Semi-Automated Crack Detection Using Computer Vision

N Sujeeka[1], S Lowhikan[2] and H M Y C Mallikarachchi[3]

Abstract

The characteristics of surface cracks reflect the health state and functional degradation level of structures. Inspection and measurement of cracks on concrete surfaces are therefore of crucial importance and considered to be a fundamental component of safety and health monitoring of concrete structures. The conventional manual tracing of crack detection is time-consuming and depended largely on the inspector's experience and knowledge. This paper presents a novel semi-automated approach for crack detection using computer vision and image processing techniques. The proposed method requires a pre-defined length to determine the calibration factor which gives the relation between the pixels of the image and the actual scale. High-resolution images of surface cracks are captured and subsequently processed according to the proposed method to obtain accurate crack measurements. The acquired digital images are subjected to a few image pre-processing steps for noise reduction and accurate determination of crack boundaries. The study also includes a comparison of the efficiency of edge detection and segmentation techniques in crack detection. Furthermore, this approach can be easily accommodated on mobile phone platforms that simplify and accelerate the process of crack detection. Experimental results demonstrate that the established method is capable of precisely perceiving the surface crack measurements.

1. Introduction

The presence of cracks is a useful indicator of the state of health, integrity, and durability of a structure. Type of cracks can vary from micro-cracks to large structural cracks which may or may not affect the functionality of a structure. However, all types of cracks enable harmful and corrosive chemicals to penetrate the structure, thus damaging both its integrity and aesthetics [1]. Early identification and monitoring of cracks are therefore essential to implement an effective method of remediation and to avoid subsequent crack failures. Figure 1 shows a few examples of structures made of bricks and concrete.

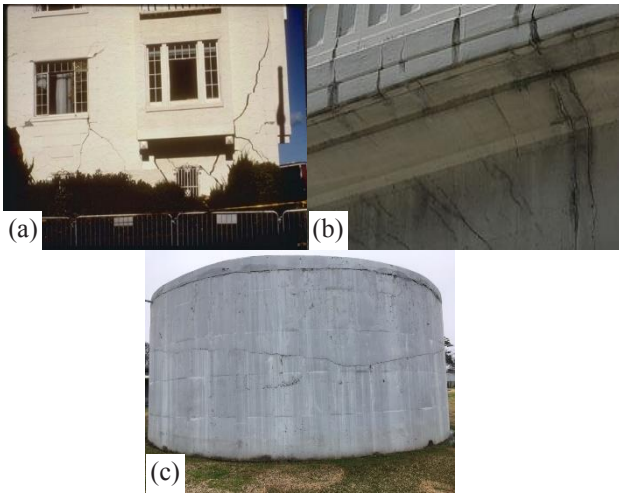


Figure 1: Visual cracks (a) building (b) bridge (c) water tank (source: infrastructurepc.com)

Crack width is an important parameter in the design of concrete structures and hence crack width measurement is crucial in structural health monitoring.

N Sujeekam[1], S Lowhikan[2], and H M Y C Mallikarachchi[3] are with the department of Civil Engineering, University of Moratuwa, Sri Lanka. (e-mail: nsujeeka@gmail.com; lowhikan1@gmail.com; yasithcm@uom.lk)

There are various techniques used to extract the geometric information of the surface cracks all around the world including advanced techniques such as laser doppler vibrometer, confocal scanning laser microscopy, and several types of image-based crack measurement [2]-[4].

However, the local context largely relies on conventional manual and visual inspections which is time-consuming and highly subjective which questions the reliability of measurements. The process requires manual tracing of cracks and the use of a crack width gauge with eye estimation. Figure 2 illustrates the crack measurement process using a crack width gauge/ruler by comparing the calibration marks. Note that even a simple structure can have a large number of cracks and the location of measurement as well as the width of the crack is judged by the inspector.

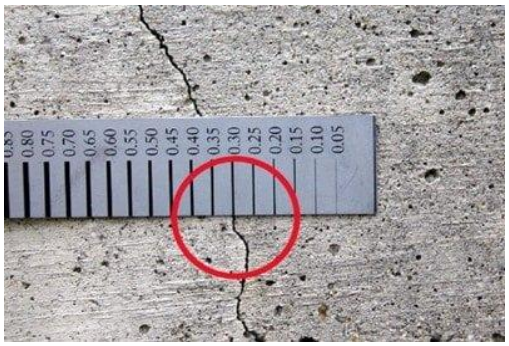


Figure 2: Measurement of crack width using crack width gauge (source: theconstructor.org)

A significant amount of research has been performed especially focused on developing an automatic and semi-automatic optical image-based crack measurement system by various groups [4]-[13]. However, the main concern has been on developing advanced image segmentation and edge detection-based boundary detection algorithms combined with essential pre- and post-image processing steps. Such studies are complex in nature and require the help of a personal computer which is not portable.

This study aims to contribute to the local industry by developing an application for an instant in situ surface crack measurement that can be operated on commonly accessible mobile phones. Initially, a semi-automatic algorithm was developed using the MATLAB software package, which allows the user to manually select the boundary points where crack measurement is required. The algorithm was then implemented on a smartphone platform to take in situ measurements. The proposed application is verified by assessing images taken from 8 different crack locations and width measurements were taken at a few selected locations along each crack as shown in Figure 3 and Figure 4. The research also provides a comparison of the performance of edge detection and segmentation techniques.

Following the introduction, Section 2 describes the steps followed in developing the android application and Section 3 explains the hierarchical structure of the application. The fundamental aspects of the image processing steps adopted are briefed in Section 4. Section 5 discusses the results and Section 6 concludes the paper.

2. Development of Mobile Application

An algorithm built on the MATLAB platform can be constructed as an android application in two ways [14]. The first approach is to create an equivalent model to the algorithm in Simulink - an interactive graphical programming environment based on MATLAB. Simulink's custom block library consists of several built-in blocks listed under various toolboxes similar to MATLAB. It also provides the facility for building a new block with user-defined functions. The algorithm built on the MATLAB framework can be used to construct the Simulink model using its drag and drop system, eventually, the model can be launched as an application in the intended mobile phone using the Simulink Support Package for Android Devices.

The second approach is to translate the MATLAB syntax to C or C++ program code and use the source code to create an application using any of the android application designing software. This approach is adopted over the first method as it provides better Graphical User Interface (GUI) design for the application. MATLAB Coder – one of the built-in utilities of MATLAB helps one to convert the algorithm into the C or C++ language. Android Studio, one of the widely used software for android application development is selected over the other software as it makes mobile application development easier because of its open-source platform and it provides a stable Integrated Development Environment (IDE). As the program is based on the JAVA platform, the resulting C codes have been translated to JAVA using an online converter, and the framework is then designed and installed via USB debugging on the mobile phone.

3. Hierarchical Structure

The overall design concept of the proposed system is to capture the crack image, perform fundamental image processing, execute the edge detection/segmentation algorithm selected by the user, and calculate the crack width. The hierarchical structure of the android application is explained in Figure 5.

Figure 3: Photographs of cracks considered

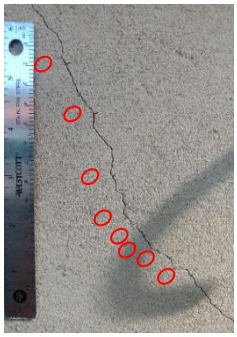


Figure 4: Locations of crack width measurement along the crack

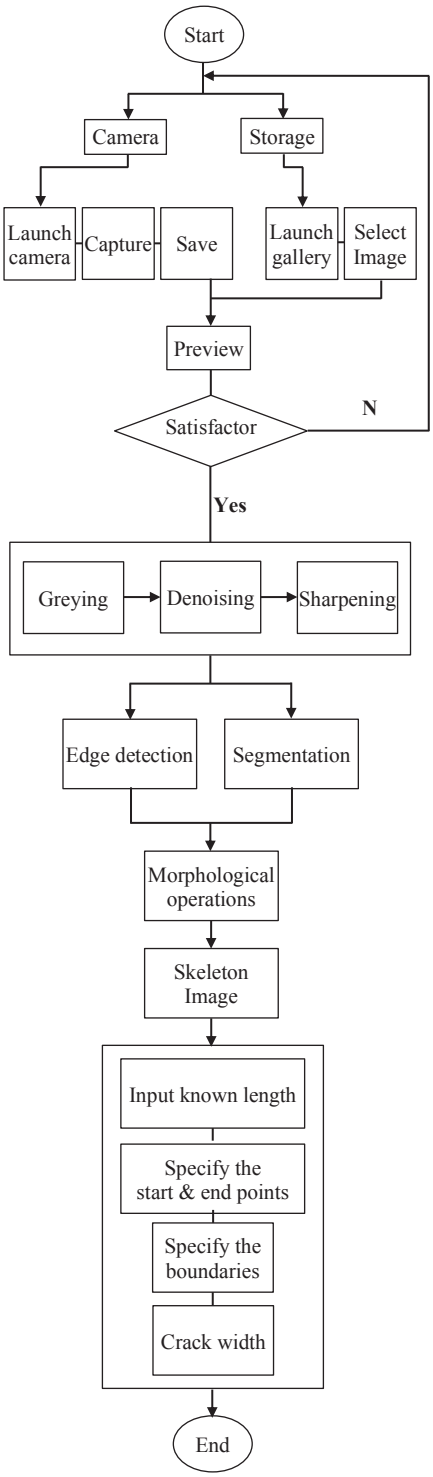


Figure 5: Hierarchical structure of the

3.1 Input

The system has provisions for taking a photograph using the built-in camera or loading an image from the internal storage. It enables the launch of the default camera or other similar applications with all the basic activities including focus, zoom, filters, display, etc. When the satisfactory image is captured it will be stored and previewed in the application's image viewer space.

It should be noted that the mobile phone should be held parallel to the surface as much as possible when taking the images. It is recommended to keep the clear distance between the surface and the camera within a range of 10 – 15 cm, to get a good focus and optimum use of resolution. The system requires the image to have a line of predefined length or a scale attached near the crack for calibration.

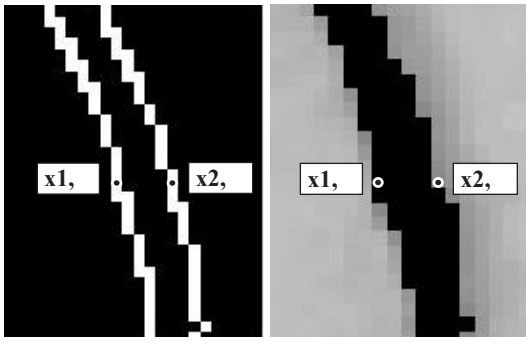
Choosing it from storage is another way to load an image into the application. The gallery or similar applications are launched in the second option, so the required image can be selected. Note that the selected image should have a scale or line of known length as mentioned above. The processing steps applied for the images are explained in detail in Section 4.

3.2 Crack Width Measurement

The user is required to select two points on the image and enter the distance between the two points in millimeters to determine the calibration factor. The application then determines the line length in pixels, eventually, the calibration factor will be estimated by dividing the actual length by the pixel length.

The user can then select the points for which the width calculation is required (see Figure 6), and the value will be displayed on the screen. It should be noted that the pixel centers are aligned with the crack boundary in the edge image while the region between the boundary is extracted in segmentation. Therefore, when selecting boundary pixels in segmentation, the one-pixel offset should be allowed for accurate width measurement.

This process requires the user to identify a crack region and take images while specifying the known length for each picture and hence completely eliminate subjectivity due to visual inspection. Further, this facilitates structural engineer to get an idea of the surface condition as well as taking new measurements along the crack since the saved images can be post-processed.



(a) (b)

Figure 6: Pixel coordinates of boundary points in (a) edge image (b) segmentation

4. Digital Image Processing

Digital Image Processing in this context is mainly focused on image enhancement and image analysis. The image processing section consists of functions such as the conversion of grayscale, denoising filters, and sharpening filters. After these fundamental activities, the user has the freedom to choose the option of edge detection or segmentation with varying thresholds to get the skeleton image of the crack.

4.1 Greying

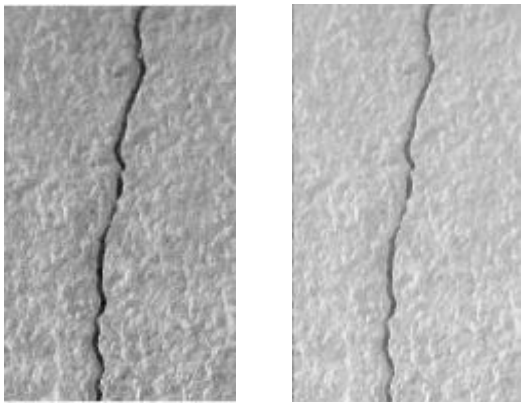
The color image loaded to the application is converted to a grayscale image. The true-color image is stored in m×n×3 data array that specifies the red, green, and blue color components for each pixel. The color of each pixel is determined by the combination of red, green, and blue intensities stored at the pixel location of each color plane [15]. The MATLAB function *rgb2gray* converts RGB values to grayscale values by forming a weighted sum of the R, G, and B components as given in equation (1) [14].

$$0.2989 \times R + 0.5870 \times G + 0.1140 \times B \quad \ldots\ldots\ldots\ldots (1)$$

4.2 Denoising

Due to environmental, transmission, and other factors, images are inevitably polluted by noise during acquisition, compression, and transmission, leading to distortion and loss of image content [16]. The image analysis of the raw images is adversely affected due to the presence of noise, hence the denoising of the images is considered to be a mandatory pre-processing step.

A two-dimensional median filter was used as it requires to simultaneously remove the noise and preserve the edges. Median filtering is performed by sliding a window over the image. The filtered image is obtained by placing the median of the values at the output image in the input window, at the center of that window position. The grayscale image and denoised image are compared in Figure 7 below. It can be noticed that the surface/background of the denoised image is smooth compared to the raw grayscale image, while the boundary properties have not been altered.

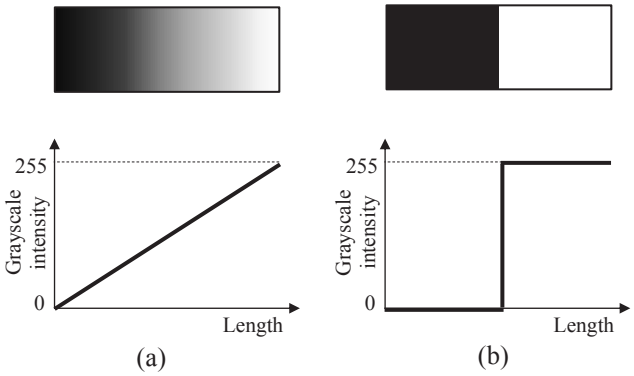


(a) (b)

Figure 7: (a) Grayscale image (b) Denoised image

4.3 Sharpening

Sharpness is simply a contrast between different colors. The fast switch from black to white looks pretty sharp while the gradual transition from black to grey to white is blurred as seen in Figure 8 below. Image sharpening is any enhancement technique that highlights the edges and fine detail by increasing the contrast between the bright and dark regions. The image is sharpened by the application of unsharp masking, which subtracts the blurred (unsharp) version of the image from itself [14].



(a) (b)

Figure 8: (a) Blurry edge (b) Sharp edge

In order to analyze the effect of sharpening, grayscale intensity values of pixels across the crack at a random location were observed. The contrast enhancement at the crack boundary due to the application of unsharp masking is given in Figure 9. The actual crack region lies between the pixel index range of 309 – 316. It can be noted that the difference in intensity values at locations 309 and 316 is increased which would make the edge detection or the segmentation process easier.

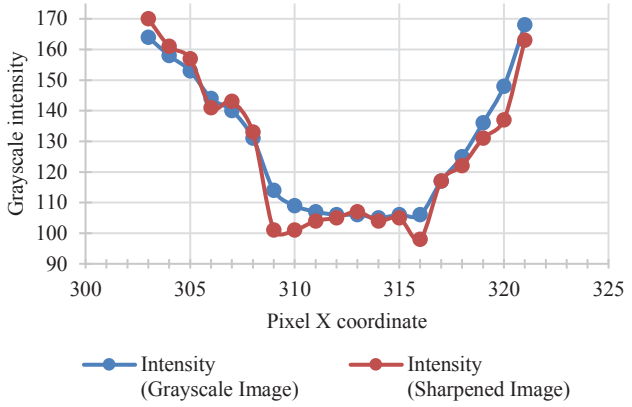


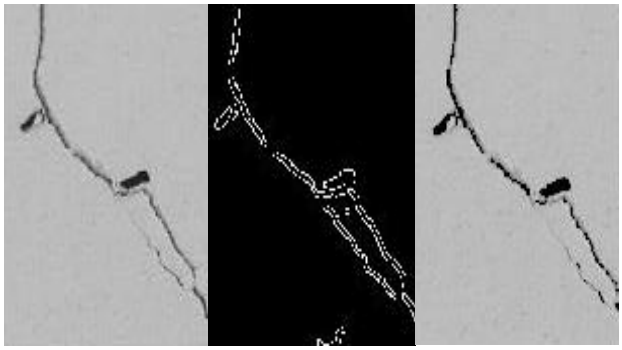
Figure 9: Gray scale intensity variation

4.4 Edge Detection and Segmentation

The application offers two options for image analysis: edge detection and segmentation. Generally, edge detectors are mathematical methods that can recognize points in an image at which the grey-level intensity expresses discontinuities [17]. Among common edge detection algorithms such as Sobel, Canny, Prewitt, and Roberts, the Canny edge detection algorithm was used in this application as it is capable of detecting thin edges as well as removing the false edge from the image [18].

Image segmentation is another technique widely used in digital image processing to partition an image into multiple parts or regions, often based on the characteristics of the pixels in the image. Image segmentation could involve separating the foreground from the background, or clustering regions of pixels based on similarities in color or shape. Many automated image thresholding and segmentation techniques are available, but segmentation based on manual thresholding is adopted as a simpler and faster solution. The user may determine the optimum threshold value through a trial and error method for the precise segmentation of crack boundaries.

In the case of cracks on a rough surface, tiny bumps on the surface can produce small patches of segmentations and edges in the processed images. The images are then subjected to certain morphological operations to reduce the distribution of these regions. It should be noted, however, that patches cannot be completely removed in this technique. The resulted image after the edge detection and segmentation technique is shown in Figure 10.



(a) (b) (c)

Figure 10: Identification of cracks (a) grayscale image (b) edge image (c) segmentation

5. Results and Discussion

The measured crack width using the crack width ruler and the proposed technique (1) edge detection and (2) segmentation are compared in Table 1. Considering the difference between the measured and estimated crack width values, the percentage error is determined with respect to the crack width gauge measurements. The edge detection technique has a 4.39% of mean error and standard deviation of 3.28%, whereas the segmentation technique has 9.77% and 7.33 % of the mean and standard deviation of error respectively.

In order to accelerate the process, the segmentation technique based on a global threshold value was used rather than incorporating different regional threshold values. Thus, in images where the grayscale intensity varies substantially along the length of the crack, some regions may be lost or incorrectly extracted during the segmentation process. In these instances, it is better to use the edge detection method.

Based on the results of the analysis, it can be stated that the edge detection technique is more suitable for images of cracks on smooth surfaces where there is less probability of developing unwanted edge patches. Regardless, of the magnitude of the grayscale intensity, the edge detection technique will determine the boundaries precisely. The segmentation technique is best suited for crack images with a constant or small range of intensity variation along the crack and is not affected by the unevenness of the surface.

Table 1: Comparison of crack widths measured with crack gauge and proposed techniques

Location	Crack Gauge (mm)	Edge Detection		Segmentation	
		Width (mm)	Error (%)	Width (mm)	Error (%)
1	0.6096	0.6128	0.52	0.5728	-6.04
	0.5080	0.5126	0.91	0.4296	-15.43
	0.5080	0.5126	0.91	0.5370	5.71
	0.5080	0.5063	-0.33	0.5012	-1.34
	0.5080	0.5012	-1.34	0.4654	-8.39
	0.5080	0.5063	-0.33	0.6086	19.80
	0.4572	0.4668	2.10	0.5370	17.45
	0.5080	0.5025	-1.08	0.5382	5.94
	0.3556	0.3526	-0.84	0.3242	-8.83
2	0.2032	0.2083	2.51	0.1679	-17.37
	0.2032	0.2083	2.51	0.1920	-5.51
	0.1524	0.1397	-8.33	0.1473	-3.35
	0.2540	0.2329	-8.31	0.2375	-6.50
	0.1016	0.0931	-8.37	0.0931	-8.37
	0.1524	0.1473	-3.35	0.1473	-3.35
	0.2032	0.1863	-8.32	0.2329	14.62
	0.2032	0.1863	-8.32	0.2329	14.62
	0.1524	0.1473	-3.35	0.1863	22.24
	0.2032	0.1976	-2.76	0.1976	-2.76
3	0.5588	0.5100	-8.73	0.5802	3.83
	0.4572	0.4868	6.47	0.5170	13.08
	0.5080	0.5322	4.76	0.5702	12.24
	0.5080	0.4640	-8.66	0.4334	-14.69
	0.5588	0.5170	-7.48	0.4956	-11.31
	0.5588	0.5702	2.04	0.5702	2.04
	0.4064	0.4433	9.08	0.4433	9.08
	0.5080	0.5281	3.96	0.4577	-9.90
	0.5080	0.5376	5.83	0.5402	6.34
4	0.3556	0.3492	-1.80	0.3492	-1.80
	0.2540	0.2494	-1.81	0.2494	-1.81
	0.2540	0.2494	-1.81	0.2494	-1.81
	0.3048	0.2909	-4.56	0.2993	-1.80
	0.2540	0.2494	-1.81	0.2822	11.10
	0.2032	0.2057	1.23	0.1996	-1.77
	0.1524	0.1578	3.54	0.1578	3.54
	0.2032	0.2057	1.23	0.2494	22.74
	0.2032	0.2117	4.18	0.2231	9.79
	0.1524	0.1411	-7.41	0.1497	-1.77
5	0.1016	0.0982	-3.35	0.1389	36.71
	0.1016	0.1042	2.56	0.1042	2.56
	0.1016	0.1042	2.56	0.0776	-23.62
	0.1016	0.0982	-3.35	0.1042	2.56
	0.1016	0.1042	2.56	0.1042	2.56
	0.1016	0.1098	8.07	0.0776	-23.62
	0.1524	0.1389	-8.86	0.1736	13.91
	0.1016	0.1098	8.07	0.1098	8.07

6	0.1524	0.1511	-0.85	0.1676	9.97
	0.1524	0.1676	9.97	0.1257	-17.52
	0.2032	0.1874	-7.78	0.2514	23.72
	0.1542	0.1511	-2.01	0.1874	21.53
	0.0508	0.0419	-17.52	0.0419	-17.52
	0.1016	0.0937	-7.78	0.0937	-7.78
	0.1542	0.1511	-2.01	0.1778	15.30
	0.1016	0.0937	-7.78	0.1185	16.63
	0.2032	0.2095	3.10	0.2095	3.10
	0.1016	0.0937	-7.78	0.0838	-17.52
7	0.5588	0.5705	2.09	0.5173	-7.43
	0.3556	0.3746	5.34	0.3971	11.67
	0.2032	0.1902	-6.40	0.1902	-6.40
	0.4572	0.4871	6.54	0.4201	-8.11
	0.4064	0.3822	-5.95	0.3803	-6.42
	0.3556	0.3506	-1.41	0.3803	6.95
	0.3048	0.3066	0.59	0.3066	0.59
	0.4064	0.3803	-6.42	0.4564	12.30
8	0.6096	0.6284	3.08	0.6043	-0.87
	0.7620	0.7426	-2.55	0.6189	-18.78
	0.5080	0.5033	-0.93	0.5019	-1.20
	0.5588	0.5527	-1.09	0.5791	3.63
	0.4572	0.4633	1.33	0.4247	-7.11
	0.5588	0.5405	-3.27	0.5405	-3.27
	0.5588	0.5405	-3.27	0.5033	-9.93
	0.2540	0.2316	-8.82	0.2702	6.38
Mean	-	-	**4.39**	-	**9.77**
Standard Deviation	-	-	**3.28**	-	**7.33**

6. Conclusion

This study has developed a computer vision-based crack width measurement algorithm and implemented it on a mobile phone platform to facilitate in-situ measurements. The proposed technique provides two frequently used techniques for image analysis: edge detection and segmentation, and requires input images to consist of a predefined length to determine the calibration factor. Results have shown that the edge detection technique of the proposed system can be used to measure width with accuracy greater than 91% whereas the segmentation technique is more appropriate for images with uniform grayscale intensity along the crack.

Since this is a semi-automated system where the user manually chooses the locations, it is convenient to get measurements at the targeted locations only. It also enables the user to take new measurements in the future as the images are saved in the internal storage once they have been captured. It is intended to further extend the study by incorporating features for the estimation of crack length in the future.

References

[1] R. S. Adhikari, O. Moselhi, and A. Bagchi, "Image-based retrieval of concrete crack properties for bridge inspection," *Autom. Constr.*, vol. 39, pp. 180–194, 2014, doi: 10.1016/j.autcon.2013.06.011.

[2] R. Longo, S. Vanlanduit, J. Vanherzeele, and P. Guillaume, "A method for crack sizing using Laser Doppler Vibrometer measurements of Surface Acoustic Waves," *Ultrasonics*, vol. 50, no. 1, pp. 76–80, 2010, doi: 10.1016/j.ultras.2009.08.001.

[3] B. Menéndez, C. David, and A. M. Nistal, "Confocal scanning laser microscopy applied to the study of pore and crack networks in rocks," *Comput. Geosci.*, vol. 27, no. 9, pp. 1101–1109, 2001, doi: 10.1016/S0098-3004(00)00151-5.

[4] A. Mohan and S. Poobal, "Crack detection using image processing: A critical review and analysis," *Alexandria Eng. J.*, vol. 57, no. 2, pp. 787–798, 2018, doi: 10.1016/j.aej.2017.01.020.

[5] Y. Fujita and Y. Hamamoto, "A robust automatic crack detection method from noisy concrete surfaces," *Mach. Vis. Appl.*, vol. 22, no. 2, pp. 245–254, 2011, doi: 10.1007/s00138-009-0244-5.

[6] H. N. Nguyen, T. Y. Kam, and P. Y. Cheng, "An Automatic Approach for Accurate Edge Detection of Concrete Crack Utilizing 2D Geometric Features of Crack," *J. Signal Process. Syst.*, vol. 77, no. 3, pp. 221–240, 2014, doi: 10.1007/s11265-013-0813-8.

[7] B. Y. Lee, Y. Y. Kim, S. T. Yi, and J. K. Kim, "Automated image processing technique for detecting and analyzing concrete surface cracks," *Struct. Infrastruct. Eng.*, vol. 9, no. 6, pp. 567–577, 2013, doi: 10.1080/15732479.2011.593891.

[8] P. Wang and H. Huang, "Comparison analysis on present image-based crack detection methods in concrete structures," *Proc. - 2010 3rd Int. Congr. Image Signal Process. CISP 2010*, vol. 5, pp. 2530–2533, 2010, doi: 10.1109/CISP.2010.5647496.

[9] N. D. Hoang, "Detection of Surface Crack in Building Structures Using Image Processing Technique with an Improved Otsu Method for Image Thresholding," *Adv. Civ. Eng.*, vol. 2018, 2018, doi: 10.1155/2018/3924120.

[10] A. M. A. Talab, Z. Huang, F. Xi, and L. Haiming, "Detection crack in image using Otsu method and multiple filtering in image processing techniques," *Optik (Stuttg).*, vol. 127, no. 3, pp. 1030–1033, 2016, doi: 10.1016/j.ijleo.2015.09.147.

[11] K. Khalili and M. Vahidnia, "Improving the Accuracy of Crack Length Measurement Using Machine Vision," *Procedia Technol.*, vol. 19, pp. 48–55, 2015, doi: 10.1016/j.protcy.2015.02.008.

[12] M. Salman, S. Mathavan, K. Kamal, and M. Rahman, "CrackDetectionRoad_GaborFilter_2013," *Proc. 16th Int. IEEE Annu. Conf. Intell. Transp. Syst. (ITSC 2013)*, pp. 2039–2044, 2013, doi: 10.1109/ITSC.2013.6728529.

[13] Y. Sen Yang, C. M. Yang, and C. W. Huang, "Thin crack observation in a reinforced concrete bridge pier test using image processing and analysis," *Adv. Eng. Softw.*, vol. 83, pp. 99–108, 2015, doi: 10.1016/j.advengsoft.2015.02.005.

[14] "MATLAB Documentation R2019b," 2019.

[15] B. N. Miller and D. L. Ranum, *Python Programming in Context*. Jones & Bartlett Publishers, 2008.

[16] L. Fan, F. Zhang, H. Fan, and C. Zhang, "Brief review of image denoising techniques," *Vis. Comput. Ind. Biomed. Art*, vol. 2, no. 1, 2019, doi: 10.1186/s42492-019-0016-7.

[17] D. Sundararajan, *Digital Image Processing-A Signal Processing and Algorithmic Approach*. Springer Singapore, 2017.

[18] N. A. Channar, "A Comparative Study of Edge Detection Techniques in Digital Images," 2016.