algorithm examines the data that is most similar to the required estimates. According to Imandoust and Bolandraftar,(2013), when using the KNN technique, the data is separated into two parts: training data, on which the algorithm bases its predictions, and test data, on which the programme makes predictions.

The method is used to forecast the test data that comprises of the value. After partitioning the training data into vectors, one of many ways is employed to compute the distance between the test data and its neighbour. The weighted Euclidean distance, which is the closest distance between the two components, is the most common approach. This may be seen in the image below, where the Euclidean distance is utilised to identify B's nearest neighbour.

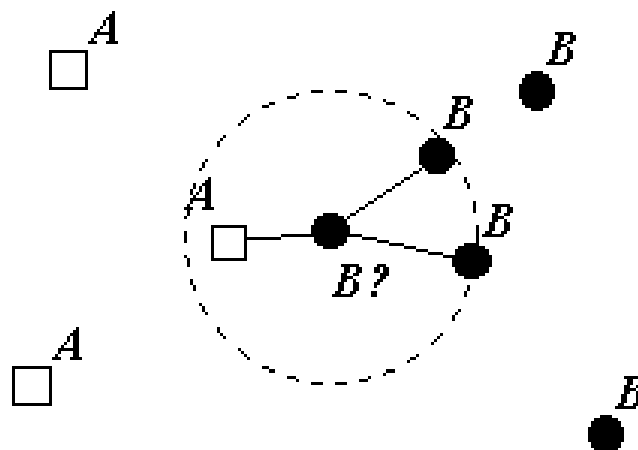


Figure 2.11: KNN algorithm classification concept from the source (Imandoust & Bolandraftar, (2013).

One of the main advantages of the KNN approach, according to Jinho et al. (2012), is that it works well with multi-modal classes since its conclusion is based on a small neighbourhood of related items. As a consequence, the technique may deliver correct answers even if the target class is multi-modal. However, one of the KNN algorithm's major flaws is that it computes similarities equally for all characteristics. This may lead to classification mistakes, particularly if there are only a few traits that are suitable for classification. They use the KNN algorithm classification approach to execute a classification test based on their research. The study's goal is to determine the image categorization accuracy percentage. The MATLAB software was utilised as the primary computer language. The data collection contains 3505 images and is divided into four categories: aeroplanes, cars, faces, and motorcycles. To increase variation within each trial, the testing and training images were separated into a 1:5 ratio. The classification results were presented as a confusion matrix table, which may be used to compare classifier performance across classes.

Table 2.1: Average performance of KNN classification from the source (Jinho et al., 2012).

	Airplanes	Cars	Faces	Motorbikes
Airplanes	87.24%	0.09%	8.47%	4.19%
Cars	5.28%	61.1%	7.79%	25.80%
Faces	5.57%	0%	87.74%	6.69%
Motorbikes	5.56%	0%	10.05%	84.39%

The classifier's average performance for each class, with the actual class on the left and the predicted class on the right. The diagonally coloured boxes indicate the percent of correctly categorised images. The overall classifier with the best K value which is 5 had an average accuracy of 78.03 percent.

For KNN accuracy comparison, Rakesh & Khachane, (2012) had performed 3 types of image classification for automatic medical image classification and abnormality. The detection method that had been used is K Nearest Neighbour (KNN) with the K value 5, Support Vector Machine (SVM) linear kernel classifier and Support Vector Machine (SVM) RBF kernel classifier. The number of datasets is 26 normal images and 25 abnormal medical images. From the accuracy result, the KNN algorithm classification method has the highest accuracy compared to another method which is 80% accuracy. The accuracy result was stated as below.

Table 2.2: Accuracy result for medical images classification from the source (Rakesh & Khachane, 2012).

Evaluation Test Classifier	Sensitivity (%)	Specificity (%)	Accuracy (%)
KNN Classifier	80	81	80
SVM with linear kernel Classifier	77	56	67
SVM with RBF kernel Classifier	80	56	69

By referring to the accuracy result, KNN is the best classification method applied in this study because it has a high accuracy level compared to other classification methods.

For the number of K values that will be used in the KNN algorithm, Chih et al., (2014) did an analysis on the impact of different values of K toward classification accuracy. The dataset that had been used in the study is the classification of Parkinson dataset. There are 13 number of K values that were tested, and the classification accuracy was recorded. Figure 2.14 is the result of classification accuracy from the research.

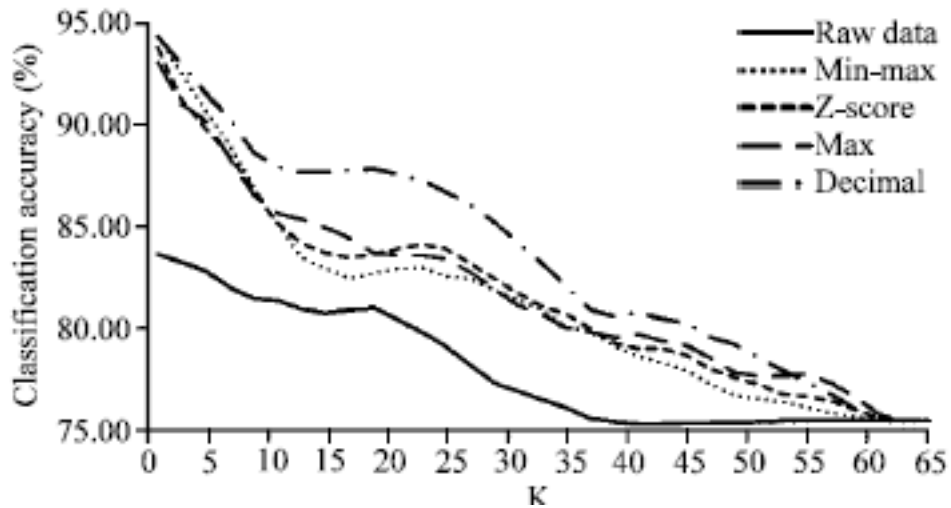


Figure 2.12: Classification accuracy different value of K source from (Chih et al., 2014).

From the classification result, the classification accuracy decreases as the value of K increases. When the value of K increases, the number of the close nearest neighbour dataset will be high. When the number of nearest neighbours is too high that will be used for class prediction, the actual class prediction for testing dataset accuracy will be drop because the reference data for class prediction is too many. Thus, a suitable number of K values is essential for class prediction in this study.

2.7.2 K Nearest Neighbour (KNN) Algorithm Training Dataset

For KNN images classification, the number of training datasets plays an important role. This is because the algorithm has less information about the images when the number of datasets used for training is less. This will reduce the testing accuracy since the algorithm has less information about the image features such as crack information by training images to be compared and analyzed with testing images during the testing process. Training image is the image that is used for algorithm training. The algorithm will identify important

information during the training, such as the number of black and white pixels, images sizes, images brightness, and contrast.

The training output is used to compare the testing data information in order to classify the testing data according to its category. The testing data is the data used during classification testing. The number of testing data is usually randomly selected to test the classification accuracy. By referring to Zhang et al. (2018), the classification accuracy is affected when the different number of trainings for each class is used to train the algorithm. From the study made by Zhang et al. (2018), they performed classification testing by using 10 categories of the dataset obtained from the UCI Repository of Machine Learning Database. The study aims to observe the testing accuracy when the different number of datasets for the different classes is used training process. The number of K values used for this study is 5. For testing datasets, 10 random images were chosen from each class for testing to classify their class by using KNN algorithm. The training dataset category is Madelon, LSVT, CNAE, Gisette, Hill, Libras, Dbworld, and Arcene.

From the experiment, the average accuracy obtained is 72.9%. From the article, the lowest dataset for training which is 90 show the lowest classification accuracy that is 56.1%. The highest accuracy is 5000 training datasets which are 86.0% correct class prediction. The accuracy decreases when 10000 training datasets use to train the algorithm. Then, the accuracy is 74.6%. When the number of training datasets is too much, the algorithm will become overtrained due to a lot of information of dataset gain per one process. The effect of overtraining will make the algorithm produce a training pattern that recreates the old dataset information pattern and extract wrong information from images. This will make the testing image classification accuracy become drop. Instead of overtraining, too high a training dataset will make the algorithm take more time to process. Thus, for this project, only two dataset classes will be used. The class is crack and non-crack. The quantity will be finalized

in methodology for the number of training and testing datasets. The result of accuracy testing classification for Zhang et al. (2018) is shown in Figure 2.15.

Dataset (#(features))	kNN
Madelon (500)	0.691 / 0.27
LSVT (310)	0.780 / 0.05
CNAE (856)	0.762 / 0.18
Gisette (5000)	0.860 / 0.22
Hill (100)	0.621 / 0.04
Libras (90)	0.561 / 0.03
DBworld (4702)	0.702 / 0.03
Arcene (10000)	0.746 / 0.07
Musk (168)	0.836 / 0.15
Arrhythmia (279)	0.736 / 0.13
AVERAGE	0.729 / 0.12

Figure 2.13: KNN classification accuracy and time taken image processing from source (Zhang et al.2018).

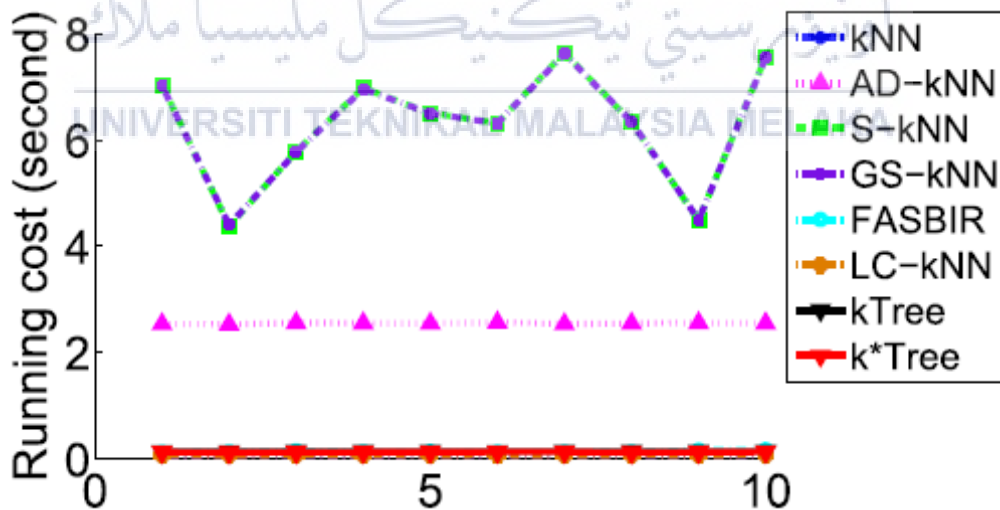


Figure 2.14: effect of K value toward running cost from source (Zhang et al.2018).

2.8 Classification Analysis

2.8.1 Confusion Matrix

The confusion matrix is the classification analysis tool that had been propose to employ in this research. Using Jason. (2020) statement as an example, the findings of the categorization problem prediction are summarised in a confusion matrix. Count values are used to sum and break down the number of accurate and wrong predictions by class. The count values to sum and number of the accuracy of class prediction are the essential data to perform confusion matrix. When producing predictions, the classification model gets confused, as seen by the confusion matrix.

A confusion matrix also known as an error matrix is a quantitative approach to describing image categorization accuracy. It is a table that displays how the categorization result and a reference image relate.

The rows shows the anticipated class (Output Class) while the genuine class (Target Class) is shown by the column , for the confusion matrix data plot according to Vijay and Bala (2019). The diagonal cells correspond to observations that have been properly categorised. The observations that were incorrectly classified are shown by the off-diagonal cells. The number of observations as well as the percentage of the total number of observations are shown in each cell.

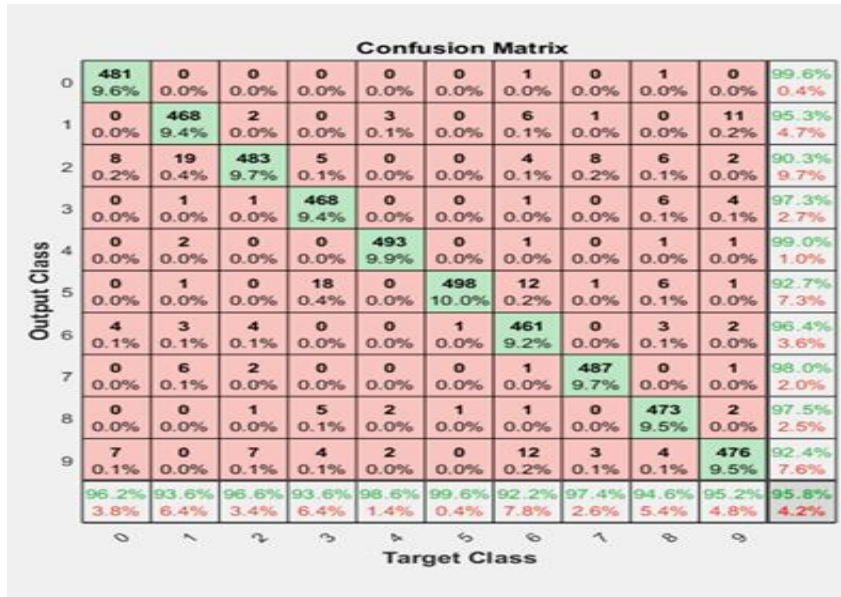


Figure 2.15: Confusion Matrix plot from the source (Jason, 2020).

The percentages of all the cases anticipated to belong to each class that are properly and incorrectly identified are presented in the column on the far right of the figure, according to Fujita et al. (2020). The accuracy or positive predictive value and false discovery rate are two often used measurements. The percentages of all the instances belonging to each class that are correctly and incorrectly classified are provided in the table's bottom row. The recall (or true positive rate) and false negative rate are two often used measures. In the cell at the bottom right of the figure, the total accuracy is shown.

To plot the True class labels which are true positive rate, the criteria are shown as the following:

- When each element provides the class label of a single observation, the vector is called a categorical vector. There must be the same number of elements in both the outputs and targets parameters. Plot confusion shows all underlying classes if the category vectors identify them, even if some underlying classes contain no observations. Both parameters must specify the same underlying categories in the same order when they are ordinal categorical vectors.

- The number of classes is N , and the number of observations is M in this N -by- M matrix. Each matrix column must be formatted as one-of- N (one-hot), with one element equal to 1 indicating the true label and all other elements equal to 0.

For the Predicted class labels, several criteria must be followed before the data is plotted.

Below are the Predicted class label criteria:

- A categorical vector with each element representing the class label of a single observation. The outputs and targets arguments must each have the same number of items. Plot confusion shows all underlying classes if the category vectors identify them, even if some underlying classes contain no observations. Both parameters must specify the same underlying categories in the same order when they are ordinal categorical vectors.
- The number of classes is N , and the number of observations is M in this N -by- M matrix. Each matrix column may be in one-of- N (one-hot) format, with a single element equal to 1 indicating the anticipated label, or in the form of one-to-one probabilities.

Confusion matrices are used to show counts based on expected and actual values, according to Ajay et al. (2020). The output "TN" stands for True Negative and it shows how many negative samples were properly detected. Similarly, "TP" stands for True Positive, which refers to the number of positive cases that have been accurately detected. The terms "FP" and "FN" stand for False Positive and False Negative values, respectively. "FP" stands for False Positive value, which is the number of actual negative examples classified as positive, and "FN" stands for False Negative value, which is the number of actual positive examples classified as negative. Accuracy is one of the most widely utilised criteria while performing categorization. The following formula is used to calculate the accuracy of a model through a confusion matrix plot.

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

Figure 2.16: Accuracy formula for confusion matrix from the source (Ajay et al., 2020).

When working with imbalanced datasets, classification accuracy may be misleading; nonetheless, there are a variety of metrics based on the confusion matrix that may be used to evaluate performance. In order to get the best result from the confusion matrix, users must provide both real and anticipated values to the function.



CHAPTER 3

METHODOLOGY

3.1 Introduction

The methods for crack identification and classification utilising the KNN algorithm approach in great depth were discussed in this part. The previous chapter provides a thorough assessment of the literature as well as key information. Then, methods proceeding to the next step will be done. This chapter is critical in ensuring that all progress is made systematically within the project timetable and that the project's objective is met. This chapter begins with the study and image selection from SDNET2018 for the image processing process. In this part, the number of images that will be used as stated according to their category. Hence, image processing and feature extraction were described in this part. The process was done by using MATLAB software. The image processing process involved the image segmentation process and image colour conversion, while the feature extraction process involved the canny method to identify the edges in the images. Besides, this chapter also described the crack classification method that will be used. After the detection and classification process was done, the result will be evaluated by using a confusion matrix in order to measure the classification accuracy.

3.2 General Methodology

This section explains on how this project require to be carry out to accomplish the objectives of this project such as identify, process with analyse data and information smoothly.

The actions that need to be conduct in the work are listed as below:

1. Suitable Objective, Problem Statement and Scope.

Study and understand the objective, problem statement and scope of the project before the classification test started.

2. Literature Review.

Journals, articles, magazines or any sources that provides information regarding the work are reviewed properly.

3. Preparation of crack concrete images for image processing. To ensure smooth image processing, the image must be devoid of any obstructions like shadows, impurities between the crack, or other superfluous factors might cause interruption for the image to be process.

4. Image Processing and Crack Detection Simulation.

Image processing method are applied to improve the quality of the images and this will make crack detection process become more uncomplicated. MATLAB Image Processing Toolbox was used to apply the image processing method. The K-Nearest Neighbour Algorithm will be implemented during classification process.

5. Data Analysis.

The accuracy on classification of cracked and not cracked image were evaluated by using confusion matrix. The result and problem during test were discuss in this part.

6. Report writing.

The research will be written in a report at the end of this work.



The methodology of this study is summarized in the flow chart as shown in figure

3.1.

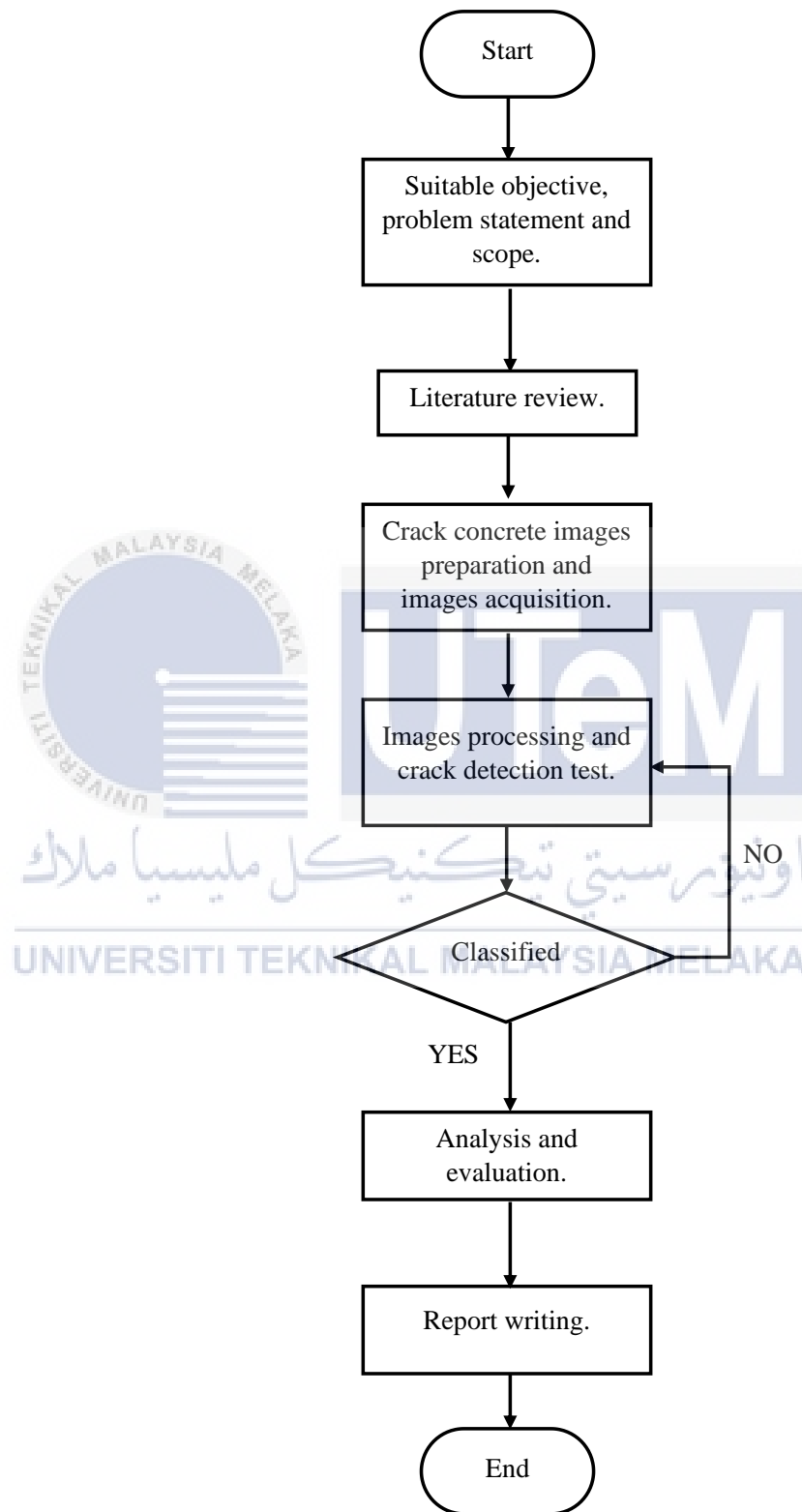


Figure 3.1: Flowchart for general methodology.

In order to make sure the crack image detection and classification process get the best result, there are several procedure need to be done during image processing and crack detection test process. The procedure for the test process consists of five crucial steps, which is images preparation from SDNET2018, images processing, feature extraction, classification using the KNN algorithm and lastly, result evaluation. The relationship of this flow work and general methodology is that this flow work describes the detailed procedure for images processing and crack detection test. In comparison, general methodology describes the crack detection process generally. Figure 3.2 is the flow work of images processing and crack detection test process.



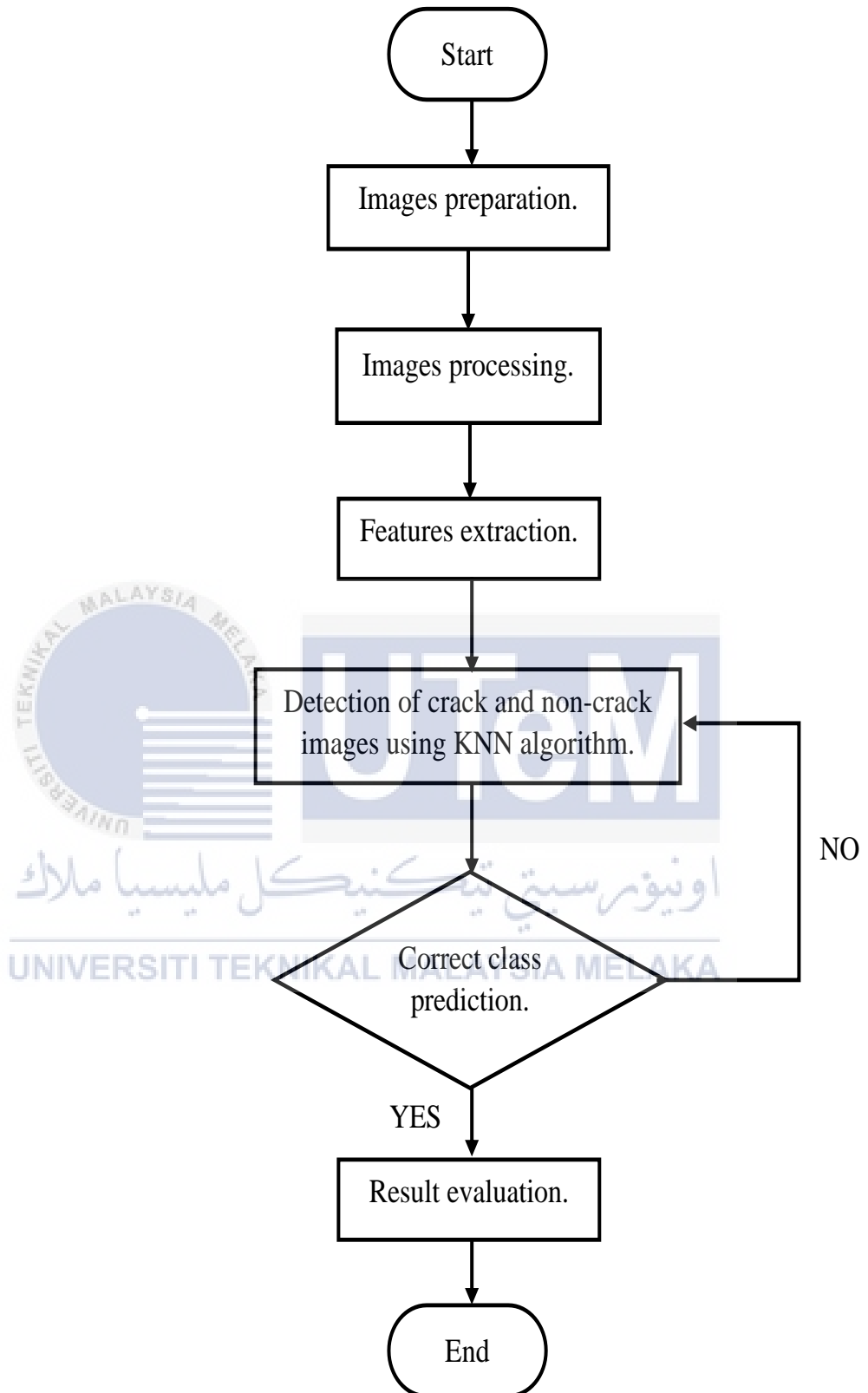


Figure 3.2: Flow work of the images processing and crack detection test process.

3.3 Image Preparation

In this process, the images were used from SDNET 2018 database. Two types of images were chosen from this database which is crack and non-crack. The image should show the cracks and devoid obstacles like shadows, impurities between the crack and other undesirable factors. The best crack detection result cannot be achieved when the images processing is interrupted by the mentioned obstacles. Below is the sample crack and non-cracked from SDNET2018 that used in this work.



Figure 3.3: Sample images of crack concrete surface.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA



Figure 3.4: Sample images of crack concrete surface.



Figure 3.5: Sample images of non- crack concrete surface.

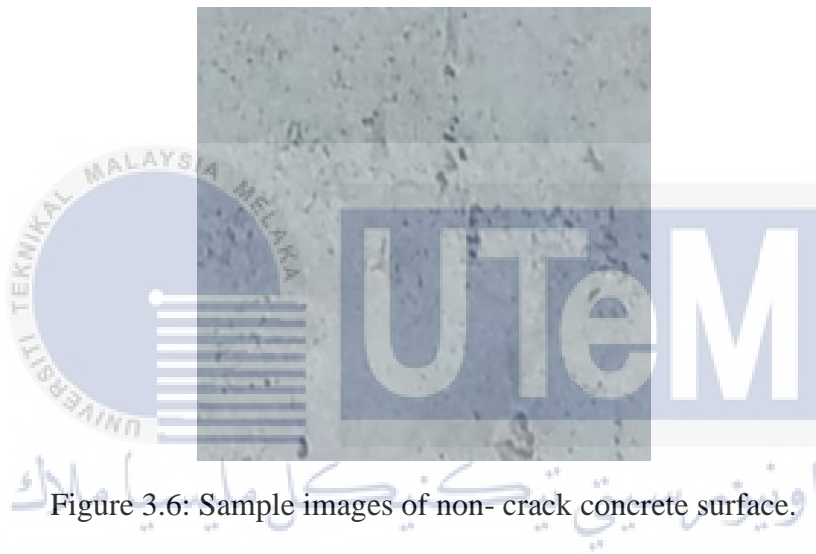


Figure 3.6: Sample images of non- crack concrete surface.

3.3.1 Images Quantity

The total number of images used in this work were grouped into two different categories of images quantity. The group are 200 images that contain 100 crack and 100 non-crack images. These images were categorized for KNN algorithm training purposes. For the testing process, different percentage of image numbers is used. First is 20% from 200 datasets is used and the next is 10% from 200 datasets is used. "Training datasets" is a dataset used to train the KNN algorithm to analyze the crack and non-crack information such as white and black pixels in the images. This will make the algorithm recognize the datasets criteria. While "testing datasets" is the dataset used to test the algorithm's ability to classify the images is cracked or not.

The ability to classify the crack images class is measured by the correct classification accuracy. The number of training and testing can be the same or different amount. Two different amounts of dataset training will be used to observe the highest testing classification accuracy for this work.

After the images testing accuracy achieved more than 70% correct classification, the different number of datasets for training is used to improve the testing accuracy. The number of datasets is 800 crack images and 100 non-crack images. The accuracy of correct classification is observed when the crack dataset is trained more than the non-crack dataset. For the testing process, 10% of the datasets are used for testing purposes.

When the high classification accuracy is achieved, 20 random images that contain 10 crack images and 10 non-crack images are used for images classification testing. This testing is performed to observe the algorithm classification accuracy when the datasets are combined in one file.

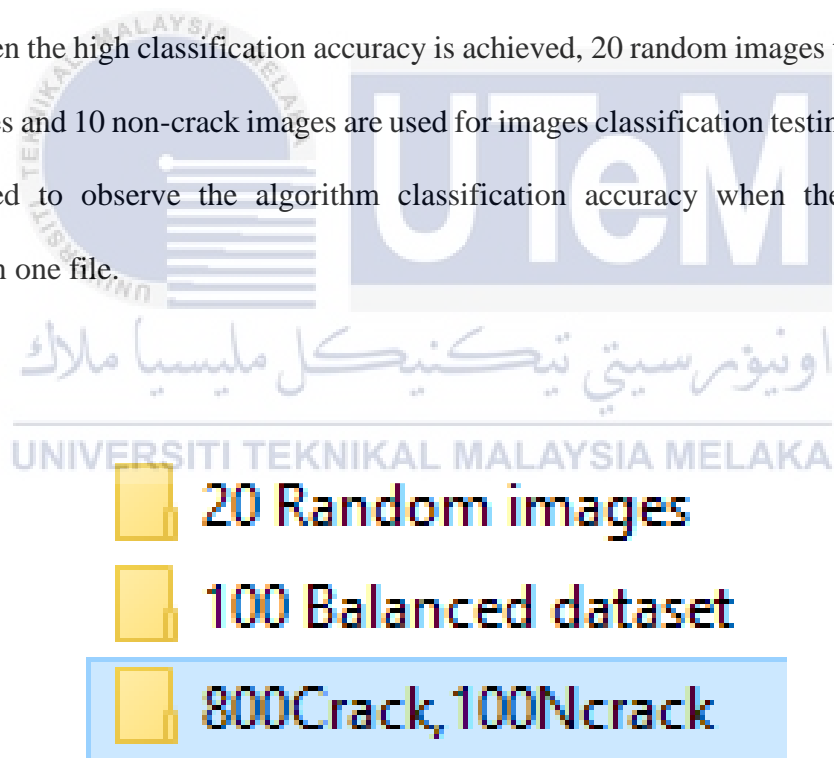


Figure 3.7: Crack image dataset folder.



Figure 3.8: Training and testing crack image datasets.



Figure 3.9: Training and testing non-crack image datasets.

3.3.2 Images Specification

The crack images sample were collected from SDNET2018. The images category that had been used in this study is wall concrete images. The table of images specifications is shown in table 3.1.

Table 3.1: Images specification.

Specification	Description
Image Format	JPG
Average Image File Size	4.5 KB
Width Dimension	256 Pixels
Height Dimension	256 Pixels
Horizontal Resolution	96 Dpi
Vertical Resolution	96 Dpi
Bit Depth	24

3.3.3 Type of Crack

In this study, there are three types of crack images used for the training and testing. The crack type is a minor crack, moderate crack and severe crack. Below is the example of crack image samples, that contain crack data used for the detection process. All the datasets are obtained from SDNET2018 database.

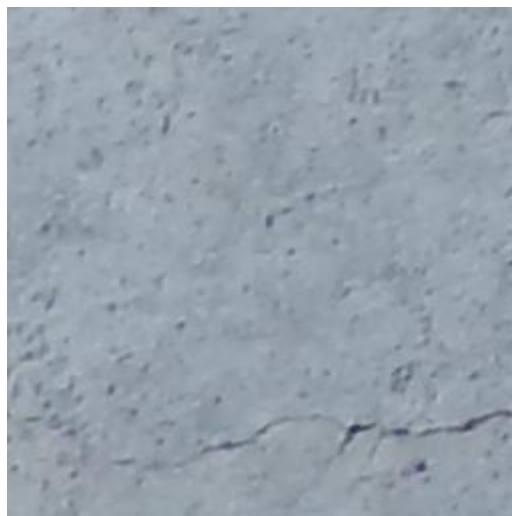


Figure 3.10: Minor crack image.



Figure 3.11: Moderate crack image.



Figure 3.12: Severe crack image.

For non-crack images, the images have three different concrete surfaces and has obstacle on the surface, such as having a hole, a rough surface and having a small object on it. Below is the example of non-crack images that is used for algorithm training and testing.

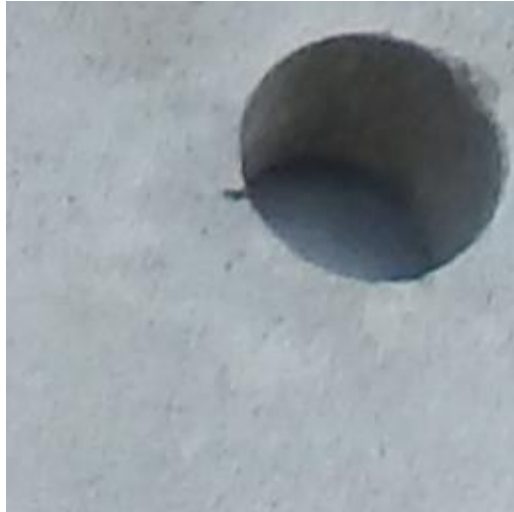


Figure 3.13: Non-crack image has hole on surface.



Figure 3.14: Non-crack image has rough surface.



Figure 3.15: Non-crack image has small object on it.

3.4 Image Processing

The image processing stage is vital since it contains numerous phases that must be completed before achieving a result. Because of the image's size, non-uniform lighting, or the image's low contrast, basic image processing was unable to deliver an accurate reading to the picture that we wished to analyse. Because of this difficulty, the existence of fractures on the concrete surface cannot be identified or identified. Therefore, image processing will be applied to enhance the image to improve the presence of a crack in the image. MATLAB Image Processing Toolbox was utilised to implement the image processing algorithms. This project applied image processing phases such as RGB to grayscale image conversion and image binarizing process.

3.4.1 Greyscale Image Conversion

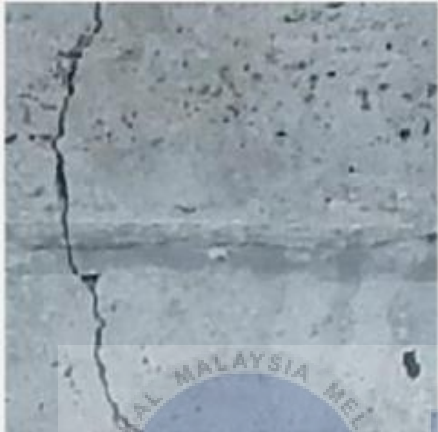
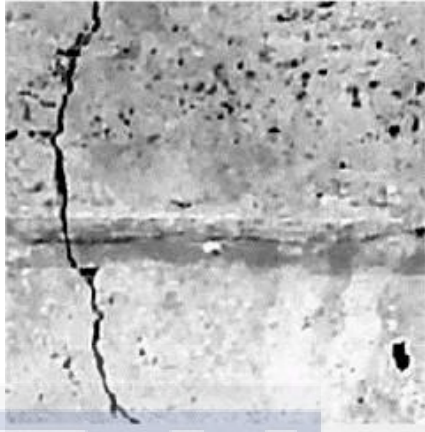
The image processing begins with the uploading of images to the MATLAB application. The "imread" is used to read inserted into the MATLAB working script. This command displayed the original image in the working script. Next, the original image's colour is converted into a greyscale format using the "rgb2gray" command. This command changes the original image colour format into a grey image format. The greyscale level during this process is in the range from 0 that represents black to 256 pixel that represents white. The MATLAB command that had been used for image colour conversion is stated below.

Table 3.2: MATLAB command for image processing.

Matlab Command	Function
<code>I = imread</code>	To read inserted image in the program.
<code>Igray_s = rgb2gray (Istrech)</code>	Convert RGB images into greyscale image.
<code>Ithres = imbinarize(Igray_s,level)</code>	Convert greyscale images into binarize images.

Table 3.3 is the table of crack greyscale image output after undergoes colour conversion process that had been done in this work.

Table 3.3: Crack greyscale image output after undergoes colour conversion process.

Original Crack Image	Greyscale Image Output
	

3.4.2 Image Binarizing Process

After the RGB image undergoes the greyscale color conversion process, the greyscale images are converted into a binary image using a specific grey intensity level. Binary pictures are images with just two potential intensity values for each pixel. They are usually shown in black and white. During this process, the image pixel is adjusted by replacing all pixels in the input image with luminance greater than level with the value 1 that refers to white pixel and replacing all other pixels with the value 0 that refer to black. The greyscale intensity level is in the range between 0 to 1. For this project, the images were tested using three greyscale intensity levels, which are 0.2, 0.4 and 0.6. Below is the command and greyscale level that has been used in this training and testing process.

Table 3.4: MATLAB Command for image binarizing.

MATLAB Command
level = 0.4; Ithres = imbinarize(Igray_s,level)

The purpose of this different amount is to analyse the best amount of grey intensity level for the training and testing process that enhances the presence of a crack in the images. The correct greyscale intensity level will reduce the images noises and produce a clear presence of crack after undergoing Canny features extraction. The Canny features extraction that uses two thresholding processes and a Gaussian filter can improve the presence of crack when the binarize image has a clear white pixel present that has less images noises. The output for the image binary process in this work is the grey image is converted into black and white image format.

3.5 Feature Extraction

This approach is used to calculate classification tasks for normalised images and is described as the process of converting input data into a format that a classifier can comprehend. For the recognition process, the image must be transformed into relevant characteristics and the classifier must then be utilized to identify the item. Throughout this procedure, the edges on the concrete images were improved by using the Canny technique. After the binary images are produced, the algorithm will read the images information first, such as the number of black and white pixels and the contrast and brightness of all input images. Then, the feature extraction is made toward the binary images. The Canny feature extraction obtained structural information from the binarizing image while greatly minimizing the quantity of data that must be analysed. The Canny method is used in this work because it improves the presence of a crack in the images and reduces the image's

noises and disruption, such as impurities at crack images. The canny method uses a Gaussian filter to reduce blurring and detect small, crisp lines from binarizing images.

This method is much better than Sobel feature extraction that is sensitive toward the images noises since the method does not apply any image filter according to Kim (2013). Next, the Canny method used two threshold processes toward the binarizing images to determine between light and dark edges and stated light edges in the output if they are related to dark edges. This makes the presence of cracks in the images clearer and improves the crack detection accuracy. For this work, after the greyscale images undergo the greyscale intensity level adjustment in binarizing process, the images will undergo a features extraction process that improves the crack image quality and enhance the presence of crack better. Two thresholding processes were applied toward the binarizing images. This will improve the presence of a crack line in the image since the pixel colour of black and white is improved two times.

After that, the Canny method will apply a Gaussian filter toward the images after the thresholding process that reduces the images' noises by blurring the image's background and improving the pixel gradient. This will make the crack line clearer. The suitable greyscale intensity level affected the presence of crack after feature extraction. The thresholding process will replace the image pixel data with other pixel data such as black when the wrong intensity level is chosen during the image binarizing process. By referring to Table 3.6, the presence of crack line is much better and images noises are reduced when the image undergoes feature extraction compared to image binarizing process output. After the image undergoes complete feature extraction, the algorithm will analyse the number of white pixels and black pixels, the image contrast and brightness. This data will be used as referenced data for KNN training and testing.

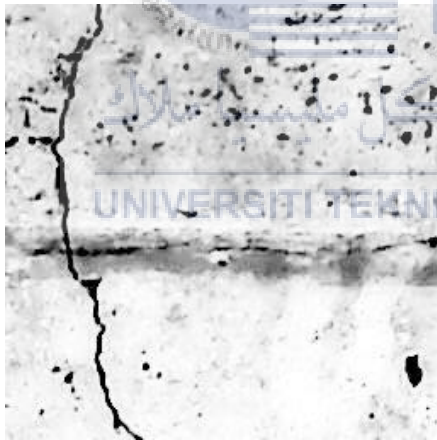

The final output from the feature extraction process is the image noises is reduced and crack line at the image become clearer. This process is applied for datasets training and testing process. Below is the Canny command that has been use in this project.

Table 3.5: MATLAB command for feature extraction.

Matlab Command	Function
BW=edge (BW,'canny');	To extract the present of crack in the binarize image.

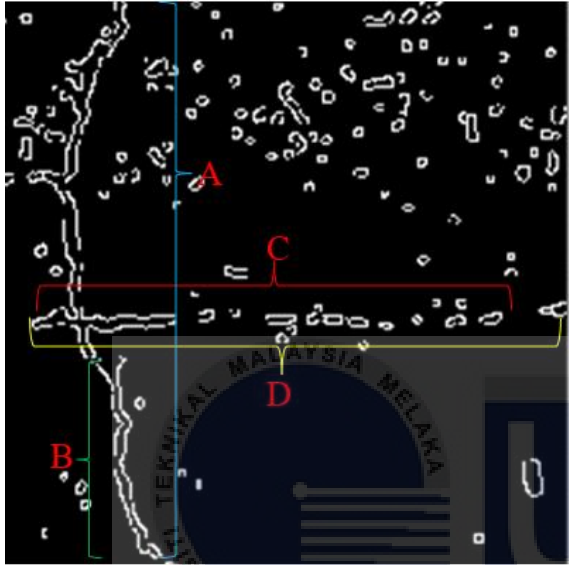
The result of crack image binarizing and feature extraction process that had been used in this work is shown in Table 3.6.

Table 3.6: The crack image binarizing and features extraction process output.

Crack Image Binarizing Process Output	Crack Image Feature Extraction Process Output
	

The length of the white pixel at horizontal and vertical positions is obtained from all the output feature extraction images by using “delta_x and delta_y” command. This command will analyse the white pixel length at each feature extraction output image and the number of black pixels of an image. The command finds all the white pixel coordinates representing a crack in feature extraction output. Then, the maximum and minimum length crack at both positions is determined by using the “max and min” function. From the “max” and “min” data, the average crack length at horizontal and vertical is determined by using the “mean” command. The length is measured in pixel units. After the measurement process is done toward the extracted image, the data is used to perform the KNN training and testing process by recalling the extracted image information by using the “imread” command to read the extracted data and “imgsetTrain” command to label the extracted information as training data. For the testing data, “imgsetTest” command is used to label the extracted information from the images to be used during the testing process. The labelled data information then is used for KNN algorithm training and testing.

Table 3.7: Crack length measurement for horizontal and vertical position.

Crack Length Measurement in Pixel	Description
	<ul style="list-style-type: none"> • The crack line is presented by a white pixel. During the extraction process, "delta_x and delta_y" command is used to measured the number of white pixels at horizontal and vertical. • A is maximum crack line length in vertical. • B is the minimum crack line in vertical. Minimum crack length is determined by measuring the shortest crack line. • C is minimum crack line length in horizontal. • D is maximum crack line length in horizontal. • After maximum and minimum crack length is obtained, the data is used for the KNN training and testing by using recall command "imread".

3.6 Images Classification with K Nearest Neighbour (KNN) Algorithm

3.6.1 Dataset for Training and Testing Quantity

After the datasets undergo image processing, the crack detection process will be done using the KNN algorithm. The KNN algorithm will obtain all the input extracted dataset image information from the previous process, such as the number of white and black pixels by using recall command “imread”. These two data are important information for the KNN algorithm since both data are used as reference data for crack lines represented by white and black pixels representing the image background. Next, from extracted data, the KNN will analyse pixel data in terms of white pixel and black pixel from all input images from crack and non-crack categories. After the KNN algorithm obtains the number of black and white pixels from process images, the KNN will use the data as reference information for dataset training. During the training process, KNN trained the training dataset based on the numbers of white pixels at horizontal and vertical by referring to the original category of the crack or non-crack dataset. The training information will be used as reference data for the testing process. The KNN algorithm randomly chooses the datasets based on the set ratio for the testing process. The testing dataset also undergoes images processing and feature extraction, then the KNN algorithm will analyse the number of white and black pixels from the testing images and compare the data information with the training dataset extracted information.

The number of testing and training needs to be set first before the classification process is done. Two different training and testing processes will be done in this work. Both training use different percentages for training and testing. The purpose of using different amounts of sample size is to observe the effect of different amounts of training and testing datasets toward classification accuracy. The first value for training and testing is 80% training and 20% testing, while for second value is 90% training and 10% testing. Table 3.7 and 3.8 are the table of training and testing data.

Table 3.8: Table for 80% training and 20% testing dataset.

Crack Image Quantity for Testing	20
Non-Crack Image Quantity for Testing	20
Crack Images Quantity for Training	100
Non-Crack Images Quantity for Training	100
K-Nearest Neighbour (KNN) Training Datasets	80% Of 100 for each class.
K-Nearest Neighbour (Knn) Testing Datasets	20% Of 100 for each class.

Table 3.9: Table for 90% training and 10% testing dataset.

Crack Image Quantity for Testing	10
Non-Crack Image Quantity for Testing	10
Crack Images Quantity for Training	100
Non-Crack Images Quantity for Training	100
K-Nearest Neighbour (KNN) Training Datasets	90% Of 100 for each class.
K-Nearest Neighbour (Knn) Testing Datasets	10% Of 100 for each class.

After the testing accuracy for both different testing datasets is achieved more than 70%, the new number of training datasets which is 800 crack and 100 non-crack datasets, will be used for the algorithm training process. This procedure aims to observe the testing classification accuracy when more crack images are higher than non-crack is used for the algorithm training. The testing image quantity that will be used for this process is 10% from 800 datasets which is 80 images. The algorithm will randomly choose 80 crack and non-cracks dataset that had been used for training and performed the testing process. The effect of different numbers of dataset for training toward the testing accuracy will be analysed after performing the classification test.

3.6.2 K Value for KNN Crack Detection

For dataset training and testing, the K value plays a vital role in class prediction. The closest neighbor value used for the class vote is the "K" in the KNN method. It manipulates the training data and uses distance measures to classify the new test data. As a result, the value was utilized to calculate the distance between test and training label points. For this work, the dataset was tested by using three different amounts of K value. The K value is 3, 5 and 7. The accuracy result of using different values of K will be analyzed based on correct class classification. Below is the MATLAB command that has been used in this study to perform the KNN algorithm classification and detection.

Table 3.10: MATLAB command for KNN algorithm.

MATLAB Command	Function
<code>[imgSetTrain, imgSetTest]= splitEachLabel(imgSet,0.8)</code>	To set the amount of training and testing dataset for KNN classification.
<code>Xtrain=[];</code>	To train the dataset for classification.
<code>Ytest = imgSetTest.Labels;</code>	To test the dataset based on set amount of testing.
<code>mdl = fitcknn(Xtrain',Ytrain,'NumNeighbors' ,5,'Standardize',1);</code>	To perform KNN crack detection and classification process.

3.6.3 KNN Testing Visualisation Result

During the KNN training process, the crack information is collected from the processed images by recall back the measured the crack line during extracted process by using "DBpatch" command. The black pixel is not measured since it is declared as the background of the images. The average white pixel at the horizontal and vertical of the

training image is measured to collect the crack line data. The algorithm will train the training dataset and obtain all the training dataset information, such as the number of white pixels and image contrast during the training process. The testing process will be done after the algorithm finish training the dataset. The training information of all training datasets such as the number of white pixels will be used as references during image classification testing. For the result, KNN visualisation and confusion matrix data plot are the output for the testing. The KNN visualisation will show all the testing dataset and their class prediction. Figure 3.19 is the KNN classification visualisation output sample figure used in this work.

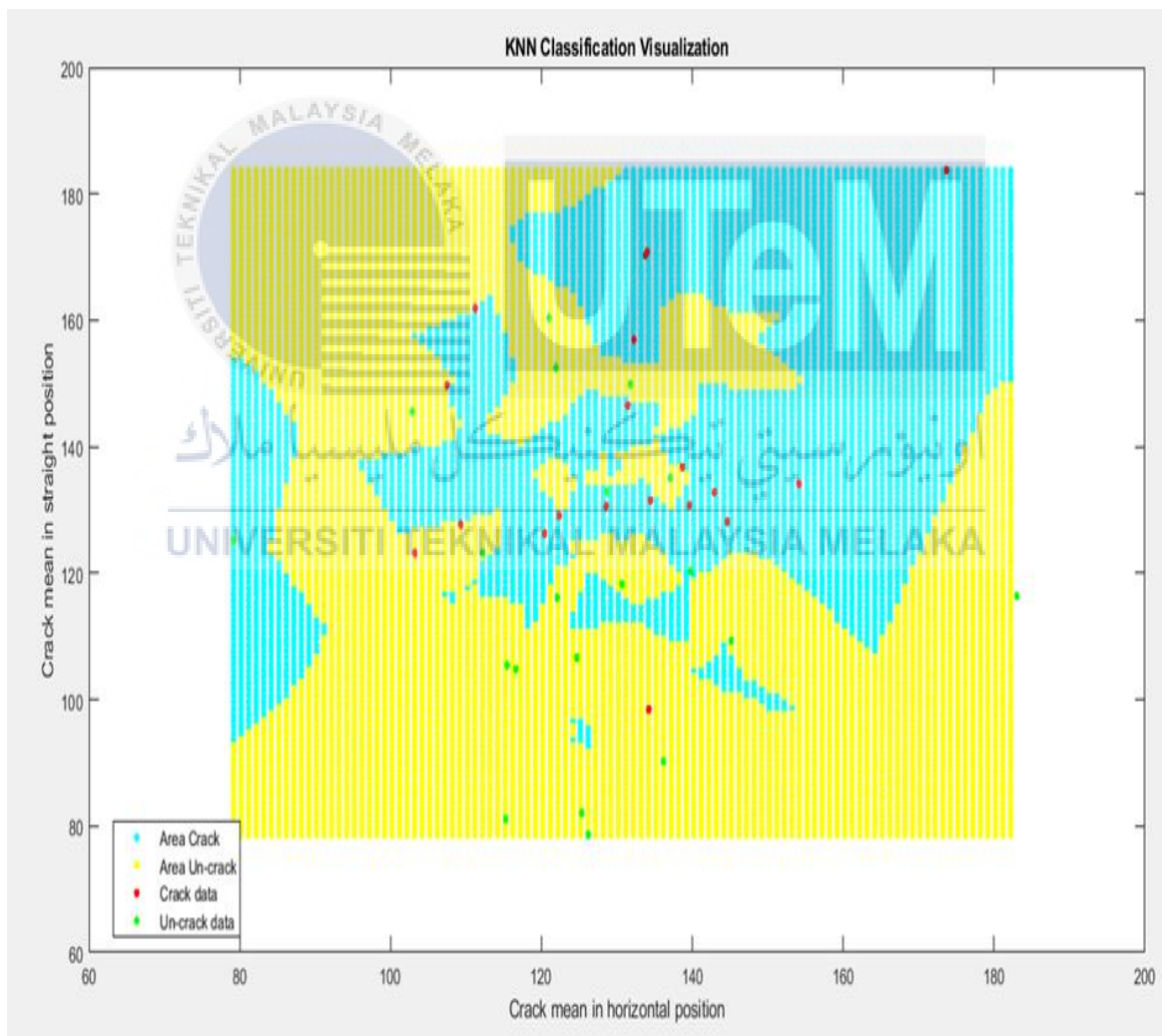


Figure 3.16: KNN visualisation graph.

3.6.4 KNN Visualisation Description

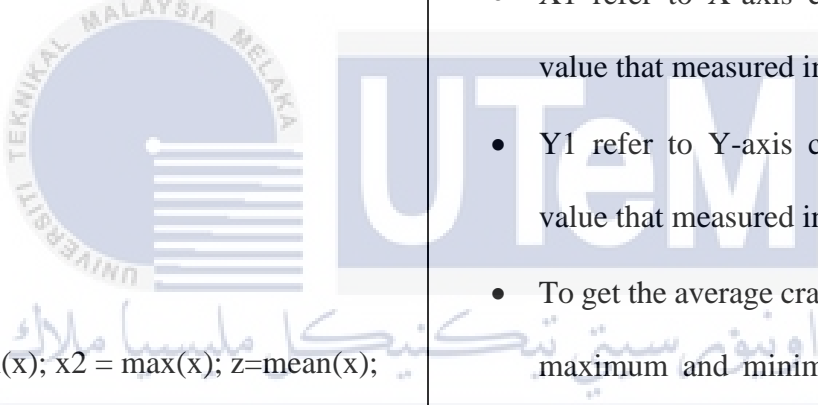
In this graph, four different colours indicate four important pieces of information. The command used to generate the graph is “gscatter” command. The X-axis in this graph indicates the mean crack length in the horizontal of the images and the Y-axis indicates the mean crack length in the vertical position obtained during the training process. The maximum X-axis value in the graph is the highest crack length, while the Y-axis maximum value is the highest crack length in vertical. The crack length is obtained from training data. To obtain the mean crack length at horizontal and vertical for each image, the KNN algorithm will measure the maximum length and minimum crack length at horizontal and vertical from the extracted information by using recall command. Then, the maximum and minimum length is total together and divided by two to get the mean value by using “mean” command.

The crack length means value data is used to plot the region of crack and non-crack based on the true dataset class. The information to plot the axis line is received from the training process. The blue region shows the crack area. The crack area is plotted based on the average amount of white pixels of the crack image gathered from the training process. While yellow region that indicates uncrack area is plotted from the information of white pixel gathered from non-crack images during the training process. The red and green dots represent the exact class category of the testing datasets. The exact class category of the testing dataset is the dataset declared as crack and non-crack during the training process. The red dot is a crack dataset and the green dots are the uncracked dataset.

To interpret the visualisation graph, first, the blue and yellow region that represents testing class prediction needs to be identified. Next, the position red dot and green dot is observed. When the red dot representing the crack data is on the blue region, the prediction is correct, classifying the crack data as crack. Next, when the green dot is plotted at the

yellow region, the testing class prediction is correct since the non-crack dataset is predicted as non-crack. While if the red dot is plotted at the yellow region, the prediction is wrong since the crack data is predicted as non-crack data. The prediction is also wrong if the green dot is a plot at the blue region. This visualisation graph will clearly show the number of correct and wrong datasets class prediction. Below is the command used in this work to measure all the crack lengths of the training dataset and produce the KNN testing visualisation graph.

Table 3.11: MATLAB command for KNN visualisation.

MATLAB Command	Description
 <pre data-bbox="272 1238 778 1346">x1 = min(x); x2 = max(x); z=mean(x); y1 = min(y); y2 = max(y); z2=mean(y);</pre>	<ul style="list-style-type: none"> • X1 refer to X-axis crack average value that measured in white pixel. • Y1 refer to Y-axis crack average value that measured in white pixel. • To get the average crack length, the maximum and minimum value of crack data that represented by white pixel need to be subtract and divided by two. The “mean” command is used to obtain the average crack length.
<pre data-bbox="300 1865 751 1899">gscatter(x1(:),x2(:),ms,'cym','!',10);</pre>	<p>This command is used to plot the KNN visualisation graph based on measured information such as position of crack that had been received during training process.</p>

3.7 Result Evaluation

3.7.1 Confusion Matrix

After the image was classified using the KNN algorithm, the result was evaluated using the Confusion matrix. The classification accuracy will be evaluated in this part. A confusion matrix is a summary table, also known as an error matrix, which is used to evaluate the performance of a classification model. Count values are used to sum and break down the number of accurate and wrong predictions by class. Below is a schematic of a two-by-two confusion matrix. For example, 10 times out of ten, the classification model predicted "Yes," and the actual result was "Yes". The number ten would then be placed in the True Positive quadrant's upper left corner. Figure 3.20 shows the term that had been used to plot the confusion matrix.

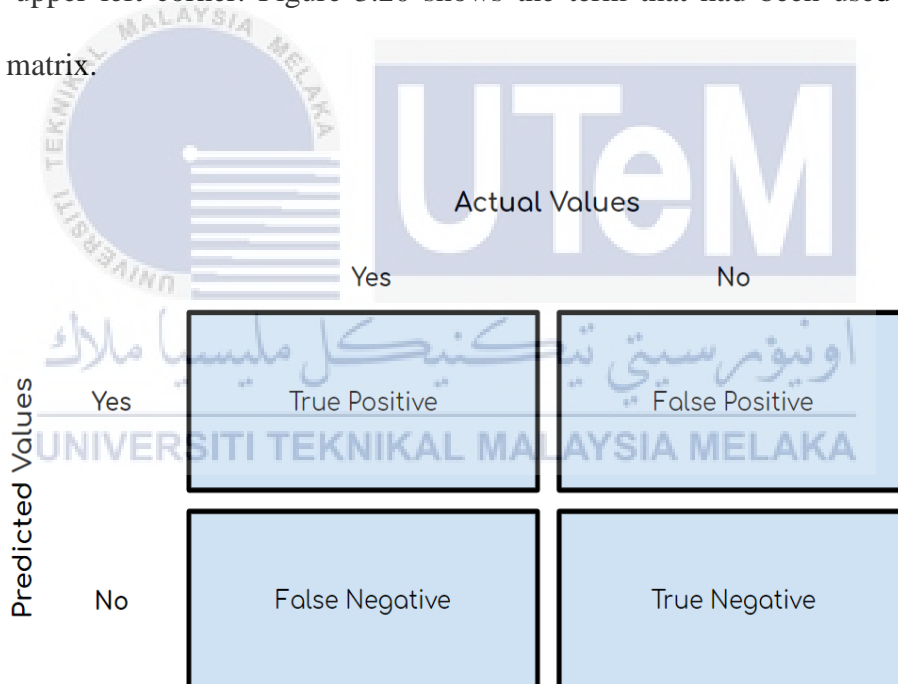


Figure 3.17: Table of 2x2 confusion matrix from the source (Shin, 2020).

- Positive (P): Observation is positive for example the present of crack.
- Negative (N): Observation is not positive for example there is no present of crack.
- True Positive (TP): The positive class is appropriately predicted by the model.

- True Negative (TN): When the model properly predicts the negative class, this is the result.
- False Positive (FP): The result when the model predicts the positive class when it is actually negative, it is also known as a type 1 error.
- False Negative (FN): The result when the model predicts the negative class when it is actually positive, also known as a type 2 error.
- The accuracy of the classification can be determined by the proportion of predictions.

For this project, there are three parameters that will be analysed from the confusion matrix table. The parameter is precision, which shows the percentage of data declared as crack, the recall, which shows the percentage of correct crack class classification, and accuracy, which shows the total amount of crack and non-crack correct class classification. Below is the table for the calculation of precision, recall and classification accuracy.

		TRUE CLASS CATEGORY	
		TRUE	FALSE
PREDICTED CLASS	TRUE	TP (True Positive) <i>Corect result</i>	FP (False Positive) <i>Unexpected result</i>
	FALSE	FN (False Negative) <i>Missing result</i>	TN (True Negative) <i>Corect absence of result</i>

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 3.18: Formula for precision, recall and accuracy source from (DataQ, 2013).

Table 3.13 is the MATLAB command that has been used in this study to perform the confusion matrix data tabulation.

Table 3.12: MATLAB command for Confusion Matrix data tabulation.

MATLAB Command	Function
overallaccuracy = accuracy_score(Ypred,Ytest);	To get the KNN testing classification result.
saveas(plotconfusion(Ypred,Ytest),	To plot confusion matrix classification result and save as image.

For this work, the confusion matrix output is same as the below figure.

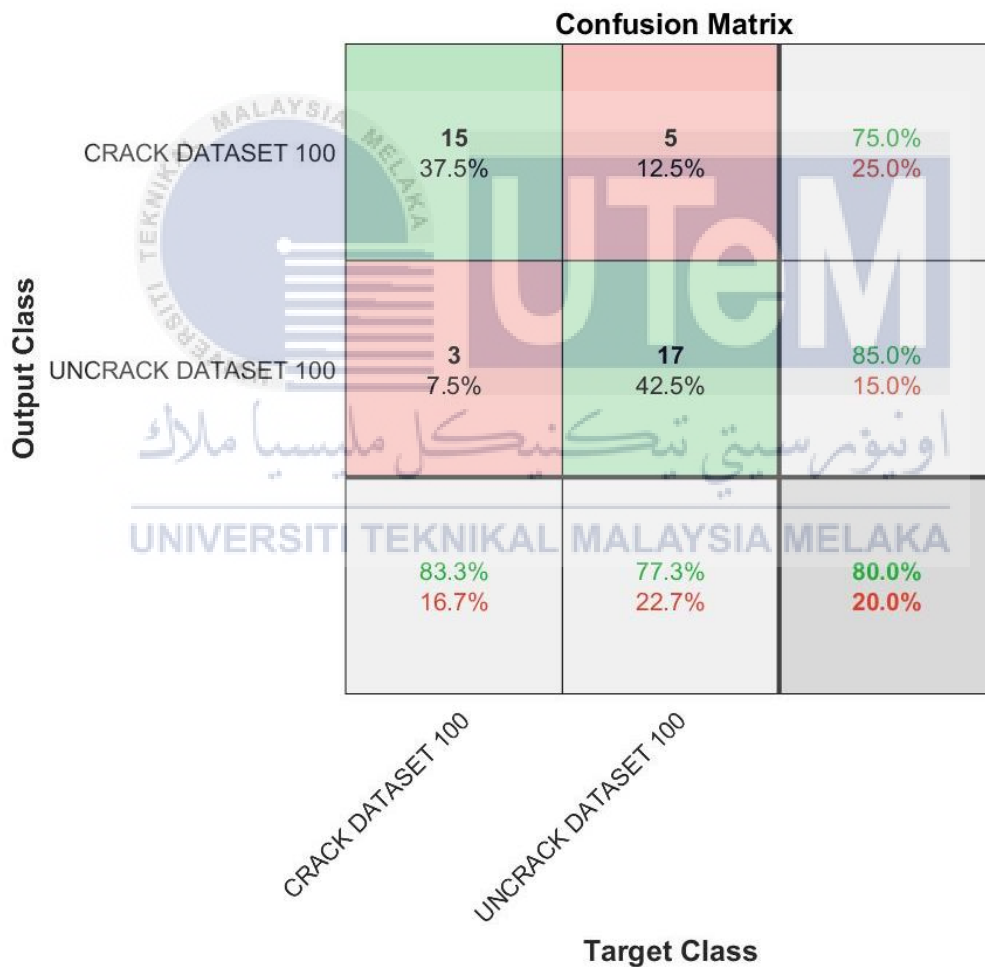


Figure 3.19: Table of Confusion Matrix.

From the confusion matrix table, there is three rows. For the first row, the left column presents the correct crack class prediction. The middle column represents wrong crack prediction, which is the crack image that declares as non-crack image. The right column represents the precision data for the classification.

For the second row, the left column presents the wrong non-crack prediction data, that is, non-crack data that is predicted as crack data. While for the middle column, the data represent the number of correct non-crack images prediction and the right column present the percentage of correct non-crack images from a total of non-crack prediction.

The last row, the left column, shows the percentage of recall for correct crack class prediction. At the same time, the middle column represents the percentage of correct non-crack images classification per the total number of correct non-crack images prediction wrong crack images prediction. Lastly, the right column represents the classification accuracy. Table 3.14 is the formula table for confusion matrix plot.

Table 3.13: Confusion Matrix calculation formula.

CONFUSION MATRIX			
PREDICTION CRACK	TP	FP	PRECISION
PREDICTION NON-CRACK	FN	TN	$\frac{TN}{TN + FN}$
	RECALL	$\frac{TN}{TN + FP}$	ACCURACY
	CRACK DATASET	NON-CRACK DATASET	

3.8 Random Images Testing

After the highest crack detection accuracy is achieved, random images testing will be done. This testing aims to observe the crack classification accuracy when the crack dataset and non-crack dataset are placed together in one file and the algorithm parameter such as the number of best K value and best number of the training dataset is used from the previous testing. This testing simulates the real situation when this algorithm is used for real crack detection. The number of datasets is 10 crack images and 10 non-crack images. The classification accuracy is analyzed based on correct class prediction. Below are the random testing datasets.



Figure 3.20: Random crack and non-crack dataset for classification test.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

This chapter shows the result and discusses the process that has been performed for crack image detection by using the KNN Algorithm. The results that have been achieved for this work are image processing and feature extraction, the KNN algorithm classification testing accuracy, confusion matrix result, and random image crack detection testing result. The different processes during crack detection using the KNN algorithm produce different outputs that affect the classification accuracy.

4.2 Images Processing and Features Extraction

For the image processing phase, the input RGB image was successfully converted into greyscale images and underwent binarizing process at three different greyscale intensity levels: 0.2, 0.4 and 0.6. During this process, the grey images will be converted into black and white image format. The image processing method by Varsha et.al (2019) is applied in this process. The method shows same output that is RGB image is successfully converted into binary images. In Varsha et.al (2019), the number of greyscale intensity level that used to convert from greyscale image to binary images is 0.2, while in this work the best greyscale intensity value that show a clear present of crack after undergoes Canny feature extraction is 0.4.

Three different greyscale intensity levels show the different outputs after the image undergoes the features extraction process. The Canny feature extraction result shows that the presence of crack concrete in the image after undergoing Canny feature extraction is more apparent when the higher greyscale intensity value is applied. This can be observed by the presence of white pixel increases when the greyscale intensity level is 0.6 compared to the lowest level, which is 0.2 that the presence of crack is lightly shown. From the result observation, the best greyscale intensity level representing the crack data is 0.4 based on the feature extraction result because the presence of white pixel that represents crack is clear and has low images noises and a smaller number of impurities that converted into the white pixel.

When the greyscale intensity is at 0.6, the presence of white pixel is too much after undergoing feature extraction. This will make the image noises and impurity also will be converted into the white pixel. While, when the greyscale intensity is set to 0.2, the presence of white pixel that represents crack shows less and light. This will make the algorithm have less white pixel data that represent crack. The relationship between the greyscale intensity level and Canny features extraction is, when the binarized images have low or too high greyscale intensity level, this will make the thresholding process during feature extraction cannot replace the image pixel correctly.

When the greyscale intensity level is low, most white pixels will be replaced with black pixels since the intensity level is low. This will make the white crack pixel data also converted into the black pixel. Thus, the crack information will be decreased for the algorithm to train. By referring to Ali & Clausi (2001), correct initial images processing method will make improved the data extracting during feature extraction process. This because, the images that have low noise will make thresholding process during feature extraction become easier for algorithm to improve the image pixel. While, if the intensity level is too high, many black pixels will be converted into the white pixel.

The noises and impurity data with high grey intensity pixels will also be converted into white pixels. This will crack data that represent by white pixel become too high. Next, the Gaussian filter will smooth the image by reducing the images noises again to highlight the presence of a crack. The process of replacing images pixel will be done again during the smoothing process to improve the presence of white pixels in the dataset. Below is the several feature extraction result for different greyscale intensity levels.

1. The grey scale intensity level is 0.2 and the type of image that used for image processing is crack image.

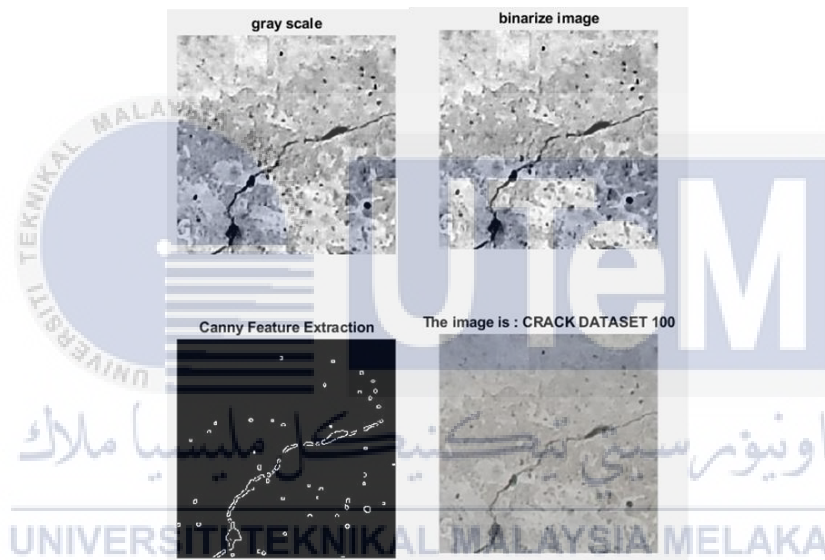


Figure 4.1: 0.2 Geyscale intensity level result.

2. The grey scale intensity level is 0.4 and the type of image that used for image processing is crack image.

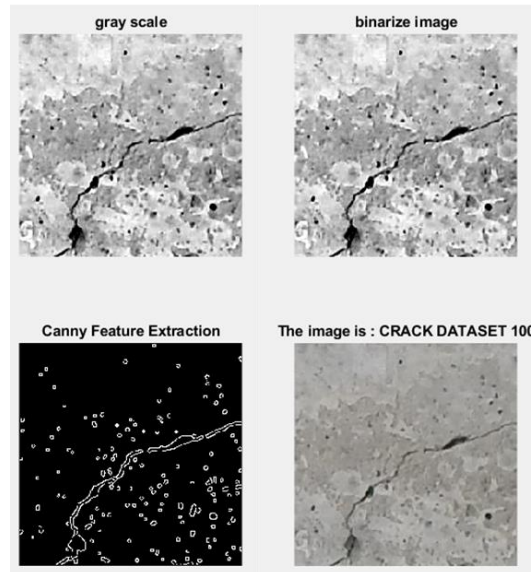


Figure 4.2: 0.4 Greyscale intensity level result.

3. The grey scale intensity level is 0.6 and the type of image that used for image processing is crack image.

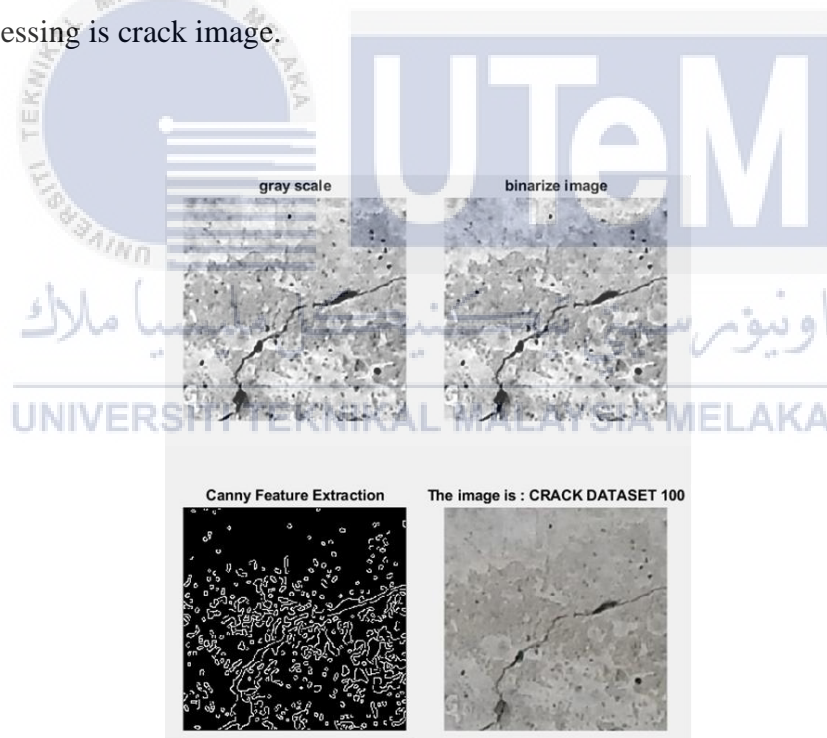


Figure 4.3: 0.6 Greyscale intensity level result.

4. The grey scale intensity level is 0.4 and the type of image that used for image processing is non-crack image.

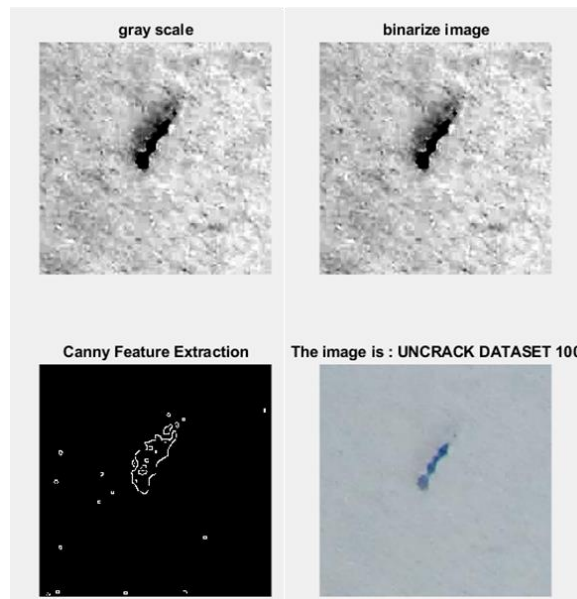


Figure 4.4: 0.4 Greyscale intensity level result for non-crack image.

For the feature extraction process, the binarized image undergoes two thresholding processes. This process will improve the number of white and black pixels based on the set greyscale intensity level. After thresholding was done for training and testing the dataset, the output image will be applied Gaussian filter to reduce the image noises and blur the image background. Next, the “delta_x and delta_y” command will calculate the number of white pixels for all training datasets from both categories after filtering process. The number of white pixels is used as reference data during the KNN testing process. The maximum white pixel amount at the horizontal position that had been obtained in this work during feature extraction is 180 pixels and the minimum is 30 pixels. While for vertical, the maximum is 170 pixels and the minimum is 30 pixels.

The measured data is obtained by using “max”, “min” and “mean” command. The pixel data from the training process is used as reference data during KNN classification testing by using recall command “imread”. The training pixel data in horizontal and vertical is used to plot the KNN visualization crack and non-crack region. Figure 4.5 is one of the

crack length measurement results obtained in this work during the training process. The maximum crack length for the vertical is 175 pixels and the minimum is 56 pixels. For horizontal, the maximum length is 175 pixels and the minimum is 160 pixels. From this measurement, KNN will use this information for the training process. The information is also used as reference data during testing to classify the testing image as crack or non-crack based on the number of white pixels of the testing data.

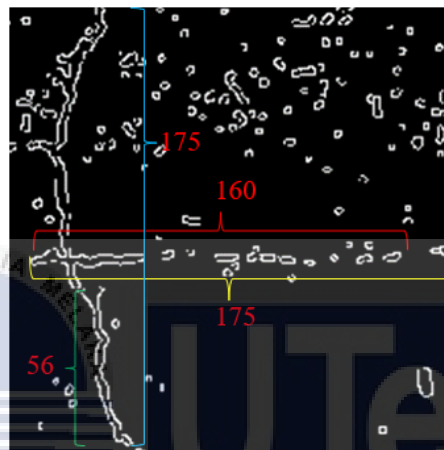


Figure 4.5: Crack length measurement for training dataset.

4.3 K-Nearest Neighbor (KNN) Image Classification Result

For the initial image detection and classification process, the number of datasets used in this project for algorithm training is 200 datasets consisting of 100 crack and 100 non-crack datasets. The different number of K values, which are 3, 5 and 7, were used during the image detection process and the greyscale intensity level was also different during the testing process. The image also undergoes a different amount of training dataset, which is 80% for training and 20% for testing and 90% for training and 10% for testing. The classification accuracy result is stated below.

4.3.1 The Testing Result For 80% Training And 20% Testing

KNN classification accuracy data result for K value 3.

Table 4.1: KNN classification accuracy data result for K value 3.

Total Crack Image and Non-Crack for Testing		40	
Crack Images Quantity for Training		100	
Non-Crack Images Quantity for Training		100	
K-Nearest Neighbour (Knn) Value		3	
K-Nearest Neighbour (Knn) Training Dataset		80%	
K-Nearest Neighbour (Knn) Testing Dataset		20%	
Classification Accuracy (%)			
Grayscale Intensity Level Adjustment	Crack	Non- Crack	Total Accuracy
0.2	80.00	70.00	75.00
0.4	75.00	85.00	80.00
0.6	80.00	45.00	62.50
Highest Total Accuracy (%)			80.00

Confusion Matrix accuracy data result for K value 3.

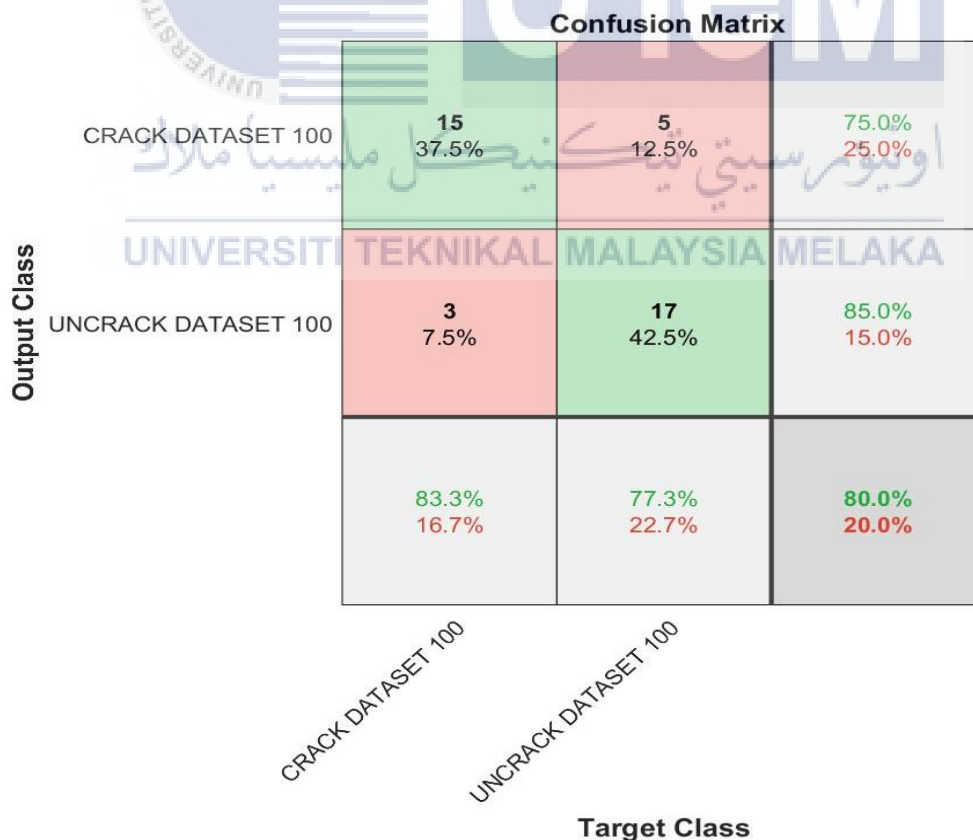


Figure 4.6: Confusion Matrix accuracy data result for K value 3.

KNN classification visualization result for K value 3.

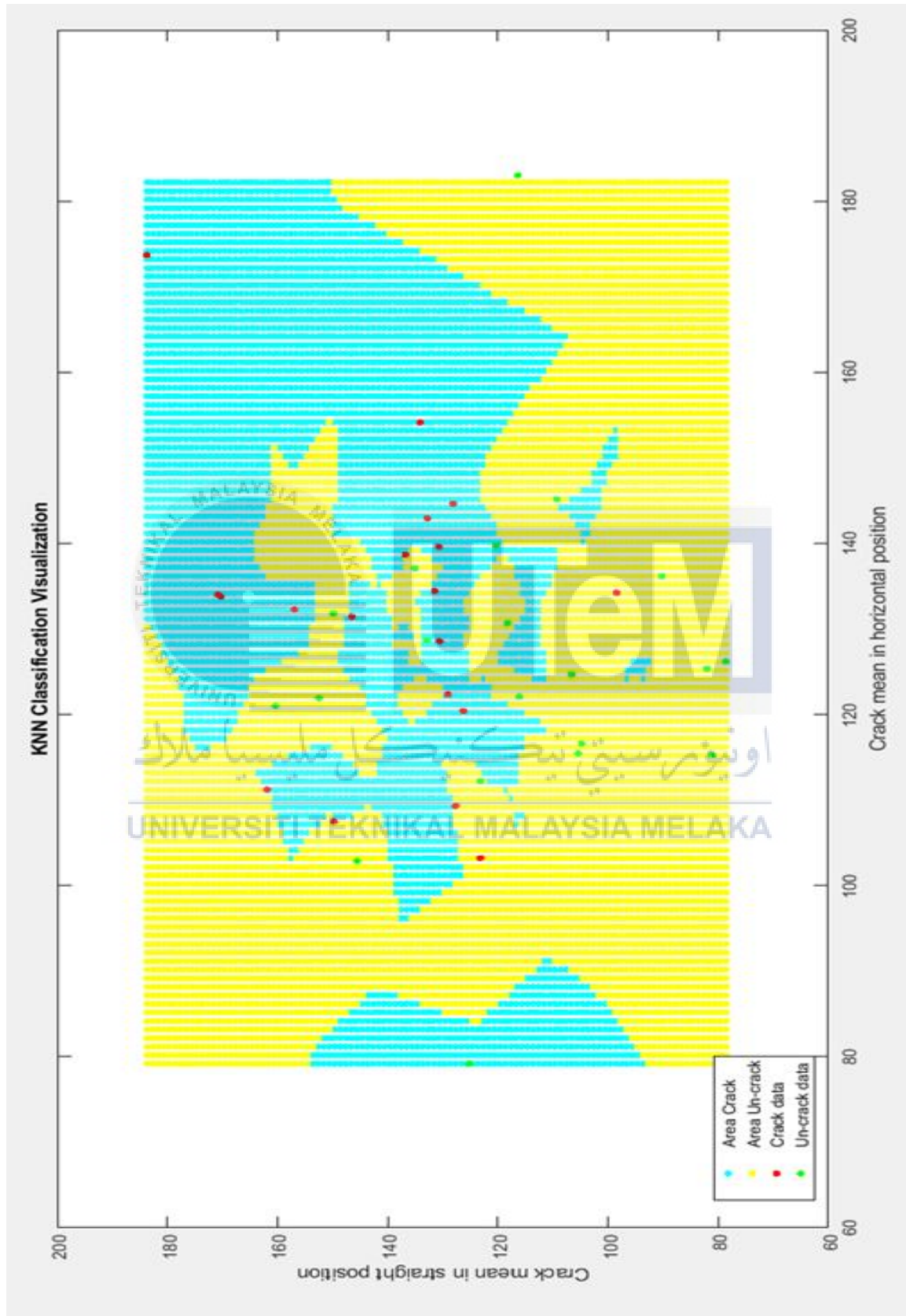


Figure 4.7: KNN classification visualization result for K value 3.

KNN classification accuracy data result for K value 5.

Table 4.2: KNN classification accuracy data result for K value 5.

Total Crack Image and Non-Crack for Testing		40	
Crack Images Quantity for Training		100	
Non-Crack Images Quantity for Training		100	
K-Nearest Neighbour (Knn) Value		5	
K-Nearest Neighbour (Knn) Training Dataset		80%	
K-Nearest Neighbour (Knn) Testing Dataset		20%	
Classification Accuracy (%)			
Grayscale Intensity Level Adjustment	Crack	Non- Crack	Total Accuracy
0.2	55.00	85.00	70 .00
0.4	75.00	65.00	70 .00
0.6	65.00	65.00	65.00
Highest Total Accuracy (%)			70.00

Confusion Matrix accuracy data result for K value 5.

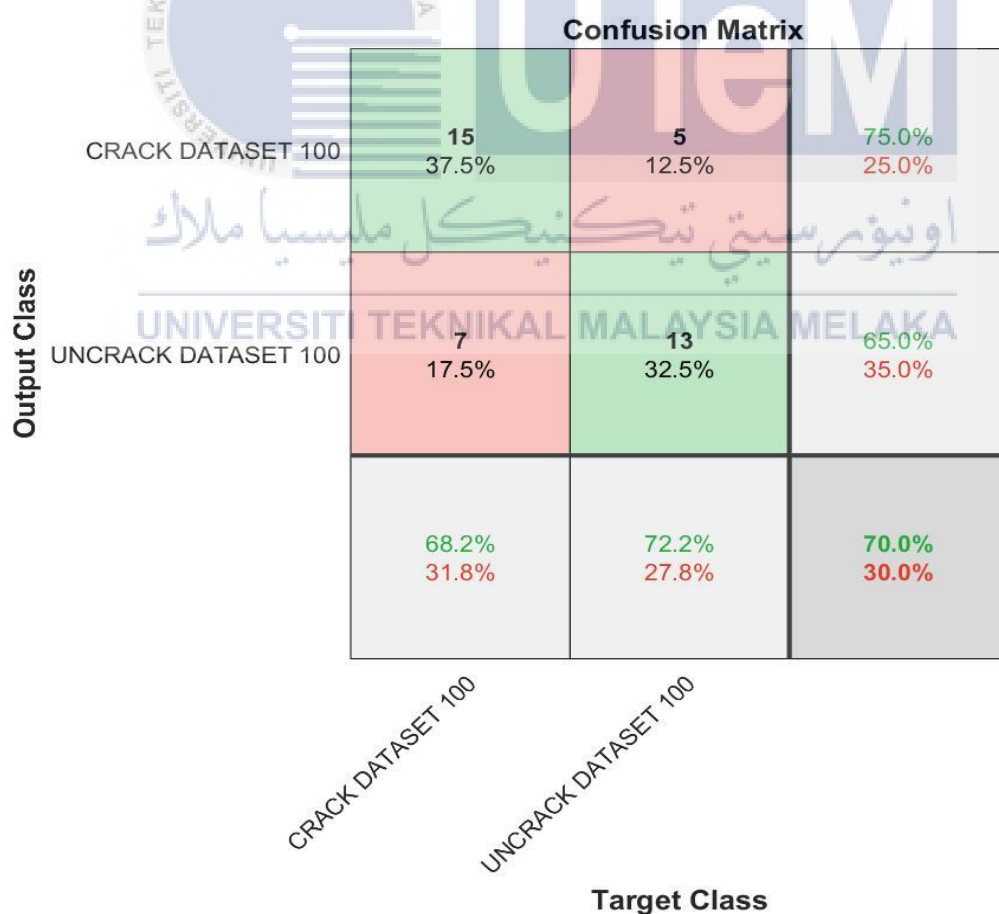


Figure 4.8: Confusion Matrix accuracy data result for K value 5.

KNN classification visualization result for K value 5.

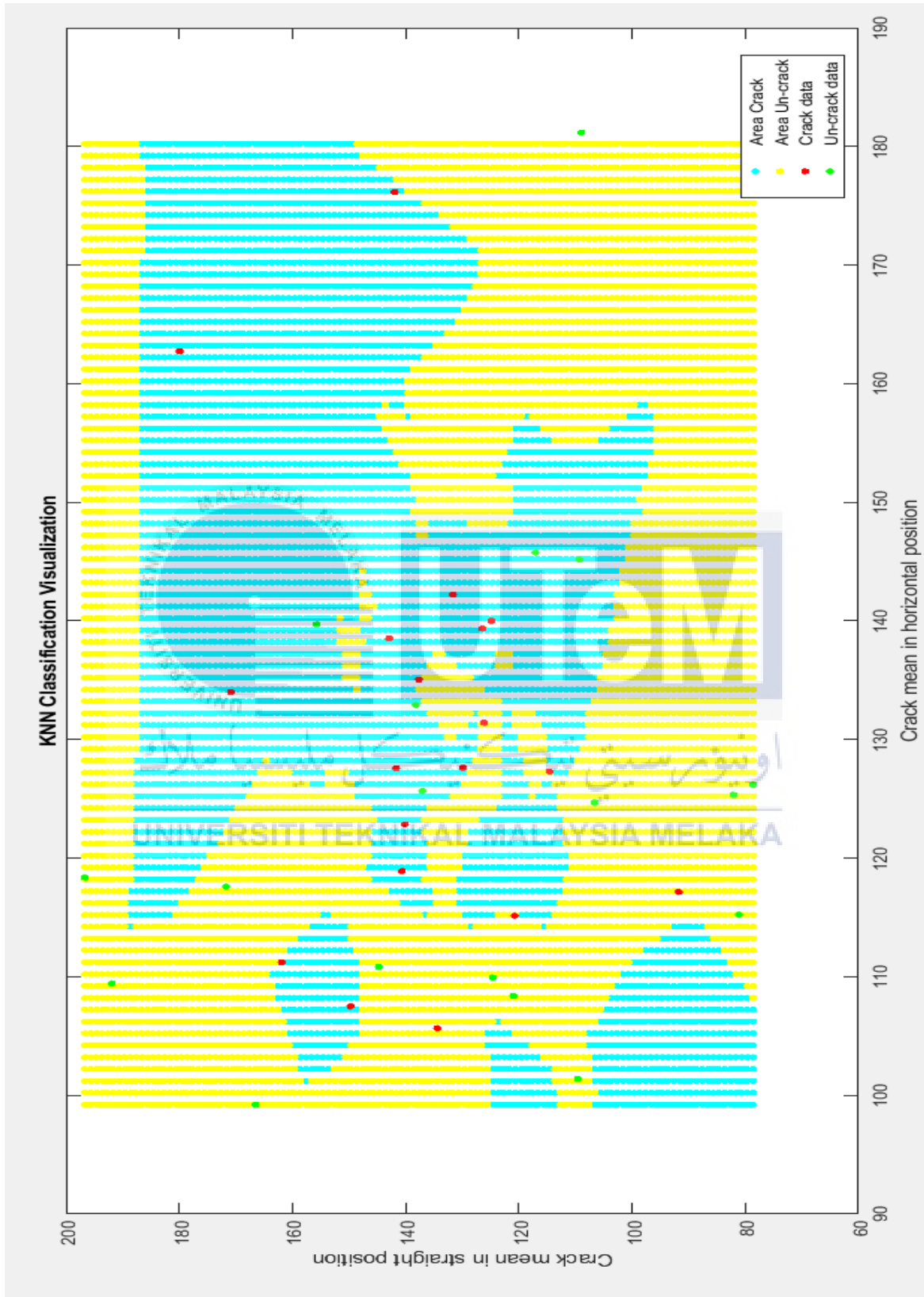


Figure 4.9: KNN classification visualization result for K value 5.

KNN classification accuracy data result for K value 7

Table 4.3: KNN classification accuracy data result for K value 7.

Total Crack Image and Non-Crack for Testing		40	
Crack Images Quantity for Training		100	
Non-Crack Images Quantity for Training		100	
K-Nearest Neighbour (Knn) Value		7	
K-Nearest Neighbour (Knn) Training Dataset		80%	
K-Nearest Neighbour (Knn) Testing Dataset		20%	
Classification Accuracy (%)			
Grayscale Intensity Level Adjustment	Crack	Non- Crack	Total Accuracy
0.2	80.00	45.00	62.50
0.4	70.00	65.00	67.50
0.6	65.00	60.00	62.50
Highest Total Accuracy (%)			67.50

Confusion Matrix accuracy data result for K value 7.

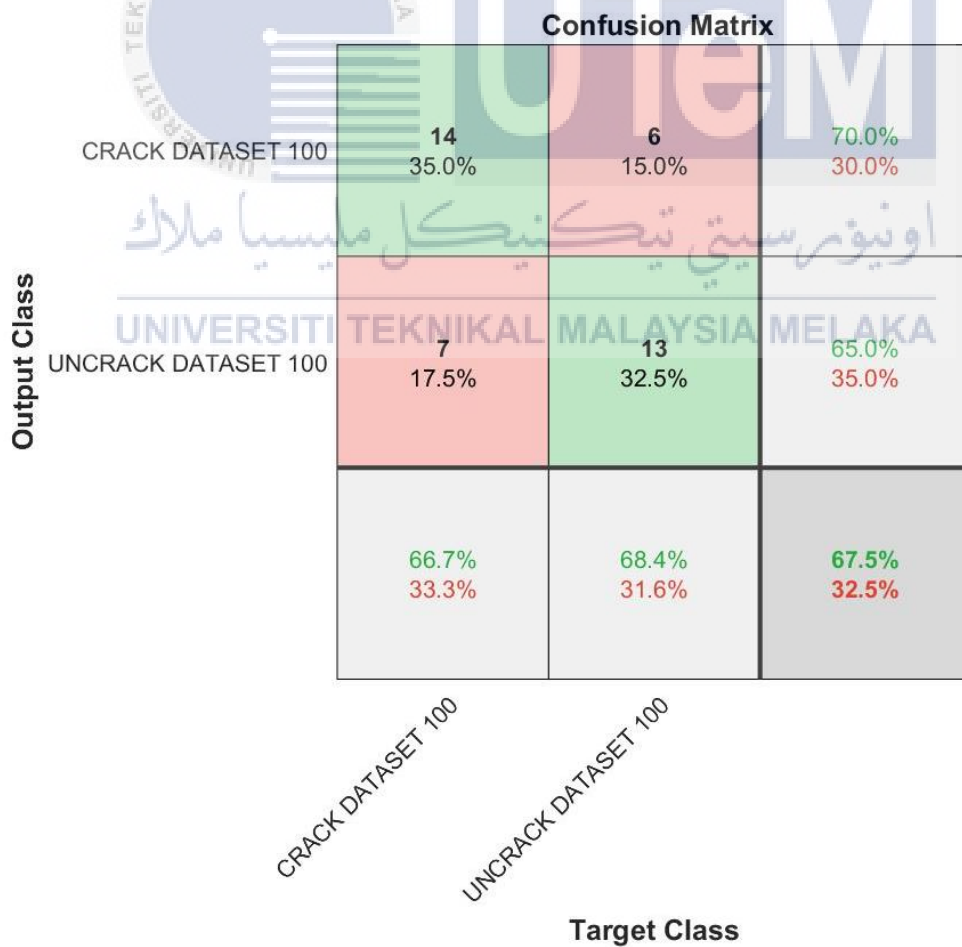


Figure 4.10: Confusion Matrix accuracy data result for K value 7.

KNN classification visualization result for K value 7.

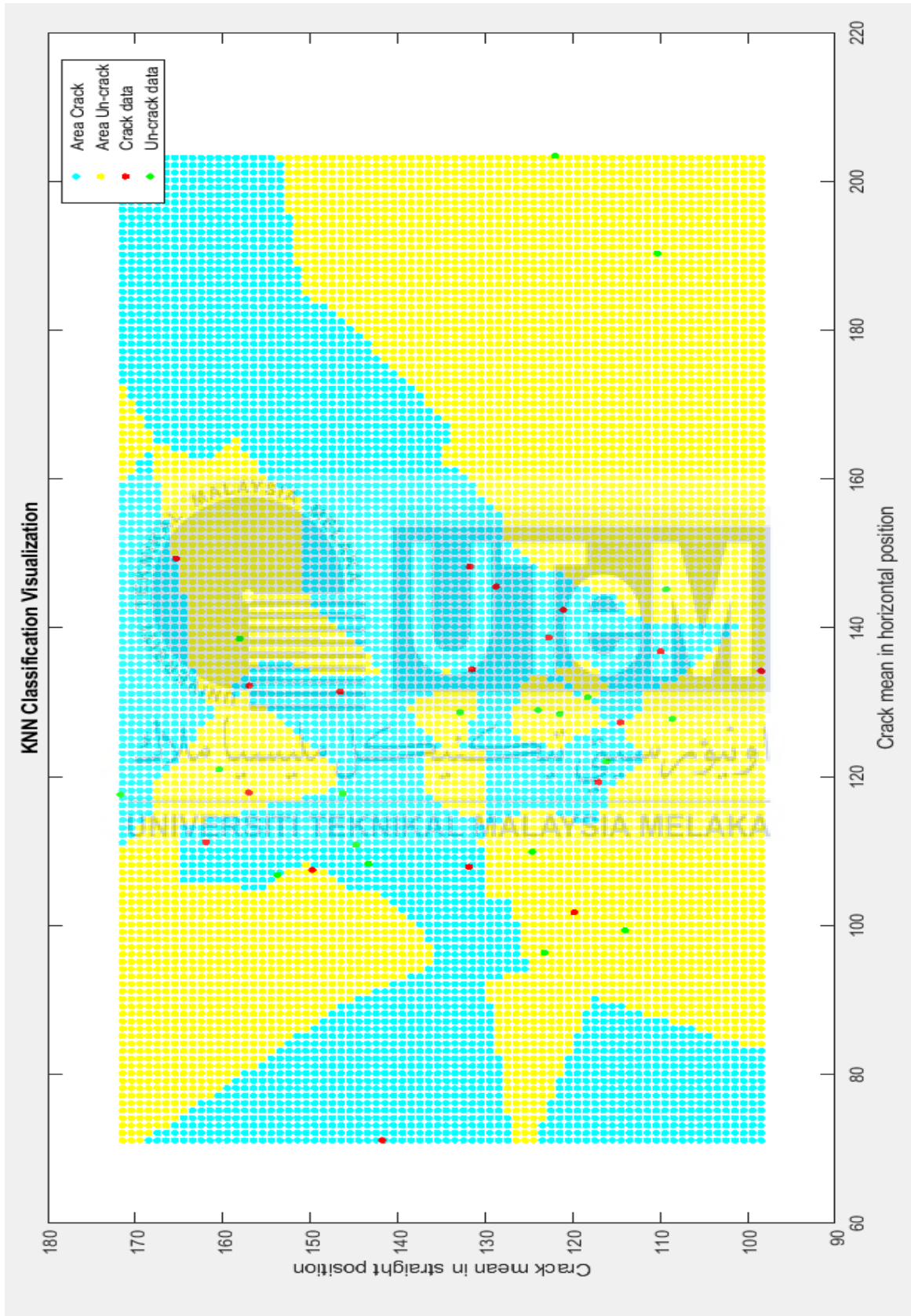


Figure 4.11: KNN classification visualization result for K value 7.

Summary of 80% Training and 20% Testing Dataset.

From the above accuracy result, the highest crack detection accuracy is 80.00%, using K value 3 and the greyscale intensity level is 0.04. The number of testing datasets is 40 images. For this accuracy, the correct crack class prediction is 75.00% correct crack image detection and 85.00% correct non-crack images detection. For K value 5, the correct classification accuracy is 70.00%, while for K value 7, the correct classification accuracy is 67.50%.

4.3.2 The Testing Result For 90% Training And 10% Testing

KNN classification accuracy data result for K value 3.

Table 4.4: KNN classification accuracy data result for K value 3.

Total Crack Image and Non-Crack for Testing		20	
Crack Images Quantity for Training		100	
Non-Crack Images Quantity for Training		100	
K-Nearest Neighbour (Knn) Value		3	
K-Nearest Neighbour (Knn) Training Dataset		90%	
K-Nearest Neighbour (Knn) Testing Dataset		10%	
Classification Accuracy (%)			
Grayscale Intensity Level Adjustment	Crack	Non- Crack	Total Accuracy
0.2	80.00	70.00	75.00
0.4	80.00	80.00	80.00
0.6	70.00	70.00	70.00
Highest Total Accuracy (%)			80.00

Confusion Matrix accuracy data result for K value 3.

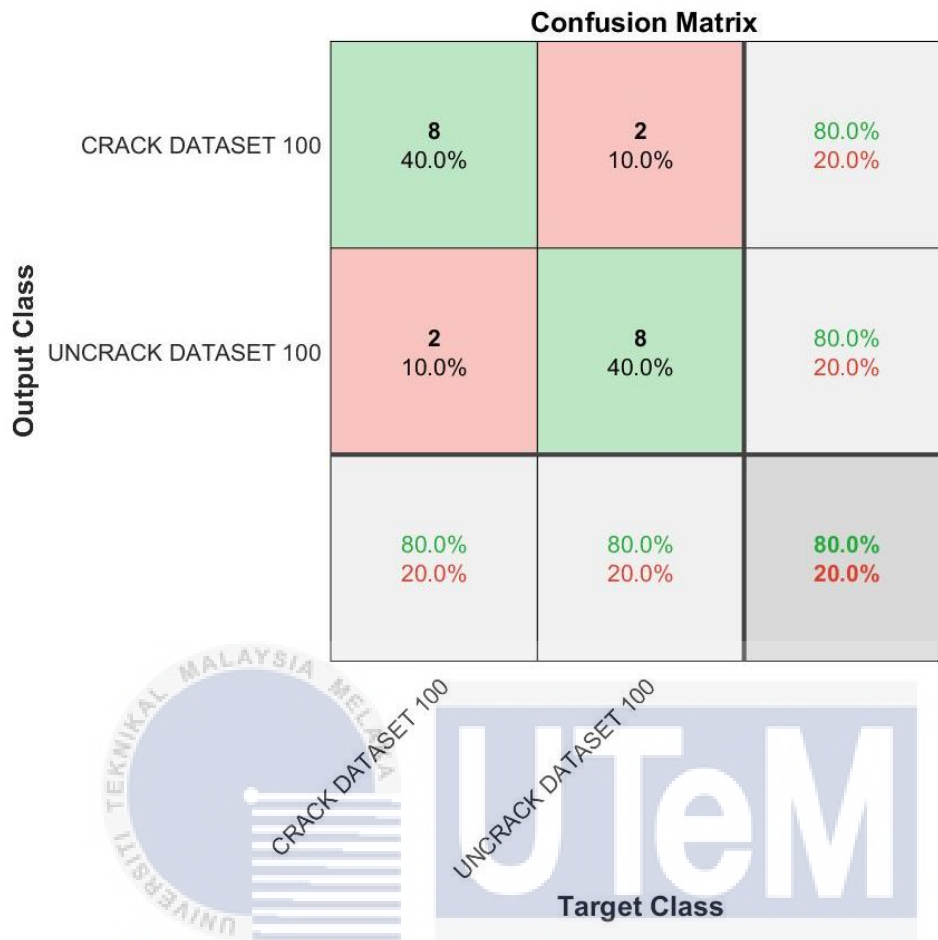


Figure 4.12: Confusion Matrix accuracy data result for K value 3.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

KNN classification visualization result for K value 3.

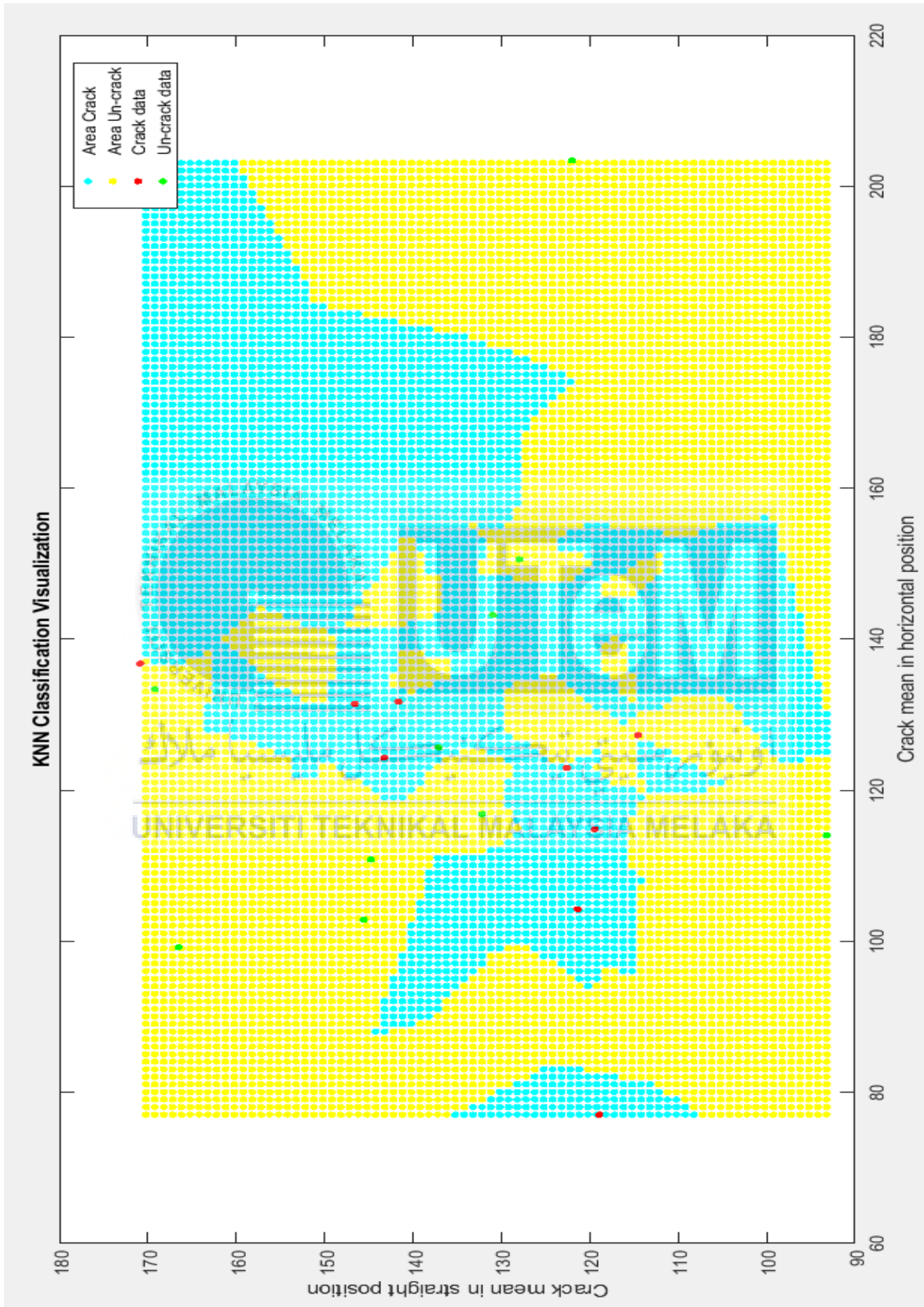


Figure 4.13: KNN classification visualization result for K value 3.

KNN classification accuracy data result for K value 5.

Table 4.5: KNN classification accuracy data result for K value 5.

Total Crack Image and Non-Crack for Testing		20	
Crack Images Quantity for Training		100	
Non-Crack Images Quantity for Training		100	
K-Nearest Neighbour (Knn) Value		5	
K-Nearest Neighbour (Knn) Training Dataset		90%	
K-Nearest Neighbour (Knn) Testing Dataset		10%	
		Classification Accuracy (%)	
Grayscale Intensity Level Adjustment	Crack	Non- Crack	Total Accuracy
0.2	70.00	60.00	65.00
0.4	80.00	60.00	70.00
0.6	50.00	60.00	55.00
Highest Total Accuracy (%)			70.00

Confusion Matrix accuracy data result for K value 5.

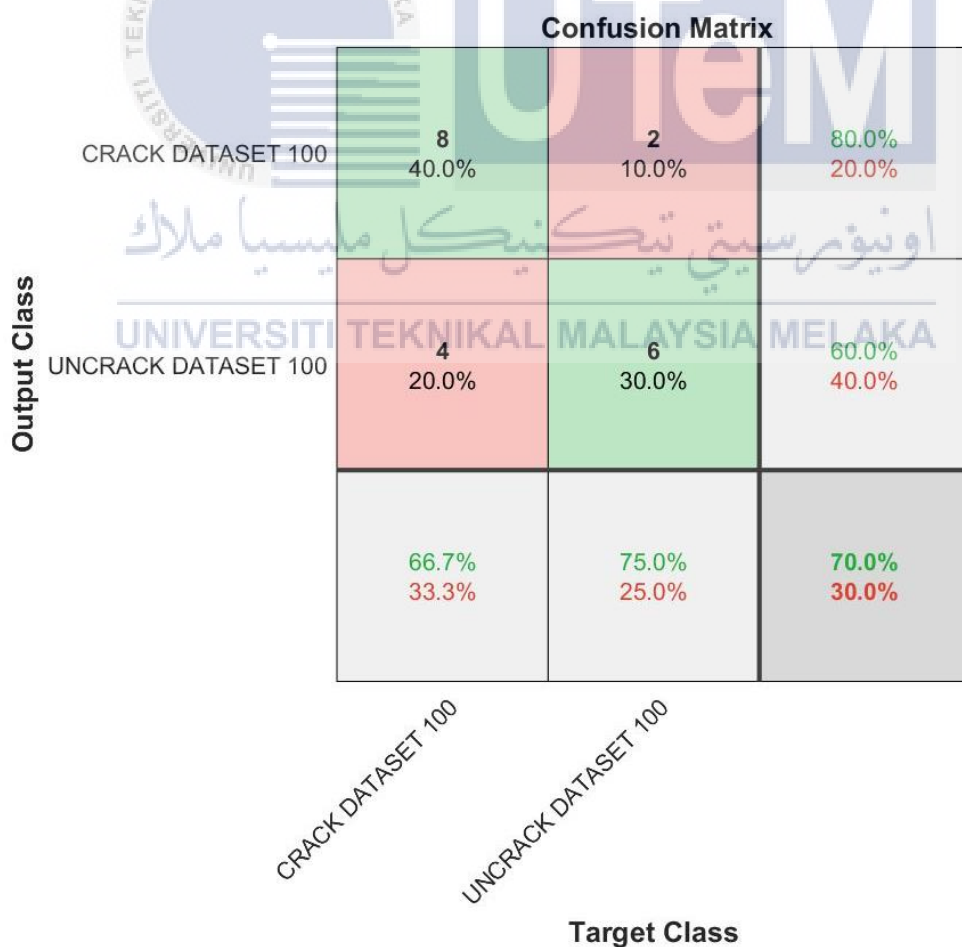


Figure 4.14: Confusion Matrix accuracy data result for K value 5.

KNN classification visualization result for K value 5.

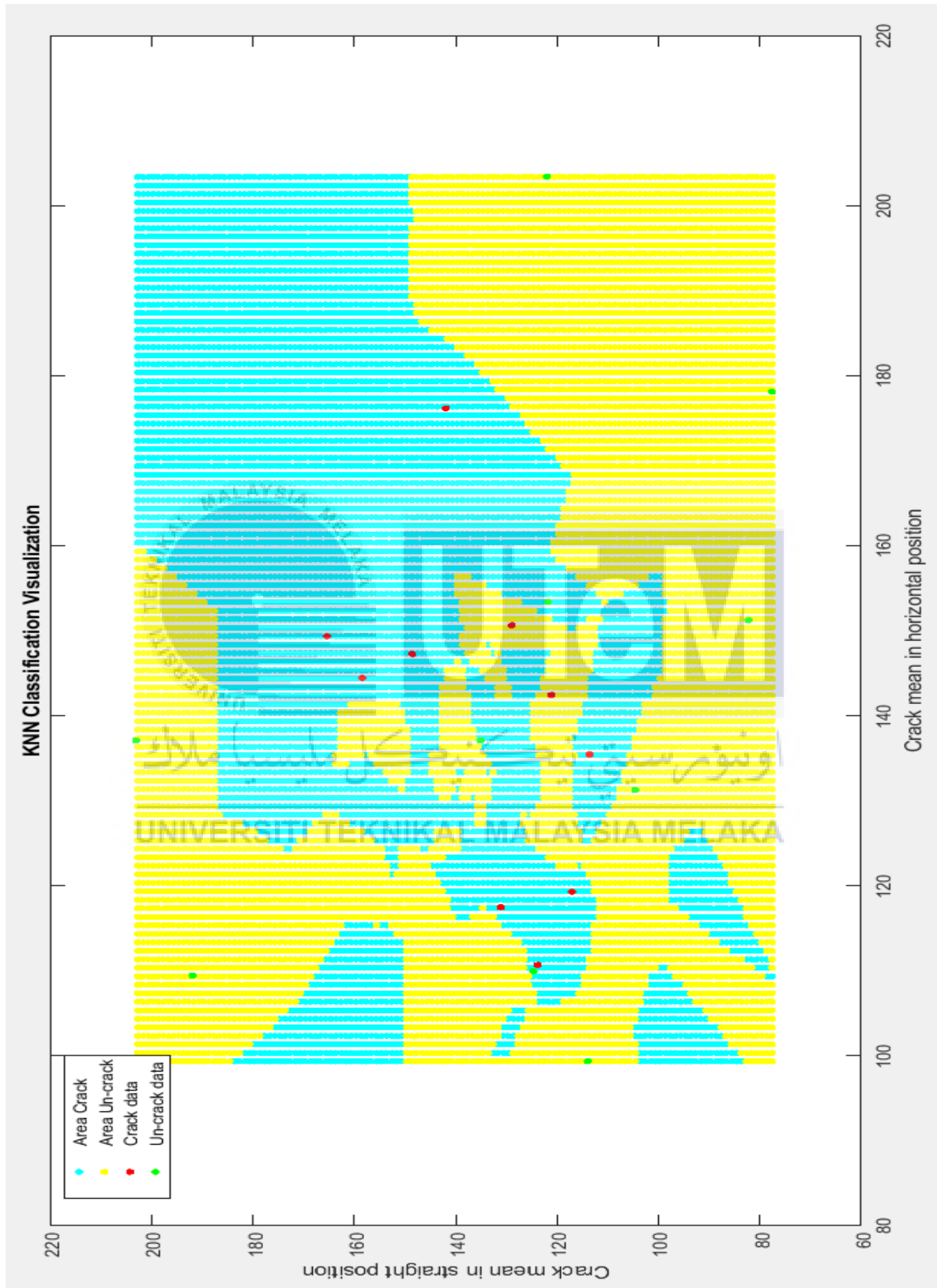


Figure 4.15: KNN classification visualization result for K value 5.

KNN classification accuracy data result for K value 7.

Table 4.6: KNN classification accuracy data result for K value 7.

Total Crack Image and Non-Crack for Testing		20	
Crack Images Quantity for Training		100	
Non-Crack Images Quantity for Training		100	
K-Nearest Neighbour (Knn) Value		7	
K-Nearest Neighbour (Knn) Training Dataset		90%	
K-Nearest Neighbour (Knn) Testing Dataset		10%	
Classification Accuracy (%)			
Grayscale Intensity Level Adjustment	Crack	Non- Crack	Total Accuracy
0.2	30,00	90,00	60,00
0.4	80,00	60,00	65,00
0.6	60,00	50,00	55,00
Highest Total Accuracy (%)			65.00

Confusion Matrix accuracy data result for K value 7.

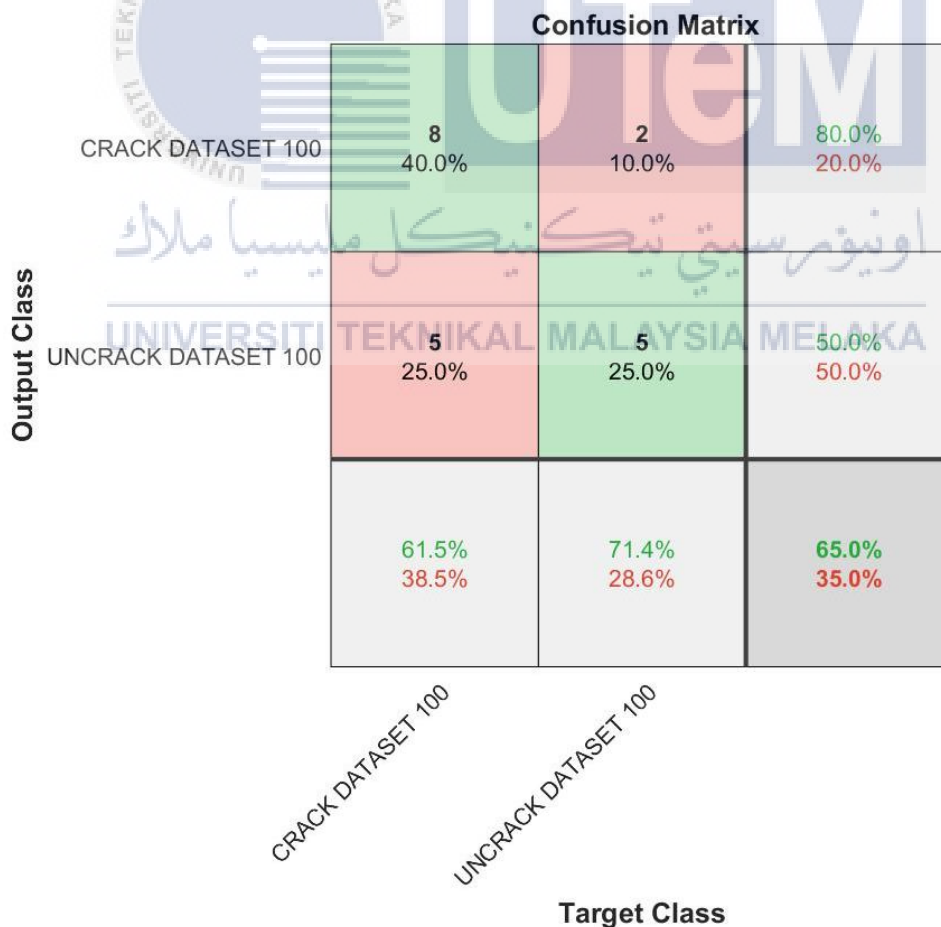


Figure 4.16: Confusion Matrix accuracy data result for K value 7.

KNN classification visualization result for K value 7.

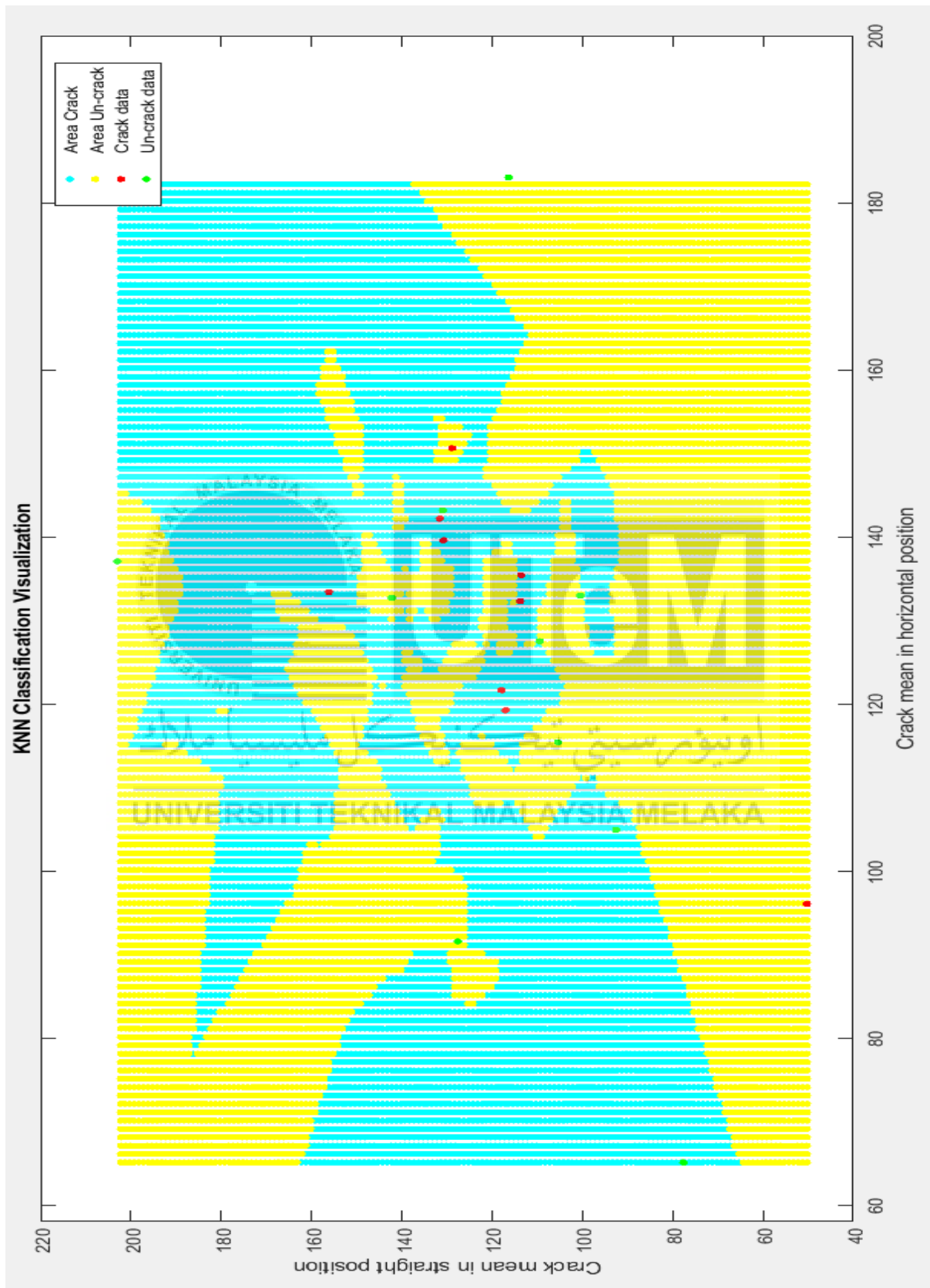


Figure 4.17: KNN classification visualization result for K value 7.

Summary of 90% Training And 20% Testing Dataset.

The highest crack detection accuracy for this testing process is 80.00%, using K value 3 and the greyscale intensity level is 0.04. The number of the testing dataset is 20 images. The correct crack class prediction for this accuracy is 80.00% correct crack image detection and 80.00% correct non-crack images detection. For K value 5, the classification accuracy is 70.00%, while for K value 7, the crack classification accuracy is 65.00%.

The dataset is trained and tested by using the different number of K values and different levels of greyscale intensity level. During this process, the highest crack classification accuracy is achieved. By analyzing the accuracy result, the best K value for the crack images detection is 3 and the level of greyscale level intensity is 0.4 is the most efficient value. For the testing and training percentage amount, the 90% training and 10% testing show a high accuracy value compared to 80% training and 20% testing. The highest accuracy value for this crack detection method is 80.00% accuracy by using the KNN value, which is 3 and the greyscale level is 0.4.

The correct crack class prediction accuracy is 80.00% correct, which refers to 8 out of 10 correct crack images prediction and 80% correct non-crack images class prediction. Thus, for 200 training datasets, the highest crack detection accuracy is 80%, using KNN value 3, the greyscale intensity level 0.4 and used 90% training and 10% testing dataset.

4.3.3 Crack Detection Testing by Using Different Value Training Dataset

From the previous testing accuracy data, the highest accuracy is 80% that use K value 3 and the amount of training is 90% from 200 datasets and 10% testing from 200 datasets. Thus, this parameter is used to perform a crack detection testing process that uses different amounts of crack and non-crack datasets. Since more than 70% accuracy is achieved in the previous testing, the next testing method using the different number of images category is

performed to observe the highest classification accuracy and the effect of different amounts of training datasets toward testing classification accuracy. Below is the dataset description and classification accuracy result.

1. Crack Images Dataset Is Higher Than Non-Crack Images Dataset.

The number of crack dataset used for algorithm training is 800 crack images and 100 non-crack images. The greyscale intensity level used is 0.04. From previous image processing and feature extraction result, greyscale intensity 0.4 show the cleared presence of crack compared to 0.2 and 0.6. Below is the classification accuracy data.

Table 4.7: Classification result for 800 crack dataset and 100 non-crack dataset.

Total Crack Image and Non-Crack for Testing		160
Crack Images Quantity for Training		800
Non-Crack Images Quantity for Training		100
K-Nearest Neighbour (Knn) Value		3
K-Nearest Neighbour (Knn) Training Dataset		90%
K-Nearest Neighbour (Knn) Testing Dataset		10%
Classification Time Taken (Second)		28.554
Classification Accuracy (%)		
Grayscale Intensity Level Adjustment	Crack	Non- Crack
0.4	88.80	100
Classification Accuracy for Random Dataset (%)		100%
		20 correct out of 20

From the accuracy result, the classification accuracy is 94.40%, which is shown to increase compared to the previous testing accuracy result when the high crack images dataset and low non-crack images dataset are used. The correct class prediction for crack images is 88.80%, which refers to 71 correct from the 80 crack dataset. While for non-crack images, the accurate class prediction is 100% that refer to 80 datasets. For random dataset testing accuracy, the accuracy is 100% correct prediction. The random dataset consists of 10 crack images and 10 non-crack images.

Confusion Matrix accuracy data result for high crack image dataset.

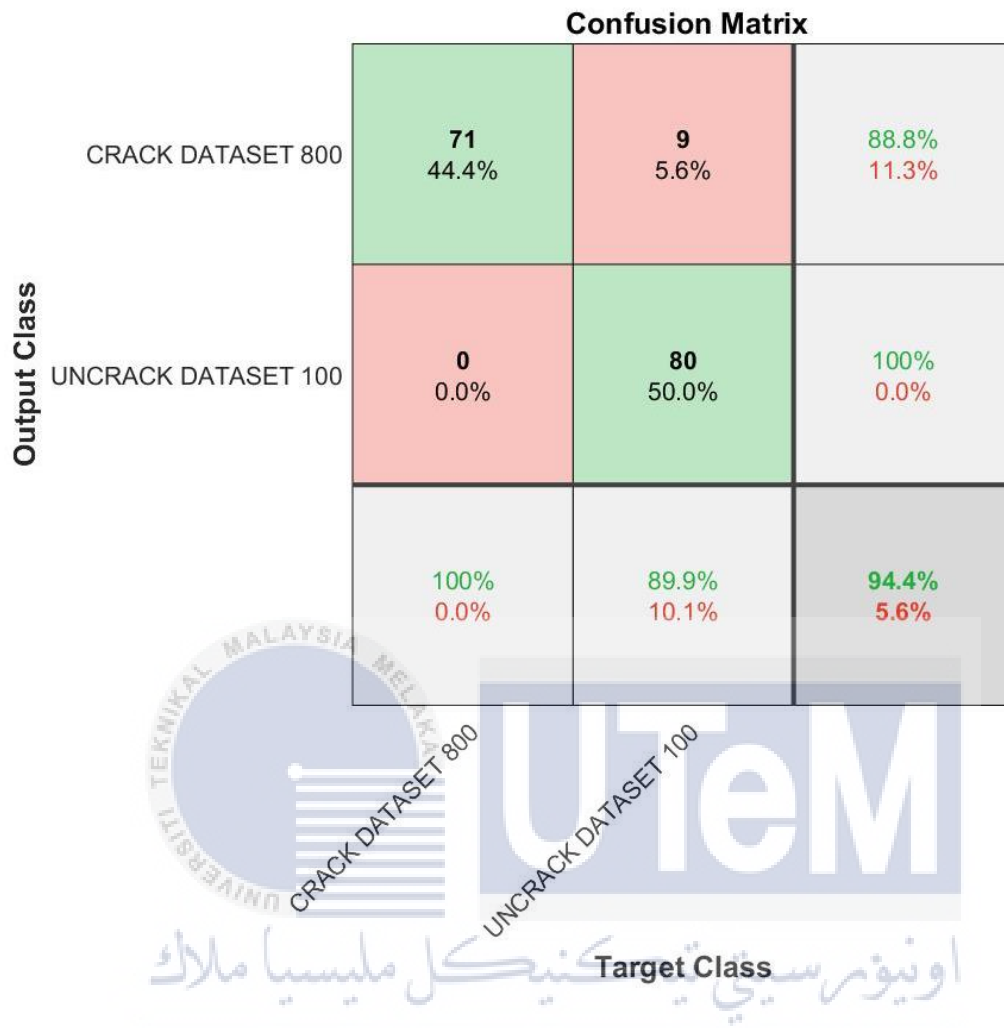


Figure 4.18: Confusion Matrix accuracy data result for high crack image dataset.

KNN classification visualization result for high crack image dataset.

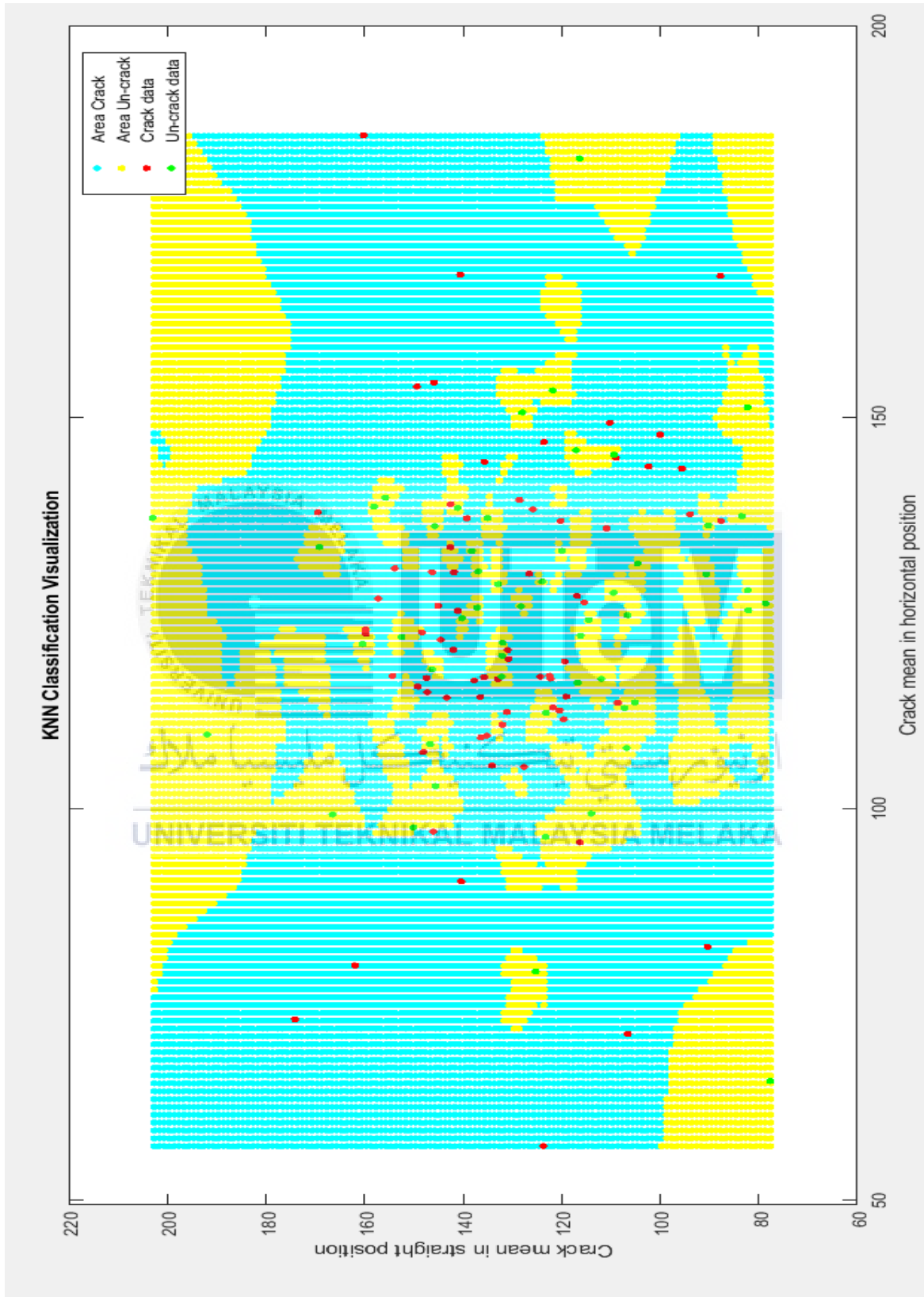


Figure 4.19: KNN classification visualization result for high crack image dataset.

2. Crack Images Dataset Is Lower Than Non-Crack Images Dataset.

The number of crack dataset used is 100 crack images and 800 non-crack images.

The greyscale intensity level used is 0.4. Below is the classification accuracy data.

Table 4.8: Classification result for 100 crack dataset and 800 non-crack dataset.

Total Crack Image and Non-Crack for Testing		160	
Crack Images Quantity for Training		100	
Non-Crack Images Quantity for Training		800	
K-Nearest Neighbour (Knn) Value		3	
K-Nearest Neighbour (Knn) Training Dataset		90%	
K-Nearest Neighbour (Knn) Testing Dataset		10%	
Classification Time Taken (Second)		26.876	
Classification Accuracy (%)			
Grayscale Intensity Level Adjustment	Crack	Non- Crack	Total Accuracy
0.4	98.60	76.20	87.50
Classification Accuracy for Random Dataset (%)			75.00
			15 correct out of 20

From the table, the classification accuracy is 87.50% correct class prediction which is low than the high crack image dataset accuracy which is 94.40%. For correct crack class dataset prediction, the accuracy is 98.60%, while for non-crack class prediction accuracy is 76.20%. For random dataset crack classification accuracy, 75% of datasets were predicted correctly, referring to 15 correct out of 20. Thus, the crack detection testing accuracy by using the high number of crack datasets for training, which is 800 is high compared to using a low dataset of crack.

Confusion Matrix accuracy data result for low crack image dataset.

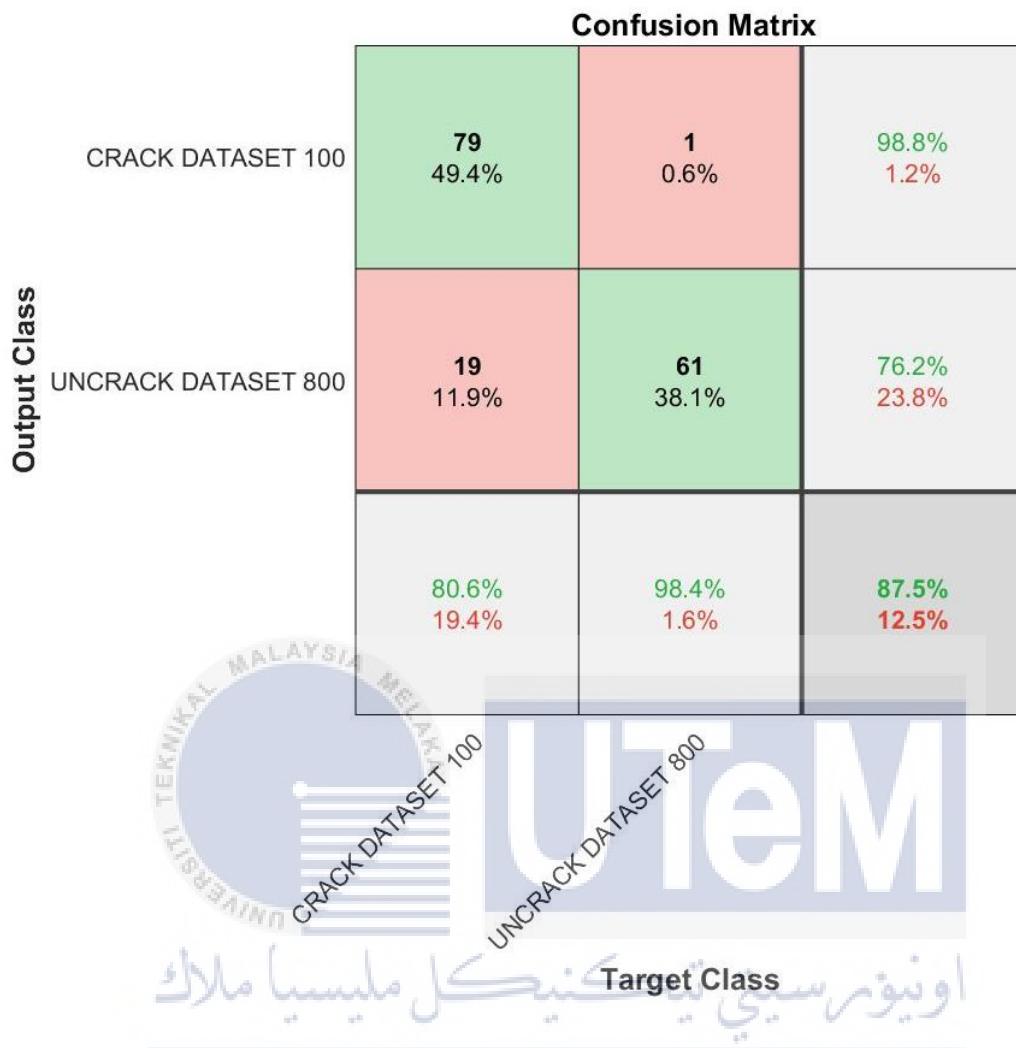


Figure 4.20: Confusion Matrix accuracy data result for low crack image dataset.

KNN classification visualization result for low crack image dataset.

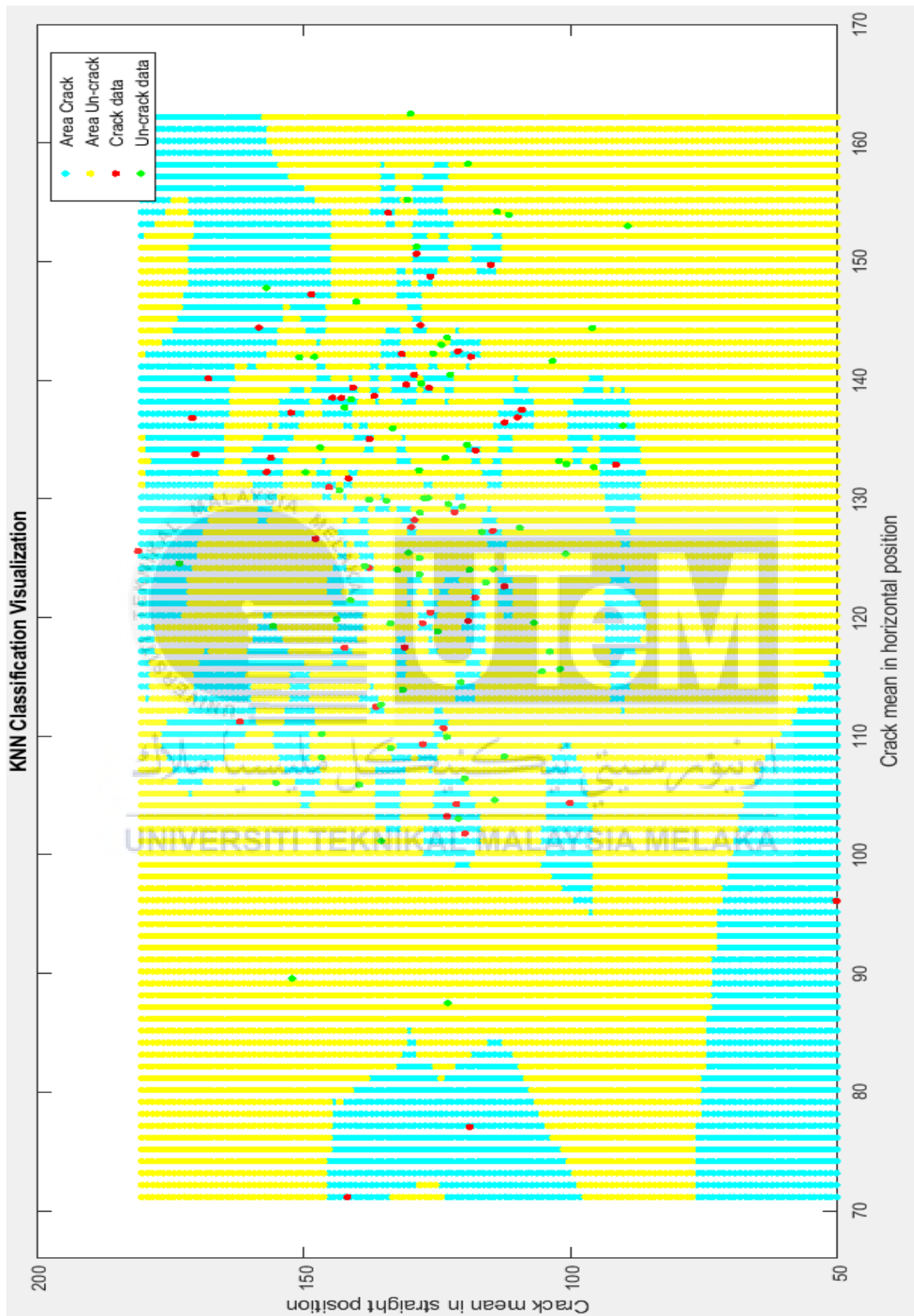


Figure 4.21: KNN classification visualization result for low crack image dataset.

By referring to the crack detection accuracy by using a high amount of crack dataset, the accuracy is higher compared to low crack dataset testing accuracy. The differences between the two accuracies are 6.90% which is 94.40% for the high crack images dataset and 87.50% for the low crack images dataset. Random image classification testing was also done to analyze the classification accuracy when crack data is placed together with non-crack data. The random images classification accuracy is higher when many crack images data is used for training compared to low crack images data is used for training. The accuracy is 100% correct when 20 random datasets are tested to detect the dataset class.

4.3.4 K- Nearest Neighbor (KNN) Visualization Discussion

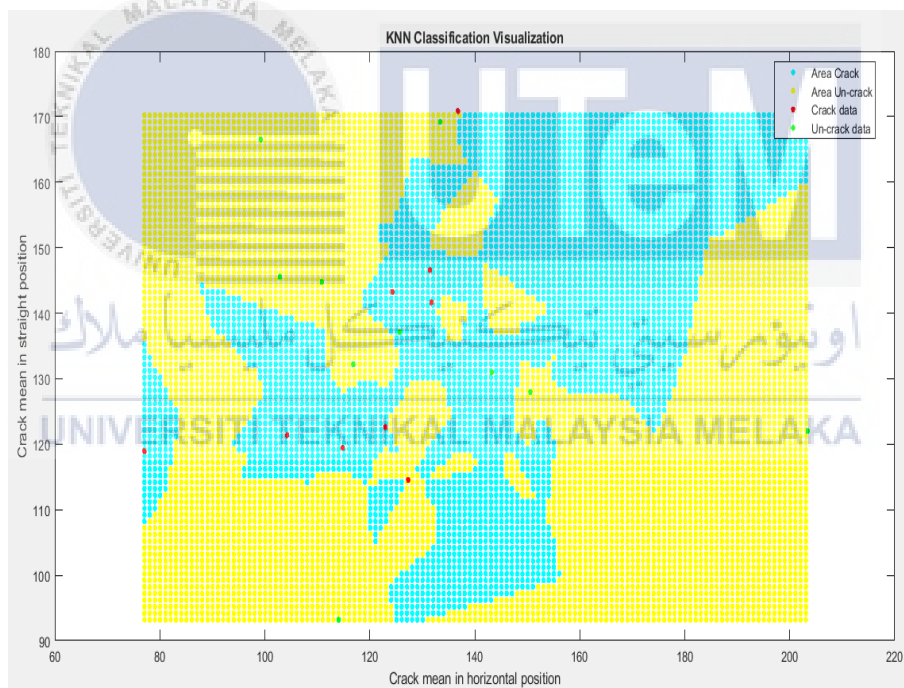


Figure 4.22: KNN classification visualisation for balance training dataset.

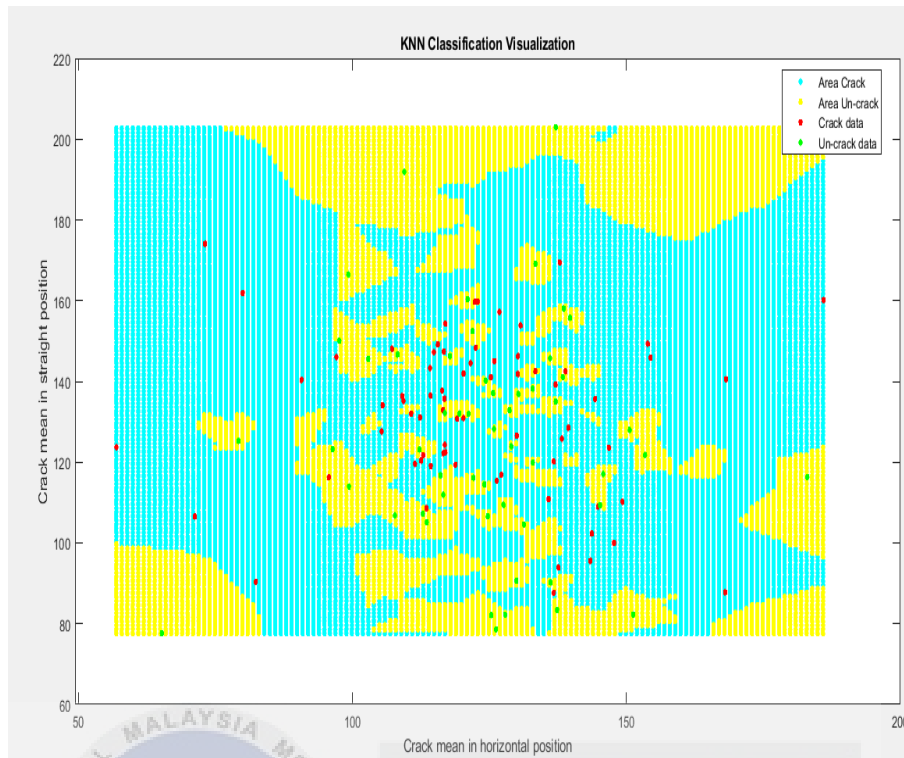


Figure 4.23: KNN classification visualisation for crack training dataset is high.

For KNN visualization analysis, two KNN visualization is chosen for discussion. The two graphs have the highest testing classification accuracy. The first KNN visualization graph is a graph of K value is 3, the number of the testing dataset is 20, including crack and non-crack. The number of the training dataset is 100 for crack and non-crack classes. From the result, 8 correct crack class predictions and 8 correct non-crack class predictions. This can be observed when only 2 red dots representing crack data were plotted on the yellow region representing non-crack prediction. The same result goes to the green dot that represents the non-crack dataset.

The blue and yellow regions show a balanced region size that is a balanced region area compared to KNN visualization that uses high crack data for training. The blue and yellow regions represent the average length of the white pixel at horizontal and vertical show balance region area because the number of the training dataset is equal. While for KNN visualization that uses high crack images than non-crack images quantity, the blue region

area is big compared to the yellow region that represents non- crack prediction. This is because the algorithm has obtained more information about the crack dataset from the training process compared to non-crack images that have less training dataset. The accuracy for the testing is 94.40%. The number of datasets for testing is 80 for each class. For correct crack prediction, 71 crack images are classified correctly while for the non-crack dataset, 80 non-crack images are predicted correctly.

From all the KNN visualization results, the blue and yellow regions show different forms when the different number of training datasets and K values are used during the testing process. This indicates that the information obtained when using different amounts of training dataset and K value during algorithm training affected the accuracy of testing prediction. When more training datasets are used, the algorithm can receive more crack and non-crack information, such as the number of white pixels and image brightness. This will make the KNN have high reference data to plot the region and obtain the average value of the white pixel in horizontal and vertical. The different K values will affect the correct class prediction accuracy. The unsuitable K value will make the testing dataset plotted at the wrong region since the number of nearest neighbors of the testing dataset is too high or low. Thus, the visualization graph shows a clear class prediction since the testing prediction result can be observed directly at the graph.

4.4 Classification Accuracy with Different K- Nearest Neighbor (KNN) Value

Analysis

By referring to the KNN algorithm classification accuracy result, the highest classification accuracy was achieved when the training number of the crack dataset is 800 and the non-crack dataset is 100 by using K value 3, the accuracy is 94.40%. For the equal number of crack and non-crack, the highest accuracy is 80.00%, which used the value of K equal to 3. Thus, this concludes that the best value of K suitable for concrete crack detection by using the KNN algorithm in this study is 3 since the 3 values have high accuracy levels compared to others by using the equal number of crack data set different values of the dataset. Below is the comparison graph of different K values and their highest accuracy.

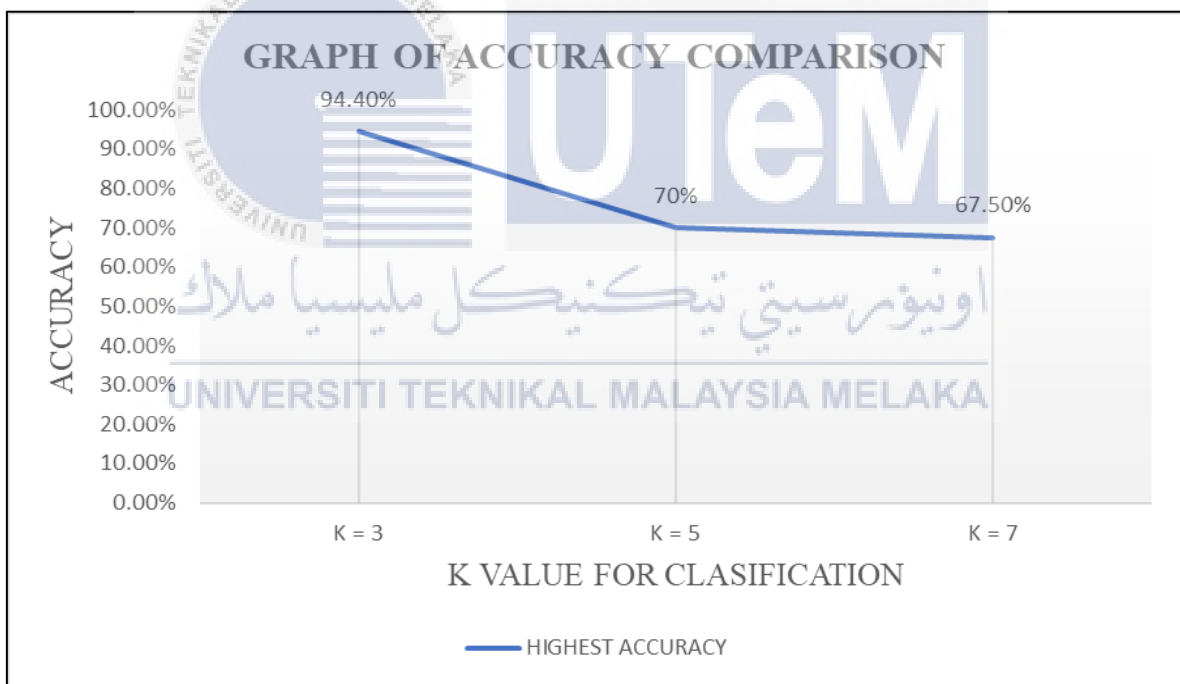


Figure 4.24: Graph of accuracy against the number of K value.

The higher the K value, the lower the correct class classification accuracy from the graph. The most insufficient accuracy is 67.50% that use K value 3 while the highest is 94.40% that use K value 7 and then followed by 70.00% that use a K value of 5. The important finding in this study is that the number of K values affected the crack detection

accuracy. To archive the highest accuracy for class prediction, a suitable number of K must be used during the testing. The accuracy drops when the K value is high because the closed Nearest Neighbor data is too many.

This will make the testing data information classified as the wrong class because the algorithm system had grouped the testing data in its closed Nearest Neighbor even though Nearest Neighbor data is from different testing data classes. The figure below shows class prediction when different K values are used.

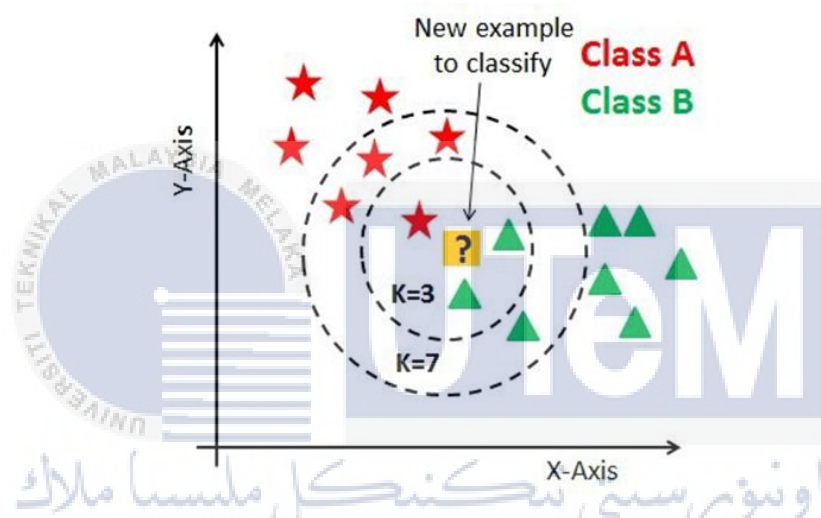


Figure 4.25: KNN clustering concept source from (Band, 2020).

By referring to the Chih et al., (2014) finding in “How the Parameters of K-Nearest Neighbor Algorithm Impact on the Best Classification Accuracy-In case of Parkinson Dataset” journal, the finding is the same that is the higher the number of K values for class prediction, the lower the correct class prediction accuracy. The best K value that has high classification accuracy for previous study is between 1 to 5 that is 84.00%. While this study, the best K value that have high classification accuracy is 3 that is 94.40% compared to 5 and 7.

4.5 Testing with Different Number Of Training Datasets Analysis

This testing method aims to improve the crack detection accuracy for testing that uses K value 3 and greyscale intensity level 0.4. The accuracy is 80.00%, which used an equal amount of crack and non-crack training datasets for training. This parameter is chosen because from the previous testing, this method produces high testing classification accuracy. The difference between crack and non-crack in the testing result shows different accuracy, which is an increase in accuracy level that is 94.40% when the number of the crack dataset is higher than the non-crack dataset used for training. The ratio of 90% training and 10% testing from overall dataset quantity is used in this method because the ratio has high accuracy during the previous testing that used an equal amount of crack and non-crack datasets. The accuracy comparison table and accuracy comparison graph when the different number of datasets is used for training is stated below.

Table 4.9: Accuracy result when number of crack dataset is higher for training process.

Training	Testing	Dataset class	Dataset quantity	Classification Accuracy
90%	10%	Crack	100	80.00%
		Non-crack		
		Crack	800	94.40%
		Non-crack	100	
		Crack	100	87.50
		Non crack	800	
Highest classification accuracy (%)				94.40%

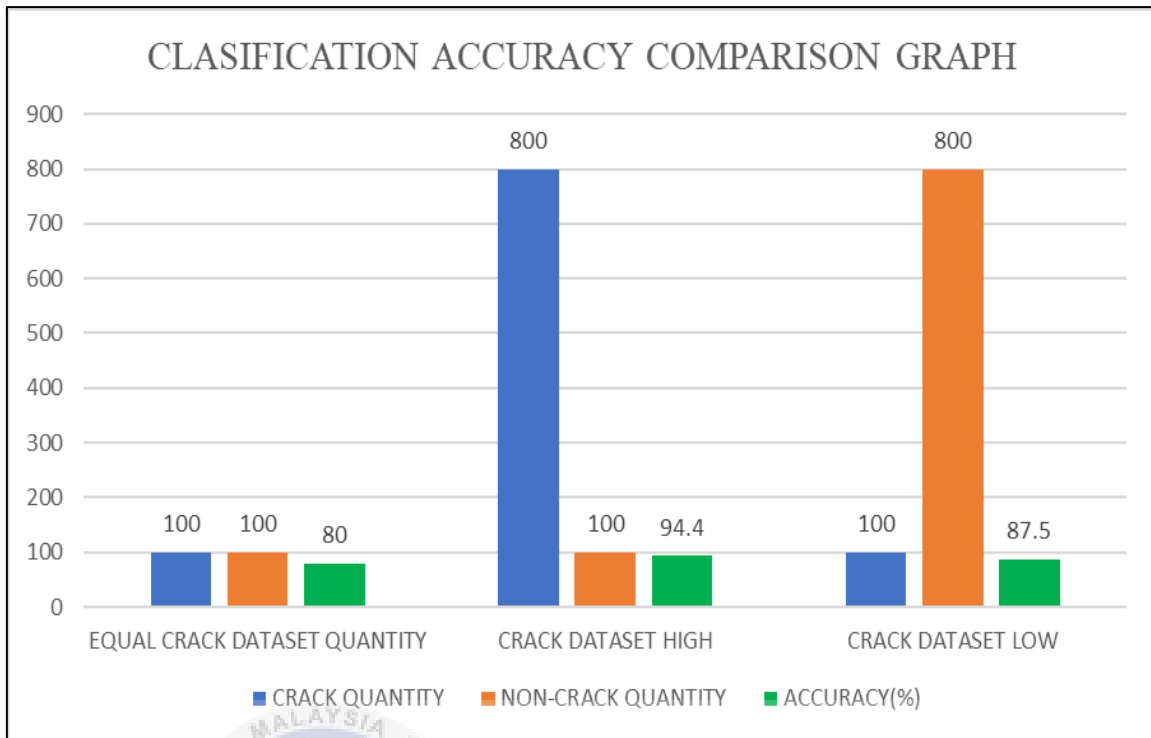


Figure 4.26: Graph of accuracy comparison for different number of datasets for training process.

From the comparison graph, the accuracy level shows an increasing 14.40% correct class prediction when high crack images are used for training compared to an equal amount of dataset. While for low crack image and high non-crack image classification accuracy, the value is lower than high crack images accuracy, which is 87.50%. Thus, it is proved that the number of datasets used for training affected the crack images detection accuracy. This work testing result shows a similar finding with Zhang et al. (2018). When less data is used for algorithm training, the accuracy for correct class prediction is low. While the number of training datasets is too high, the accuracy will drop. Thus, the correct number of training datasets is important to obtain the best classification accuracy. This is because when the number of primary class data which is the crack dataset, is higher than a second class which is non-crack, this will make the algorithm can extract more data about crack image information for training purpose since the crack dataset have more information compared to

a non-crack dataset such as different crack length and type of surface while, for the non-crack dataset that has only information about the surface.

Thus, this critical study finds that when the crack dataset is higher than the non-crack dataset during the training process, the crack detection testing accuracy will become higher than the same quantity of dataset. In this project, the ratio of 8:1 crack and non-crack dataset quantity is the best ratio to get an accuracy level of more than 90.00%. On the other hand, when more than 800 crack datasets are used for the training process, the classification accuracy will decrease and become inconsistent since the crack dataset quantity is too many, making the algorithm overtraining. This will make the classification process take a longer time to detect cracks and have low classification accuracy than 800 crack datasets. Below is the testing accuracy result when 1000 crack datasets are used for training.

Table 4.10: Accuracy result data when too high crack dataset is used for training process.

Total Crack Image and Non-Crack for Testing		200	
Crack Images Quantity for Training		1000	
Non-Crack Images Quantity for Training		100	
K-Nearest Neighbour (Knn) Value		3	
K-Nearest Neighbour (Knn) Training Dataset		90%	
K-Nearest Neighbour (Knn) Testing Dataset		10%	
Classification Time Taken (Second)		31.095	
Classification Accuracy (%)			
Grayscale Intensity Level Adjustment	Crack	Non- Crack	Total Accuracy
0.4	81.00	100	90.50
Classification Accuracy for Random Dataset (%)			80.00
			16 correct out of 20

The crack testing classification accuracy result achieved in this study was higher than Jinho et al. (2012), which is 87.24%. However, the number of datasets that Jinho et al. (2012) used is equal in each class. Thus, to get high accuracy for crack concrete detection using the KNN algorithm, the crack dataset must be higher than the number of the non-crack concrete

dataset for training. For example, in this work the training number of the crack dataset is 800 and the non-crack dataset is 100.

4.6 Confusion Matrix Testing Result

The confusion matrix result for data 90% training and 10% testing, using 3 K value and 800 crack dataset and 100 non-crack datasets for algorithm training, is chosen for discussion and analysis because the data show the highest accuracy compared to other results. The number of testing dataset in this classification test is 80 crack and non-crack images. The accuracy result is 94.40%. By referring to Ajay et al. (2020), the “True Positive (TP)” is referred to as correct primary data classification.

For this project, the TP is represented by correct crack classification. While “True Negative (TN)” is represented by correct secondary data prediction, which is correct non-crack class prediction. Ajay et al. (2020) also stated that precision, recall, and accuracy could be analyzed from the confusion matrix plot.

All of the three parameter is analyzed in this project. The precision is the total number of correct crack classification divided by the total number of predicted as a crack which is correct crack prediction and wrong crack prediction. The recall parameter for this work is the total number of correct crack predictions divided by the total number of actual crack data. Lastly, the accuracy for this project is the total number of correct class predictions divided by the total number of classification data. Below is the confusion matrix accuracy result.

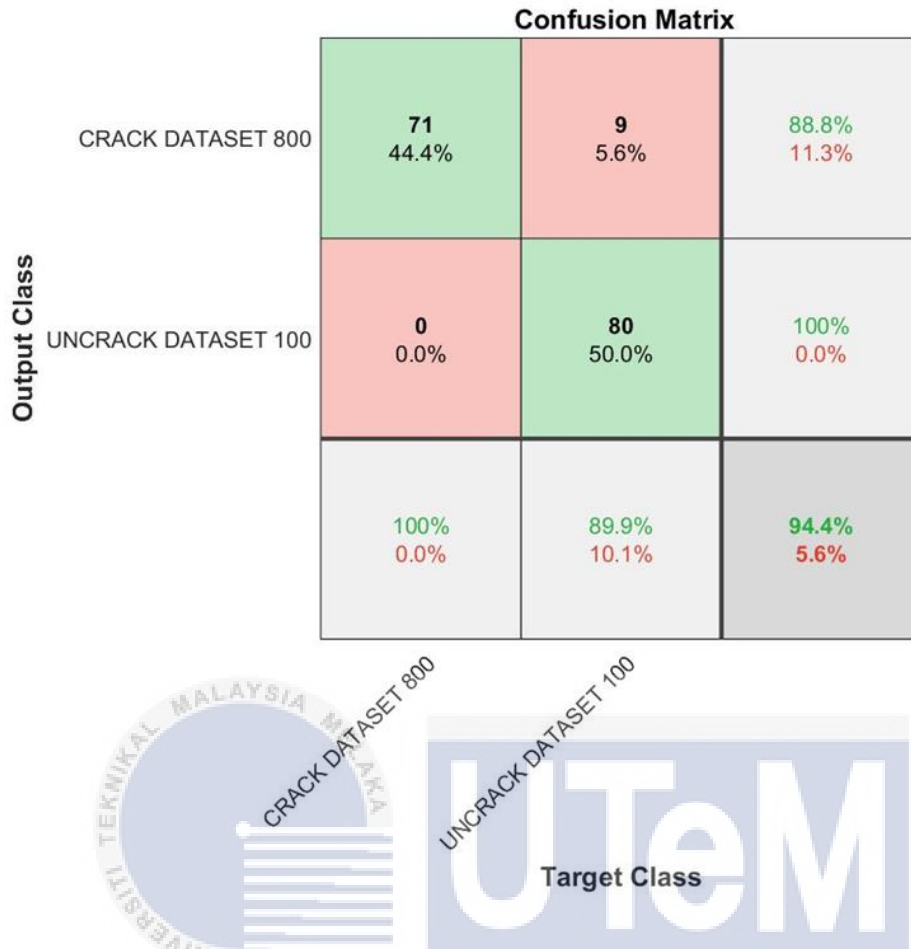


Figure 4.27: Confusion Matrix accuracy result for K value 3.

The dataset used for training is a different amount for each class, thus for the crack dataset, 90% of training is equal to 720 crack images and 10% is equal to 80 crack images. On the other hand, for the non-crack dataset that is 100 datasets, 90% training refer to 90 non-crack images and 10% is equal to 10 images. Thus, the “Oversampling” method is used to balance the amount of training dataset. In this method, the value for 10% in each class is compared and the higher value that refers to 10% is used. In this case, the 80 datasets that represent 10% is used for testing.

From the confusion matrix result, 80 image samples were tested for class classification: crack or non-crack. The percentage for correct class classification for crack images is 88.80%, which refers to 71 images from the 80 crack images dataset and 11.30% refers to 9 incorrect crack images classification. For non-crack images classification testing

accuracy, 100% of non-crack images that represent 80 non-crack samples is predicted correctly according to their true class, while 0% were classified wrongly as crack images. Thus, the overall classification accuracy for crack and non-crack classification is 94.40% classified correctly and 5.60% wrongly classified. Figure 4.28 is the table for the calculation of precision, recall and classification accuracy.

		TRUE CLASS CATEGORY	
		TRUE	FALSE
PREDICTED CLASS	TRUE	TP (True Positive) <i>Corect result</i>	FP (False Positive) <i>Unexpected result</i>
	FALSE	FN (False Negative) <i>Missing result</i>	TN (True Negative) <i>Corect absence of result</i>

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 4.28: Formula for precision, recall and accuracy calculation source from (DataQ, 2013).

Table 4.11: Data of precision, recall and accuracy.

Classification Result	Formula	Calculation	Result (%)
Precision	$\frac{TP}{TP + FP}$	$\frac{71}{71 + 9} \times 100$	88.80
Recall	$\frac{TP}{TP + FN}$	$\frac{71}{71 + 0} \times 100$	100
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	$\frac{71 + 80}{71 + 80 + 9 + 0} \times 100$	94.40

From the data table, the precision of this study is 88.80% that refer to correct crack class prediction and the recall is 100%. The highest accuracy for concrete crack detection by using KNN algorithm in for this method is 94.40% correct. Thus, by comparing all the confusion matrix data from the previous crack detection testing, the highest accuracy for the testing is 94.40 correct crack and non-crack class prediction when the KNN algorithm method is used to classify the images.

4.7 Random Images Testing Result

From the crack classification accuracy that uses the high number of the crack dataset and low non-crack dataset for training, 20 random images consisting of 10 crack images and 10 non-crack images were tested using the same algorithm parameter to predict the dataset class. The testing image is selected manually from training dataset file for both categories. While for previous testing process, the algorithm randomly selected the testing dataset. The testing parameter that had been used is K value 3, 90% training dataset and 10% testing dataset. The greyscale intensity level is 0.4 and the training crack dataset is 800 and the non-crack dataset is 100. The testing result shows a 100% correct class prediction using the stated parameter.

Below is the random images classification result by using 90% training and 10% testing dataset. The K value that had been used is 3 and the crack image and non-crack image dataset numbers for the random testing are 10 for each class. The class prediction is 100% correct when using 800 crack images and 100 non-crack images for training.

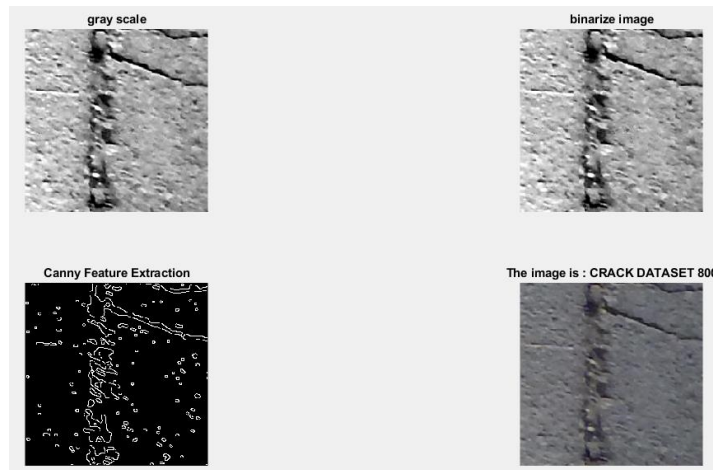


Figure 4.29: Correct severe crack dataset class prediction.

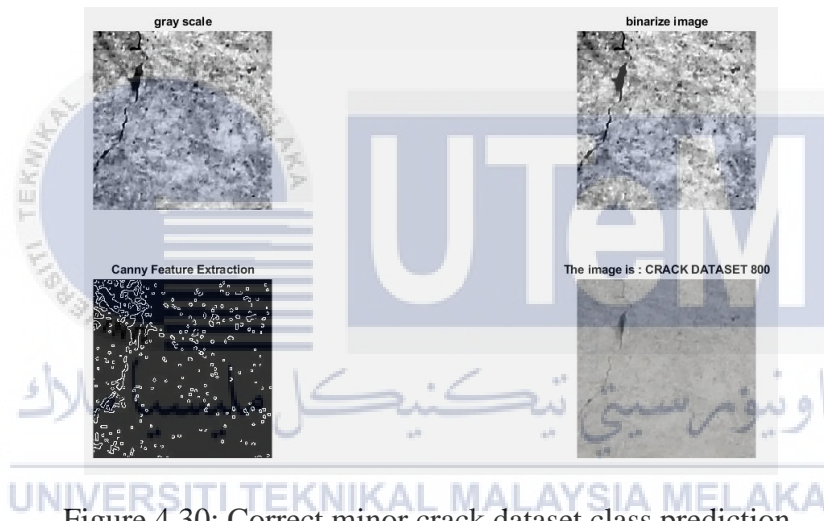


Figure 4.30: Correct minor crack dataset class prediction.

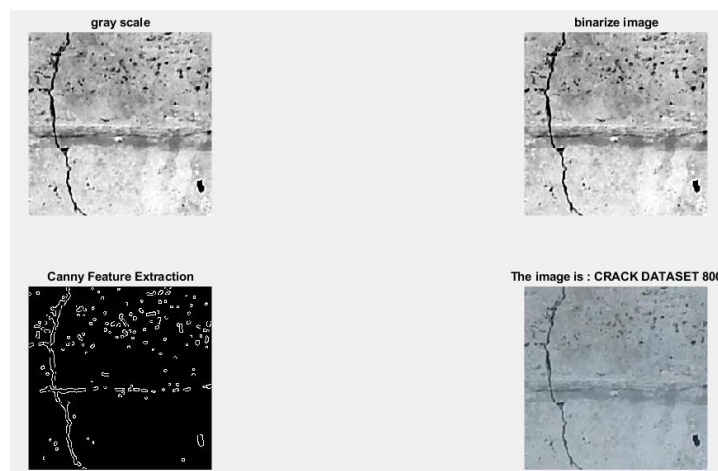


Figure 4.31: Correct moderate crack dataset class prediction.

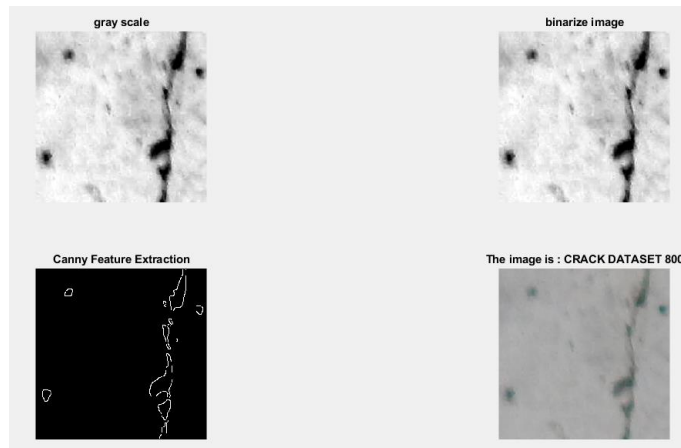


Figure 4.32: Correct blur crack dataset class prediction.

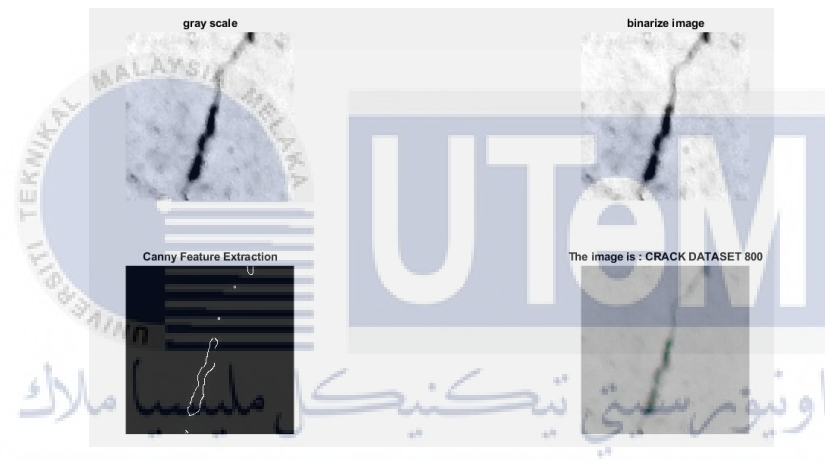


Figure 4.33: Correct moderate crack dataset class prediction.

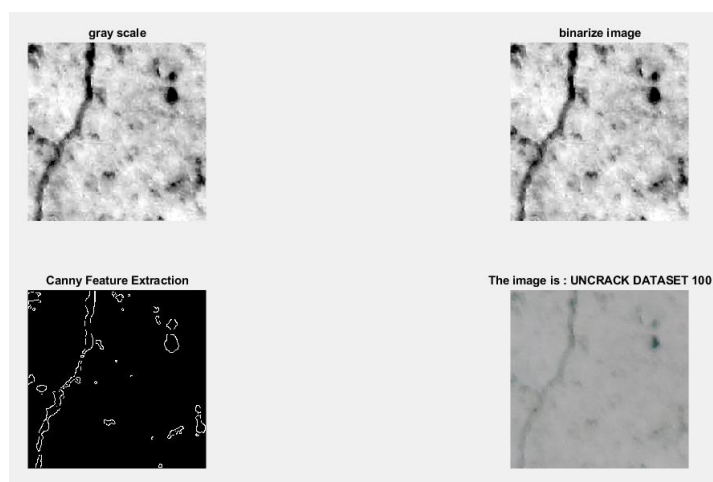


Figure 4.34: Correct blur crack dataset class prediction.

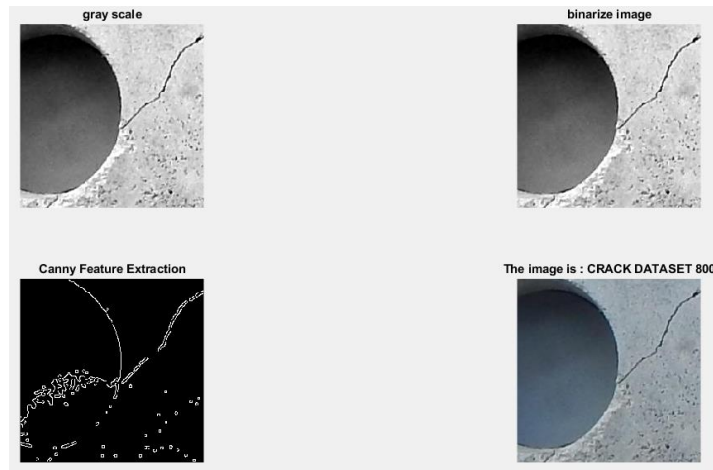


Figure 4.35: Correct obstacle crack dataset class prediction.



Figure 4.36: Correct moderate crack dataset class prediction.

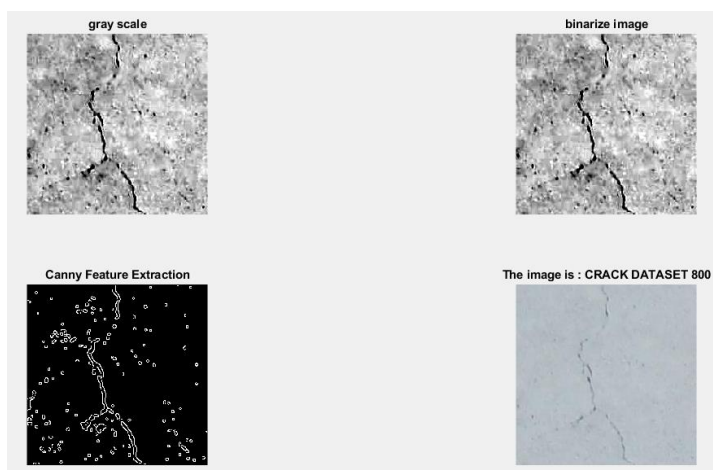


Figure 4.37: Correct severe crack dataset class prediction.

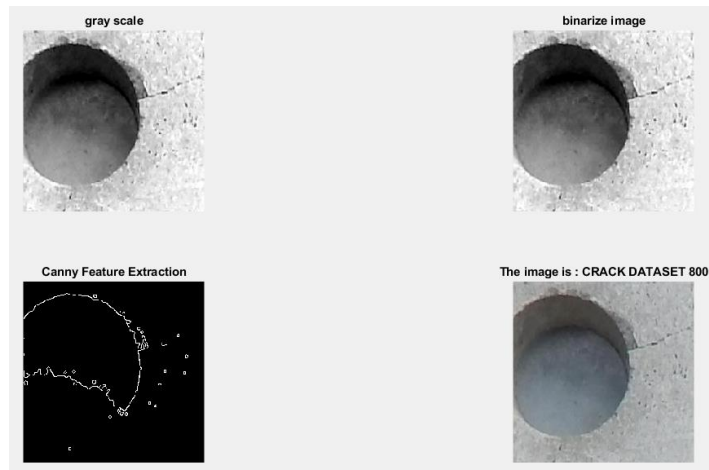


Figure 4.38: Correct obstacle crack dataset class prediction.



Figure 4.39: Correct class prediction for obstacle non-crack dataset.

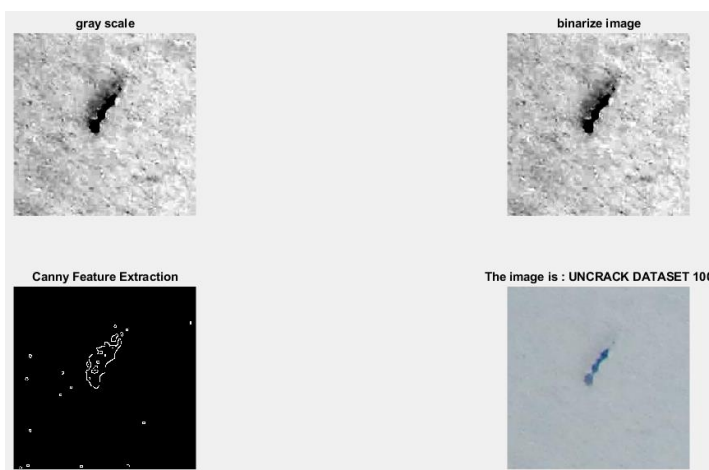


Figure 4.40: Correct class prediction for non-crack dataset.

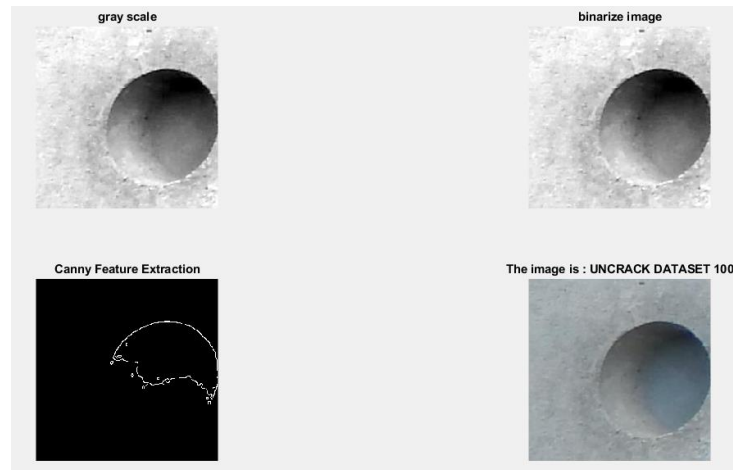


Figure 4.41: Correct class prediction for obstacle non-crack dataset.

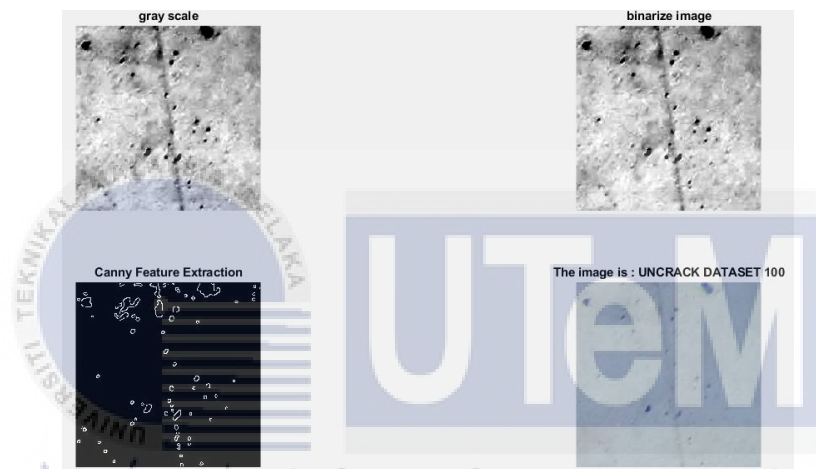


Figure 4.42: Correct class prediction for rough surface non-crack dataset.

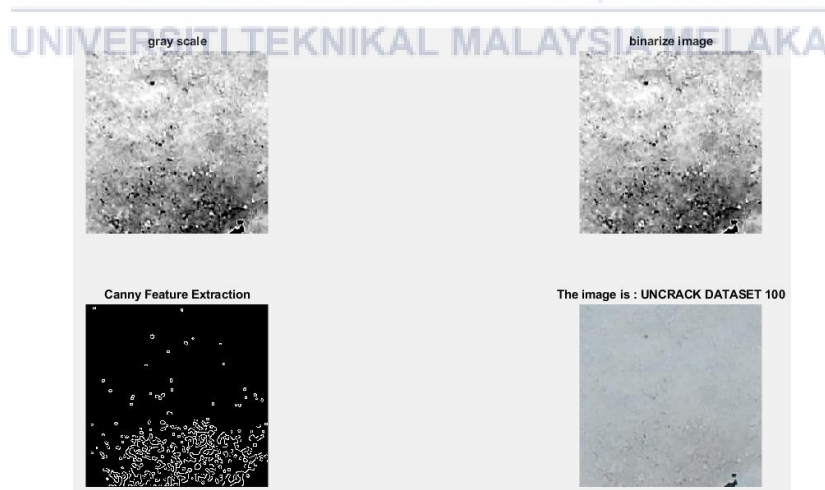


Figure 4.43: Correct class prediction for smooth surface non-crack dataset.

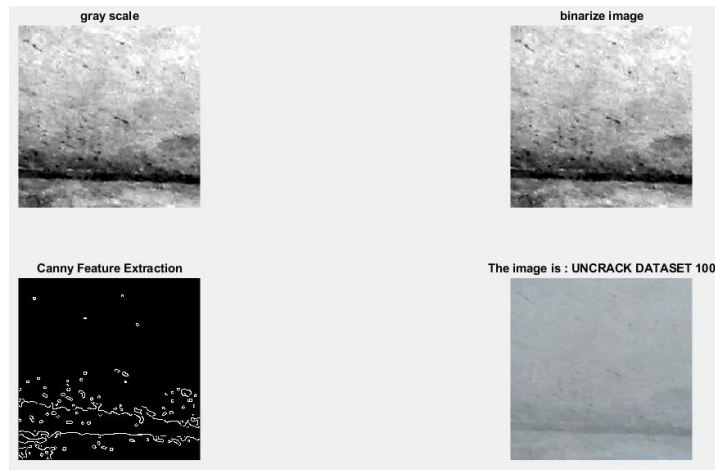


Figure 4.44: Correct class prediction for smooth surface non-crack dataset.

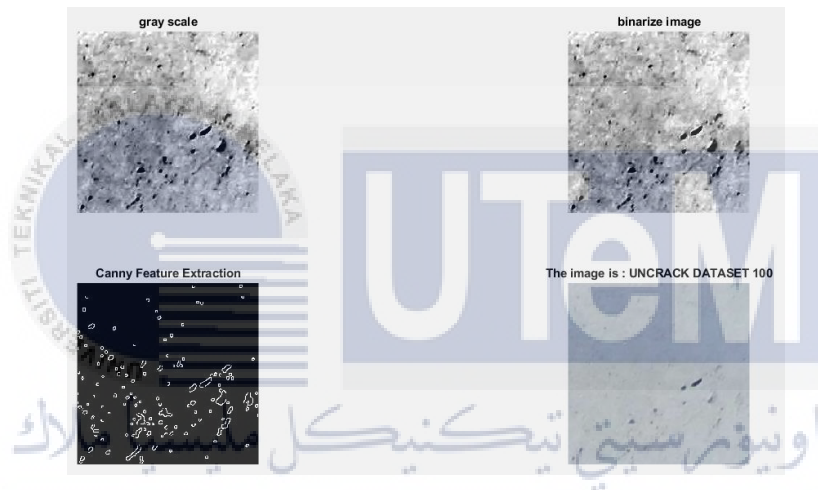


Figure 4.45: Correct class prediction for rough surface non-crack dataset.

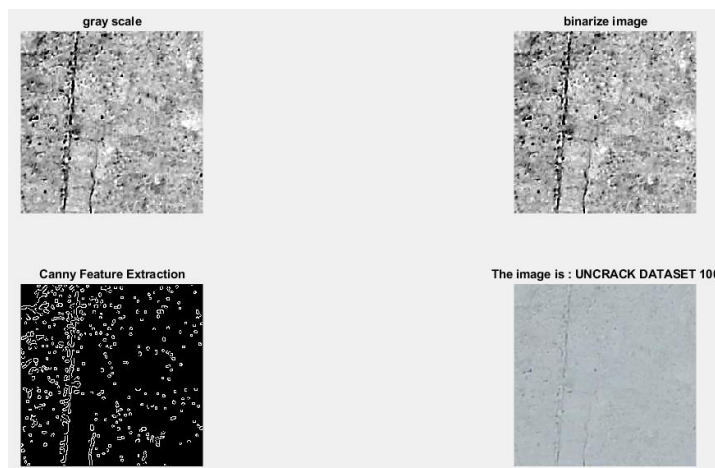


Figure 4.46: Correct class prediction for smooth surface non-crack dataset.

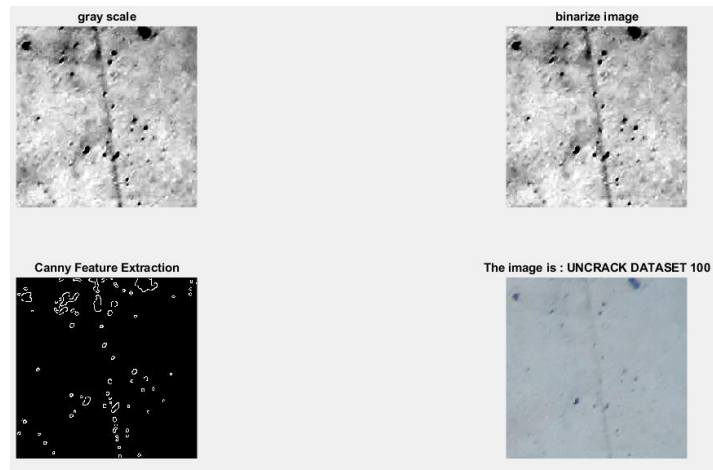


Figure 4.47: Correct class prediction for rough surface non-crack dataset.

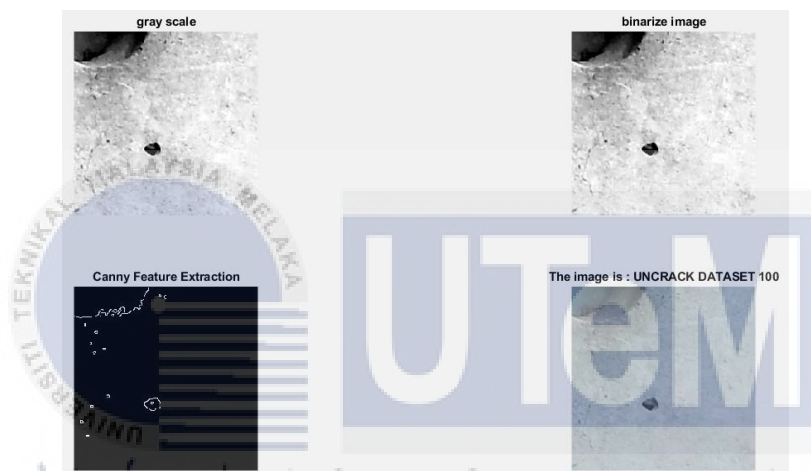


Figure 4.48: Correct class prediction for obstacle non-crack dataset.

As the conclusion from the random images testing, the best crack detection accuracy by using the KNN algorithm that had been carrying carry out from this project is 94.40% that use K value 3, the greyscale intensity level is 0.4, the training amount is 90% and testing is 10% from the dataset. The number of crack datasets used for this project to get the high accuracy is 800 crack datasets for training and 100 non-crack datasets for training and 160 testing datasets. This method also shows successful findings when 100% accuracy was achieved when the algorithm is tested by using random images consisting of 10 crack images and 10 non-crack images combined in 1 file.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In conclusion, for crack concrete detection by using K- Nearest Neighbor (KNN) algorithm, there are several factors that affect the correct classification accuracy. The first factor is the suitable greyscale intensity level that needs to be done before the dataset undergoes the feature extraction process. For this study, the best value for greyscale intensity level is 0.4, where the presence of crack structure is clear, and the number of white pixels is not too high, allowing the algorithm to differentiate between the crack data and the noises data on the images.

Next, the most efficient K value for the KNN algorithm to perform the detection process in this work is 3. This is proven when the value of 3 is used for the classification of crack, the highest accuracy was archived by using the same amount of crack dataset and non-crack dataset for training process. The accuracy is 80.00% correct class prediction. When the number of K that represent nearest neighbor of testing data is too high, this will make the KNN algorithm will wrongly predicted the testing data class since it has too much references data which is the nearest neighbor.

The third is, when the ratio of 90.00% training and 10.00% testing dataset is used for crack detection process, the highest accuracy that had been archived is 88.80% correct class classification for crack dataset and 100% correct class classification for the non-crack dataset. The number of training datasets that produce high accuracy for crack detection had been carried out in this study, that is 800 crack images and 100 non-cracks for the training

process. The accuracy is 94.40% correct crack detection. Thus, the highest accuracy for crack detection using the KNN algorithm achieved in this project is 94.40% correct crack concrete detection. Therefore, the objective of this project is successfully achieved, which is the suitable image processing method to enhance the presence of crack is using 0.4 greyscale intensity level and the accuracy of correct crack concrete detection by using KNN algorithm is 94.40% correct.

5.2 Recommendation for Future Work

After the training and testing process was done for crack concrete detection by using the KNN algorithm, there are several improvements can be made to increase the application efficiency of this method for further research. Firstly, different sizes of the crack dataset can be used for the crack detection process. In this study, the size of the dataset that had been used is the same for both categories that are gathered from SDNET2018 only because of the limitation of access crack concrete area due to COVID 19 pandemic and safety factors. The size of the dataset that had been used is 256 x 256 pixels only. For future work, different sizes can be used such as 500 x 500 pixels since most modern cameras have high image pixel size. This will make this method become more efficient for on field applications.

Secondly, the next improvement that can be done is to change from training an offline dataset to an online dataset. For this study, the dataset is trained and tested offline. This will make only one user can perform crack detection per time and the data cannot be accessed by anyone instead of the user only. To make many users can perform the detection process per time, the online platform that stores the crack database should be created such as using online data cloud "IDrive". This will make many cracks detection test can be performed per time and reduce the detection process time since everyone can perform since everyone can access the database.

REFERENCES

Ajay, Chong, D., & Batarseh, F. A. (2020). Foundations of data imbalance and solutions for a data democracy. In *Data Democracy: At the Nexus of Artificial Intelligence, Software Development, and Knowledge Engineering*. Elsevier Inc. <https://doi.org/10.1016/B978-0-12-818366-3.00005-8>

Amlan, & Devdas. (2000). Effect of Concrete Compressive Strength With Various Natural Additives Fiber for Green Environment. *Journal Civil Engineering and Earth Resources*, November, 24. <https://doi.org/10.1080/23311916.2020.1870790>

Arun, & Sumathi. (2017). Crack detection using image processing : A critical review and analysis. *Alexandria Engineering Journal*, 57(2), 787–798. <https://doi.org/10.1016/j.aej.2017.01.020>

Bir, P. (2019). Image Classification with K Nearest Neighbours. <https://doi.org/10.37200/IJPR/V24I5/PR2020214>

Chih-Min, M., Wei-Shui, Y., & Bor-Wen, C. (2014). How the Parameters of K-Nearest Neighbor Algorithm Impact on the Best Classification Accuracy-In case of Parkinson Dataset. In *Journal of Applied Sciences* (Vol. 14, Issue 2, pp. 171–176). <https://doi.org/10.3923>

DataQ. (2013). Perbedaan: precision, recall & accuracy. Data's Base.

<https://dataq.wordpress.com/2013/06/16/perbedaan-precision-recall-accuracy/>

Fujita, Y. (2006). A Method for Crack Detection on a Concrete Structure Ube National College of Technology. Conference: Pattern Recognition, 2006. ICPR 2006. 18th International Conference On, 1, 18–21. <https://doi.org/10.1109/ICPR.2006.98>

Ibrahim, A., Osman, M. K., Yusof, N. A. M., Ahmad, K. A., Harun, N. H., & Raof, R. A. A. (2019). Characterization of cracking in pavement distress using image processing techniques and k-Nearest neighbour. 14(2), 810–818. <https://doi.org/10.11591/ijeecs.v14.i2.pp810-818>

Imandoust, S. B., & Bolandraftar, M. (2013). Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events : Theoretical Background. Engineering Research and Application, 3(5), 605–610. <https://doi.org/2746-2752>

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Jason, B. (2020). What is a Confusion Matrix in Machine Learning. Code Algorithms From Scratch. <https://machinelearningmastery.com/confusion-matrix-machine-learning/>

Jinho, Kim, B., Savarese, S., & Arbor, A. (2012). Comparing Image Classification Methods : K-Nearest-Neighbor and Support-Vector-Machines. Computer Engineering and Application, 6, 133–138. <https://dl.acm.org/doi/10.5555/2209654.2209684>

Jitendra, M. S. N. V, Srinivasu, P. N., A, S. S., Nithya, A., & Kiran, S. (2020). Crack Detection on Concrete Images Using Classification Techniques in Machine Learning. *Critical Review*, 7(9), 1236–1241. <https://doi.org/10.31838/jcr.07.09.224>

Kaish, A. B. M. A., Woon, C., & Raman, S. N. (2018). applied sciences Early-Age Cracking in Concrete : Causes , Consequences , Remedial Measures , and Recommendations. September. <https://doi.org/10.3390/app8101730>

Koch, C., & Brilakis, I. (2011). Automated Pothole Distress Assessment Using Asphalt Pavement Video Data. *Journal of Computing in Civil Engineering*, 27(4), 370–378. [https://doi.org/10.1061/\(asce\)cp.1943-5487.0000232](https://doi.org/10.1061/(asce)cp.1943-5487.0000232)

Larosche, C. J. (2009). Types and causes of cracking in concrete structures. In *Failure, Distress and Repair of Concrete Structures: Vol. c*. Woodhead Publishing Limited. <https://doi.org/10.1533/9781845697037.1.57>

Maguire, Dorafashan, & Thomas. (2018). SDNET2018: A concrete crack image dataset for machine learning applications. *SDNET2018: Structural Defects Network 2018*, 186. <https://doi.org/https://doi.org/10.15142/T3TD19>

Maurits, K. (2020). Silo surface maintenance using drones. Tilburg University. <https://towardsdatascience.com/silo-surface-maintenance-using-drones-15d6c3f2439b>

Nhat. (2018). Detection of Surface Crack in Building Structures Using Image Processing Technique with an Improved Otsu Method for Image Thresholding. *Advances in Civil Engineering*, 2018. <https://doi.org/10.1155/2018/3924120>

Rakesh & Khachane, (2012). Automatic Medical Image Classification and Abnormality Detection Using K-Nearest Neighbour. *International Journal of Advanced Computer Research*.

https://www.researchgate.net/publication/305403850_Automatic_Medical_Image_Classification_and_Abnormality_Detection_Using_K-Nearest_Neighbour

Shin, T. (2020). Understanding the Confusion Matrix and How to Implement it in Python. *KOHO | Top*. <https://towardsdatascience.com/understanding-the-confusion-matrix-and-how-to-implement-it-in-python-319202e0fe4d>

Sumera Sultana, & R. Ganesh. (2015). FPGA Implementation of Binary Morphological Processing for Image Feature Extraction. *International Journal of Engineering Research And*, V4(10). <https://doi.org/10.17577/ijertv4is100506>

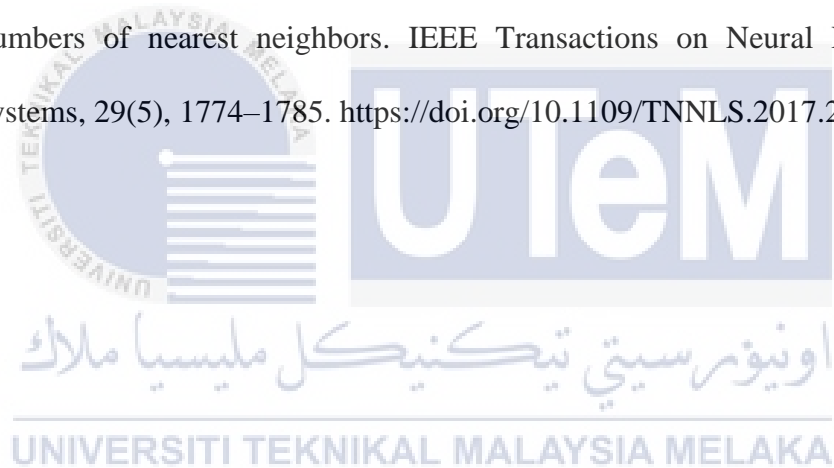
Varsha, K. S., Akhil, T., & Surya Teja, P. (2019). Crack detection on concrete surfaces using digital image processing. *Jianzhu Jiegou Xuebao/Journal of Building Structures*, 35, 356–361. <https://doi.org/10.14006/j.jzjgxb.2014.S2.052>

Vijay, & Bala. (2019). Confusion Matrix model evaluation. In *Data Science Practitioner*. Elsevier Inc. <https://doi.org/10.1016/B978-0-12-411511-8.00006-2>

Wan, X., Wang, W., Liu, J., & Tong, T. (2014). Estimating the sample mean and standard deviation from the sample size, median, range and/or interquartile range. *BMC Medical Research Methodology*, 14(1). <https://doi.org/10.1186/1471-2288-14-135>

Wenyu, Zhenjiang, Dapeng, & Yun. (2014). Automatic Crack Detection and Classification Method for Subway Tunnel Safety Monitoring. 19307–19328.
<https://doi.org/10.3390/s141019307>

Zhang, S., Li, X., Zong, M., Zhu, X., & Wang, R. (2018). Efficient kNN classification with different numbers of nearest neighbors. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5), 1774–1785. <https://doi.org/10.1109/TNNLS.2017.2673241>



APPENDICES

A. MATLAB Code for Training and Testing.

```
clear;clc;close all

% To recall the dataset address for training and testing.
DBPath = 'C:\Program Files\MATLAB\code\dataset';
imgSet =
imageDatastore(DBPath,'IncludeSubfolders',true,'LabelSource','foldernames');

% To balanced the number of dataset for testing by using oversampling.
labels=imgSet.Labels;
[G,classes] = findgroups(labels);
numObservations = splitapply(@numel,labels,G);
desiredNumObservationsPerClass = max(numObservations);
files =
splitapply(@(x){randReplicateFiles(x,desiredNumObservationsPerClass)},imgSet.Files,G);
files = vertcat(files{:});
labels=[];info=strfind(files,'\');
for i=1:numel(files)
    idx=info{i};
    dirName=files{i};
    targetStr=dirName(idx(end-1)+1:idx(end)-1);
    targetStr2=cellstr(targetStr);
    labels=[labels;category(targetStr2)];
end
imgSet.Files = files;
imgSet.Labels= labels;

%To split the balance dataset in ratio 90:10
[imgSetTrain, imgSetTest]=splitEachLabel(imgSet,0.9);
```

***** Dataset Training Pre-process *****

preprocess training image

%To get all training data

```
X = imgSetTrain.Files;
```

%To get all training labelling

```
Ytrain = imgSetTrain.Labels;
```

```
Xtrain=[];
```

```
for i=1:length(X) % To get all the number of training dataset.
```

```
    I = imread(X{i, 1}); % To read image
```

```
    Istrech = imadjust(I,stretchlim(I));
```

```
    Igray_s = rgb2gray(Istrech); %convert to gray scale
```

```
    level = 0.4; % To set the greyscale intensity level.
```

```
    Ithres = imbinarize(Igray_s,level); % Image binarization process.
```

```
    Ithres = ~Ithres;
```

```
    BW = bwmorph(Ithres,'clean',10); % To detect edge of crack
```

```
BW11=edge(BW,'canny');
```

```
    [y, x] = find(BW11);
```

```
    x1 = min(x); x2 = max(x); z=mean(x); % To get the crack minimum, maximum  
and average length.
```

```
    y1 = min(y); y2 = max(y); z2=mean(y);
```

```
    delta_x = x2 - x1;%To detect position crack logitude/lotitude
```

```
    delta_y = y2 - y1;%To detect position crack logitude/lotitude
```

```
    Xtrain(:,i)=[delta_x,delta_y,x1,y1,x2,y2,z,z2];% To get all feature for  
the image
```

```
End
```

```
mdl = fitcknn(Xtrain',Ytrain,'NumNeighbors',3,'Standardize',1); % Train  
feature by using KNN
```

```

***** Dataset Testing Pre-Process *****
X2 = imgSetTest.Files;
Ytest = imgSetTest.Labels;
Xtest=[];
for i=1:length(X2)
    I2 = imread(X2{i, 1});
    Istrech2 = imadjust(I2,stretchlim(I2));
    Igray_s2 = rgb2gray(Istrech2);
    level2 = 0.4;

    Ithres2 = imbinarize(Igray_s2,level2);
    Ithres2 = ~Ithres2;
    BW2 = bwmorph(Ithres2,'clean',10);

    BW22=edge(BW2,'canny');

    [y2, x2] = find(BW22);
    x12 = min(x2); x22 = max(x2);z2=mean(x2);
    y12 = min(y2); y22 = max(y2);z22=mean(y2);
    delta_x = x22 - x12;
    delta_y = y22 - y12;
    Xtest(:,i)=[delta_x,delta_y,x12,y12,x22,y22,z2,z22];
end

Ypred = mdl.predict(Xtest');% To predict the testing result data based on
training dataset in mdl.

overallaccuracy = accuracy_score(Ypred,Ytest); % To get overall accuracy by
predict the correct class classification.

saveas(plotconfusion(Ypred,Ytest),'C:\Program Files\MATLAB\code\cm.jpg') % To
plot confusion matrix based on classification accuracy data.

```

```

function files = randReplicateFiles(files,numDesired)
n = numel(files);
ind = randi(n,numDesired,1);
files = files(ind);
end

```

B. MATLAB Code for Accuracy Formula.

```
function accuracy = accuracy_score(Ypred, Ynew)
%Accuracy Score
%
% Author: M.Sc. David Ferreira - Federal University of Amazonas
% Contact: ferreirad08@gmail.com
% Date: October 2020

[name_labels,~,Ypred] = unique(Ypred); % Numeric predicted labels
[~,Ynew] = ismember(Ynew,name_labels); % Numeric new labels
accuracy = mean(Ypred == Ynew); % calculated accuracy
end
```

C. MATLAB Code For K- Nearest Neighbor Classification Visualization.

```
% The training and testing code need to be run first to get the data for
visualisation.
data=Xtrain';
% To get the data for average crack size in horizontal and vertical.
data=[data(:,7) data(:,8)];
modelformed= fitcknn(data,Ytrain,'NumNeighbors',3);
X=Xtest';
% To plot graph vertical against horizontal crack data.
figure(1);
deltax=X(:,7);
deltay=X(:,8);
e=min(deltax):1:max(deltax);
f=min(deltay):1:max(deltay);
[x1,x2]=meshgrid(e,f);
x=[x1(:) x2(:)];
ms=predict(modelformed,x);
gscatter(x1(:),x2(:),ms,'cym','.',10);
hold on;
gscatter(deltax,deltay,Ytest,'rgb','.',10);
title('KNN Classification Visualization');
xlabel('Crack mean in horizontal position')
ylabel('Crack mean in straight position')
legend('Area Crack','Area Un-crack','Crack data','Un-crack data')
```


D. MATLAB Code for Each Dataset Crack Detection Testing.

```
% To choose dataset for detect crack
[file,path]=uigetfile({'*.jpg'; '*.png'; '*.jpeg'}, 'Choose an image'); % choose
folder
s=[path,file]; % read folder path
I=imread(s); % read image choose
% Images Processing
Istrech = imadjust(I,stretchlim(I)); % stretch the image
Igray_s = rgb2gray(Istrech); % convert to gray scale
% To display gray scale
figure;
subplot(2,2,1)
imshow(Igray_s)
title('gray scale')
level = 0.4; % set geyscale intensity level for feature extraction
Ithres = imbinarize(Igray_s,level);
subplot(2,2,2)
imshow(Igray_s)
title('binarize image')
% Features extraction.
Ithres = ~Ithres;
BW = bwmorph(Ithres, 'clean',10);
BW11=edge(BW, 'canny');
subplot(2,2,3)
imshow(BW11)
title('Canny Feature Extraction')
```

```
[y, x] = find(BW);
x1 = min(x); x2 = max(x); z=mean(x);
y1 = min(y); y2 = max(y); z2=mean(y);
delta_x = x2 - x1;
delta_y = y2 - y1;
Xnew = [delta_x,delta_y,x1,y1,x2,y2,z,z2];
% from training data train in KNN compare the feature extraction with train
% databased
% Then predict
Ynew = mdl.predict(Xnew);
% To display original image
subplot(2,2,4);
imshow(I)
% set image title as predict data
a=cellstr(Ynew); % Ynew is predict data
a=a{1};
a = convertCharsToStrings(a);
title('The image is : ' + a)
```