applied sciences

MDPI

Article

Crack Detection of Concrete Images Using Dilatation and Crack Detection Algorithms

Byeong-Cheol Kim [1] and Byung-Jik Son [2,*]

[1] Department of Structural Engineering, KICT, Goyang 10223, Republic of Korea; bckim@kict.re.kr
[2] Department of Disaster Safety & Fire, Konyang University, Nonsan 32992, Republic of Korea
* Correspondence: strustar@konyang.ac.kr

Abstract: Crack detection in structures is an important and time-consuming element of monitoring the health of structures and ensuring structural safety. The traditional visual inspection of structures can be unsafe and may produce inconsistent results. Thus, there is a need for a method to easily and accurately identify and analyze cracks. In this study, algorithms for automatically detecting the size and location of cracks in concrete images were developed. Cracks were automatically detected in a total of 10 steps. In steps 5 and 9, two user algorithms were added to increase crack detection accuracy, where 1000 crack images and 1000 non-crack images were used, respectively. In the crack image, 100% of the cracks were detected, but 95.3% of the results were very good, even if the results that were not bad in terms of quality were excluded. In addition, the accuracy of detecting non-crack images was also very good (96.9%). Thus, it is expected that the crack detection algorithm presented in this study will be able to detect the location and size of cracks in concrete. Moreover, these algorithms will help in observing the soundness of structures and ensuring their safety.

Keywords: image processing; crack detection; concrete cracks; user algorithms; dilatation

check for updates

Citation: Kim, B.-C.; Son, B.-J. Crack Detection of Concrete Images Using Dilatation and Crack Detection Algorithms. *Appl. Sci.* **2023**, *13*, 9238. https://doi.org/10.3390/app13169238

Academic Editor: Laurent Daudeville

Received: 18 June 2023
Revised: 3 August 2023
Accepted: 9 August 2023
Published: 14 August 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Cracks in civil engineering structures are a prevalent occurrence and not only compromise the structural integrity of these structures but also cause significant human and financial losses [1,2]. Due to shifting loading conditions (corrosion, etc.), civil structures, especially aging concrete structures, are susceptible to crack formation. Typically, cracks in asphalt and concrete roadways show as lines with various directions and intensities. Generally, these lines are darker and linked, and a simple fracture detection can be performed by utilizing thresholds that have been correctly defined [2,3].

Image processing techniques, which have recently been attracting research attention, are used for the maintenance of structures. The most important and basic aspect of the maintenance of structures and facilities is to measure and analyze cracks. The maintenance method so far consumes a lot of time and effort because the inspector evaluates the condition with the naked eye, and the objectivity and reliability of the results are poor [4–6].

Image processing techniques can detect deterioration phenomena, such as cracks and defects, and measure the strain and displacement of a specimen without attaching a gauge. These image processing techniques can reduce the enormous costs and manpower required for traditional visual inspections, and greatly improve restrictions on the timing and location of inspections [7–10].

Image processing techniques ('white box' techniques) and less transparent artificial neural network approaches ('black box' techniques) are becoming alternative approaches, reducing human effort [11]. White box techniques have the advantages of being inexpensive to compute, traceable, transparent, and they do not require large datasets for training. Despite the success of black box techniques in concrete crack classification and detection, there is still a demand for white box techniques. Hybrid or 'gray box' approaches utilize both techniques to achieve better performance in pixel-level segmentation tasks [12].

A crack image can be segmented into binary image using various white box methods. Algorithms are generally divided into edge-based (Canny edge detector, Wavelet transformation, etc.) and threshold-based (Otsu thresholding, etc.) [13]. Although these methods are effective, many false positives in the form of noise reduce the accuracy of crack detection results.

In this study, an algorithm for dilating the image and an algorithm for determining crack size were developed to improve the accuracy of crack detection in concrete. A total of 40,000 images of METU campus buildings were used [14,15], consisting of 20,000 crack images and 20,000 non-crack images. In this study, 1000 images from each category, a total of 2000 images, were selected and analyzed. The flow chart used to automatically detect the location and size of cracks was developed using a total of 10 steps. In steps 5 and 9, two user algorithms were added to increase crack detection accuracy. As a result of the analysis, an accuracy of over 95% was found in both the crack images and the non-crack images, which is considered very excellent.

2. Crack Detection Procedure

In this study, the location and size of cracks were detected in concrete images, and the analysis was essentially performed using the Image Processing Toolbox in Matlab [16], and two user algorithms were added to improve the accuracy of crack detection.

For concrete images, datasets collected from METU campus buildings were used [14,15]. The collected datasets consist of 20,000 crack images and 20,000 non-crack images. However, in this study, 1000 images from each category, a total of 2000 images, were selected and analyzed. The size of each color image is 227 × 227 × 3, and the resolution is 51,529 pixels, a very low resolution. The 227 × 227 size represents the width and height of the image, and the unit is pixels. The number "3" at the end indicates a color image, which consists of three channels: Red, Green, and Blue. Twenty crack images and twenty non-crack images are randomly shown in Figures 1 and 2.

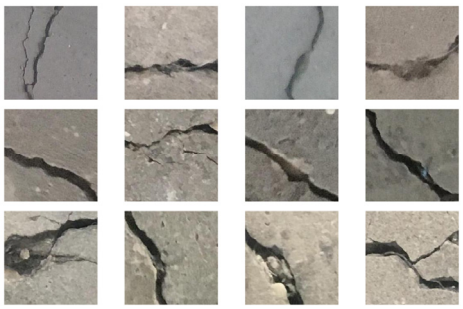


Figure 1. Sample crack images.



Figure 2. Sample non-crack images.

The crack detection procedure consists of 10 steps, as shown in Figure 3 and detailed in Tables 1 and 2. Two user algorithms are needed in steps 5 and 9, and these are explained in detail in Section 3. In Table 1, two crack images ('00330.jpg' and '00797.jpg') are shown, and in Table 2, two non-crack images ('00283.jpg' and '00318.jpg') are shown as examples.

Although the non-crack images in Table 2 are not actual cracks, they demonstrate the incorrect processing procedure that leads to false crack identification.

A concise summary of the crack detection procedure for each step is shown below.

Table 1. The analysis procedures for crack images ('00330.jpg' and '00797.jpg').

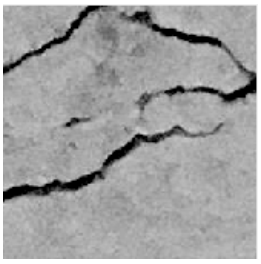
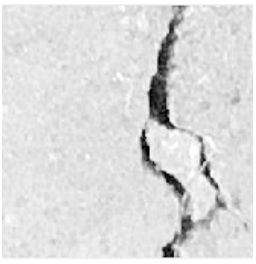
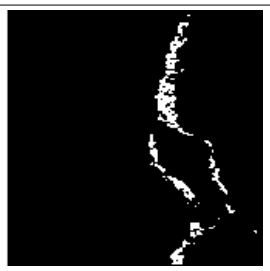
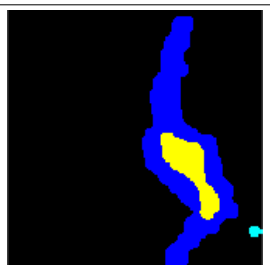
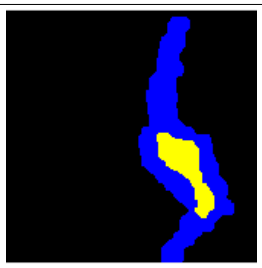
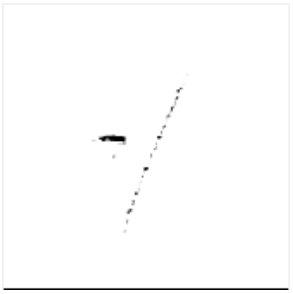
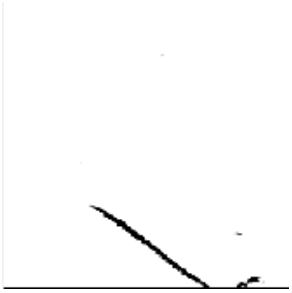
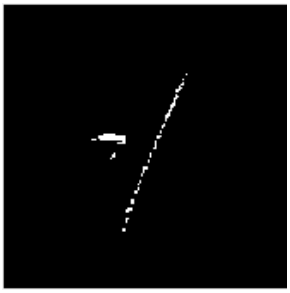
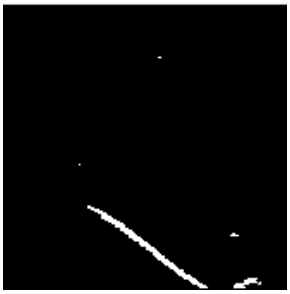
Step 1: Read the image (Original image)		Step 2: Enhance the image	
			
Step 3: Filtering		**Step 4: Binarize and complement the image**	
			
Step 5: Dilate the image (User Algorithm I)		**Step 6: Open the image**	
			
Step 7: Erode the image		**Step 8: Labeling connected components**	
			
Step 9: Detect the cracks (User Algorithm II)		**Step 10: Overlay the image**	
			

Table 2. The analysis procedures for non-crack images ('00283.jpg' and '00318.jpg').

Step 1: Read the image (Original image)		Step 2: Enhance the image	
			
Step 3: Filtering		**Step 4: Binarize and complement the image**	
			
Step 5: Dilate the image (User Algorithm I)		**Step 6: Open the image**	
			
Step 7: Erode the image		**Step 8: Labeling connected components**	
			
Step 9: Detect the cracks (User Algorithm II)		**Step 10: Overlay the image**	
			

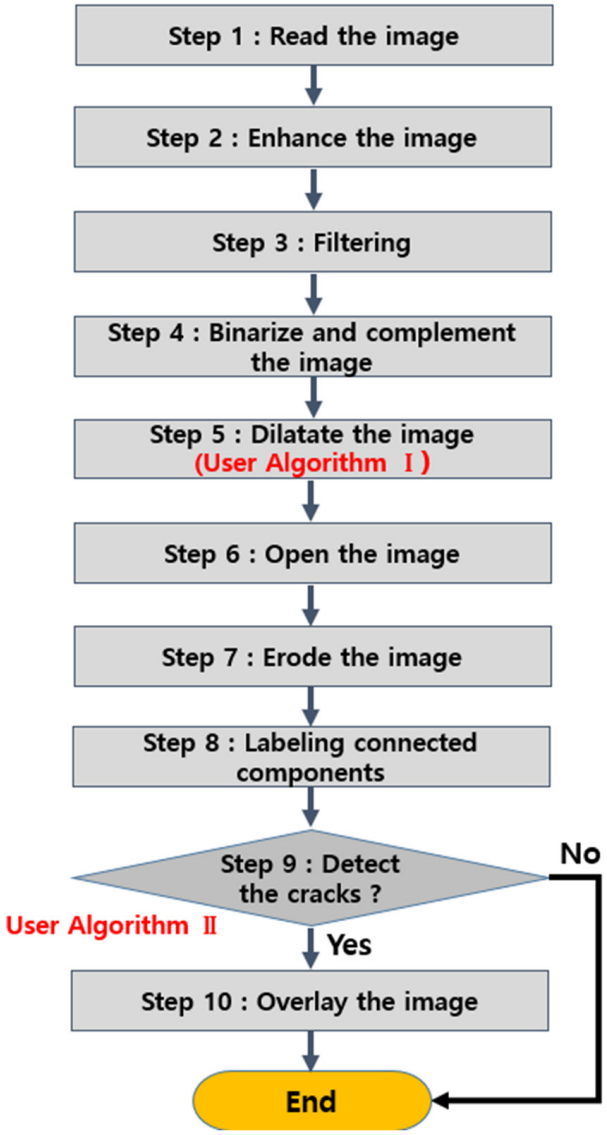


Figure 3. Flow chart for crack detection.

2.1. Step 1: Read the Image

In step 1, the original concrete image is read using the Matlab program [16].

2.2. Step 2: Enhance the Image

In step 2, the image is enhanced. Before enhancing the image, the color image is converted to a grayscale image. Various methods can be used to enhance images. In this study, due to concrete image's blurry nature, image brightening was applied to improve its quality [17–19]. A technique for improving low-light images, as referenced in [17–19], was implemented.

2.3. Step 3: Filtering

In step 3, the image is filtered to detect cracks. Various filters can detect edges (cracks), including Sobel, Prewitt, Roberts, LoG (Laplacian of Gaussian), and Canny operators. In this study, effective results were achieved using a filter denoted by Equation (1).

$$f = \begin{bmatrix} -2 & -2 & -2 \\ -1 & 2 & -1 \\ 3 & 3 & 3 \end{bmatrix} \quad (1)$$

2.4. Step 4: Binarize and Complement the Image

In step 4, the image is binarized and reversed. A binary image consists of two colors, black and white. In the filtering result of step 3, since the crack is black, the threshold value for binarization is set very large (0.99) to maximize the detection of cracks. In the next step, the cracks must be white for calculations, so the image is reversed. As a result, cracks (foreground) become white and the background becomes black.

2.5. Step 5: Dilate the Image (User Algorithm I)

In step 5, the image is dilated to improve crack detection. User Algorithm I is used and described in detail in Section 3.

2.6. Step 6: Open the Image

In step 6, a morphological opening is performed. The morphological opening operation consists of an erosion followed by a dilation, using the same structuring element for both operations. Its role is to dilate cracks and connect them.

2.7. Step 7: Erode the Image

In step 7, the image is eroded. This step balances the image by eroding it to the same extent as the dilation in step 5.

2.8. Step 8: Labeling Connected Components

Step 8 involves classifying objects (cracks) and background through the labeling of connected components. The connected components are displayed in the same color.

2.9. Step 9: Detect the Cracks (User Algorithm II)

Step 9 is the step for determining the presence or absence of cracks. In the process of extracting only cracks, it involves determining whether the object labeled in the previous step is a crack or not. User Algorithm II is required for this step and is explained in detail in Section 3.

2.10. Step 10: Overlay the Image

In step 10, the crack boundaries detected in the previous step are finally overlaid on the original image. The first crack is marked in red, the second crack in yellow, the third crack in cyan, the fourth in magenta, the fifth in blue, and any additional cracks in white. The holes inside the cracks are marked in green.

3. Crack Detection User Algorithms

Two algorithms (Algorithm I in step 5 and Algorithm II in step 9) were developed to detect the location and size of cracks.

3.1. User Algorithm I (Step 5)

User Algorithm I is an algorithm that dilates an image to better identify cracks. If the crack is dilated too much, the crack may become larger and objects other than the crack may be mistakenly recognized as a crack. If the crack dilates too little, it may not be recognized. Therefore, an algorithm that appropriately dilates the image is needed to improve crack identification.

Table 3 utilizes the binary image (im1) in step 4. The ratio (z1) of objects (cracks) in the binary image (im1) is calculated using the nnz function and rounded to n1. In line 5, a 5×5 sub-image (sub_im) is constructed. Lines 6 to 10 consist of program codes that process all 5×5 sub-images as cracks if the ratio of cracks in sub-images is greater than n1, and process them as backgrounds otherwise.

Table 3. User Algorithm I (step 5).

	User Algorithm I (Step 5)
	Input: Binary image im1(i, j) of Step 4 Output: Dilated image im2(i, j) ($1 \leq i \leq m$, $1 \leq j \leq n$)
1	Use the binary image im1(i, j) of step 4
2	z1 = nnz(im1)/(m × n) × 100 ; nnz is function to find number of nonzero
3	n1 = round(z1)
4	if z1 ≤ 1; n1 = 25; end
5	Make sub images 5 × 5 (sub_im) of im1(i, j)
6	if nnz(sub_im) ≥ n1
7	Fill the sub image with 1 (white, crack)
8	else
9	Fill the sub image with 0 (black, background)
10	end

In line 4, if the ratio of cracks (z1) is less than 1%, n1 is set to 25. In this way, cracks are only recognized when all the elements of the 5 × 5 sub-image (sub_im) are cracks (white, 1); otherwise, all are recognized as being in the background (black, 0). This process is executed in lines 6 to 10, and when the ratio of cracks is less than 1%, an erosion operation is performed instead of dilation.

Using this algorithm, the results were excellent not only for crack images but also for non-crack images.

3.2. User Algorithm II (Step 9)

After analyzing the labeled images in the previous step (step 8), it is observed that some labels are difficult to identify as cracks due to their small size. Table 4 presents User Algorithm II, which is used for removing some labels that are difficult to determine as cracks.

Table 4. User Algorithm II (step 9).

	User Algorithm II (Step 9)
	Input: Label image L(i, j) Output: Crack image C(i, j) ($1 \leq i \leq m$, $1 \leq j \leq n$)
1	Use the label image L(i, j) of step 8 ; N is the number of label
2	if N > 20; goto 12; end
3	for (k = 1 to N)
4	for (i = 1 to m)
5	Count the crack size(c1) of column
6	end
7	for (j = 1 to n)

Table 4. *Cont.*

	User Algorithm II (Step 9)
8	Count the crack size(c2) of row
9	end
10	Crack detection $-c1 \geq n/5$ or $c2 \geq m/5$
11	end
12	Stop

Line 10 is a code that classifies cracks when the size of cracks in the row or column direction is more than 1/5 of the total size. By using this algorithm, as shown in Table 1, the image in step 8 becomes the image in step 9, resulting in the detection of only cracks.

In line 2, if the number of labels in the labeled image is 20 or more, the algorithm terminates at line 12, preventing it from being recognized as a crack. For non-crack images, this code helps eliminate cases where labeling is sporadically dispersed due to its properties, causing excessive crack recognition.

4. Discussion

This study developed a program that automatically detects cracks in concrete images through filtering (step 3), using User Algorithm I in step 5, and User Algorithm II in step 9. User Algorithm I in step 3 is designed to dilate the image to improve crack identification, and User Algorithm II in step 9 determines the presence or absence of cracks. By employing these two algorithms and a 10-step process, the detection of cracks in both crack images and non-crack images was analyzed. Table 5 presents the results for crack images, and Table 6 presents the results for non-crack images.

The detection results for the crack images, as shown in Table 5, were analyzed, and 100% detection was achieved with no cracks going undetected. The quality of crack detection was classified into three types: Type C1 represents not bad results, Type C2 represents good results, and Type C3 represents excellent results. Type C3 (excellent results) refers to cases where cracks are perfectly detected, while Type C2 (good results) indicates cases where the cracks are generally detected accurately but may be slightly incomplete, as shown in Table 5. Type C1 (not bad results) represents cases slightly worse than Type C2; in the case of '00527.jpg', a non-cracked part (red in the figure) is also detected as a crack. In 885 out of 1000 images, the crack detection accuracy (Type C3 only) was 88.5%, demonstrating excellent results. Good results were found for Type C3 and Type C2; the accuracy is 95.3% (885 + 68), demonstrating very excellent results.

The results for the non-crack images are presented in Table 6. Type N1 refers to cases identified as cracks, while Type N2 refers to cases not identified as a crack. Type N1 is a case in which non-crack images are recognized as cracks and incorrectly detected. In the case of detecting cracks, 31 out of 1000 images were detected, demonstrating an error rate of 3.1%, and the accuracy was very high: 96.9% (969). Among the 31 non-crack images incorrectly recognized as cracks, there are many cases that look like cracks, such as 00283.jpg, 00318.jpg, 00361.jpg, and 00419.jpg. These images were classified as errors because they were categorized as non-crack images in images of METU campus buildings. Although crack images were incorrectly classified as non-crack images, the detection accuracy of the non-crack image is expected to be over 96.9%.

Table 5. Detection types for crack images (Type C1, C2 and C3).

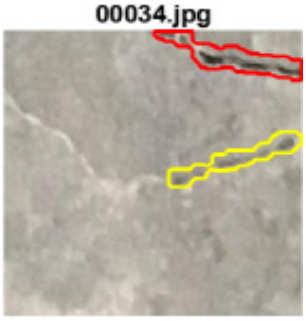
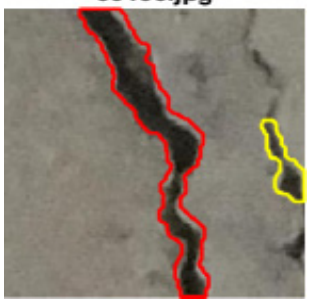
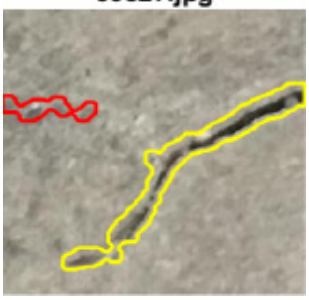
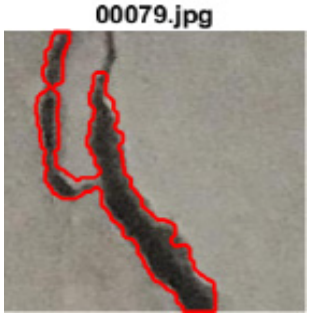
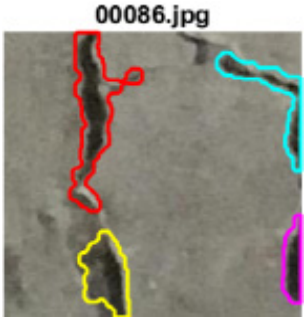
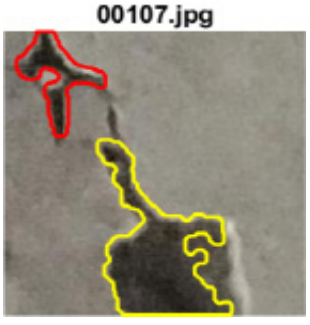
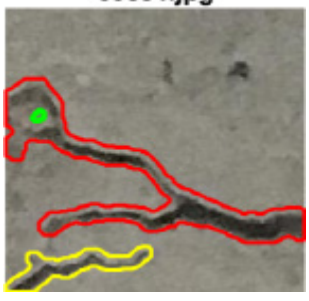
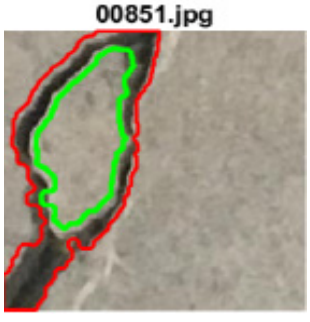
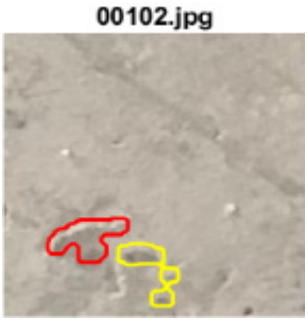
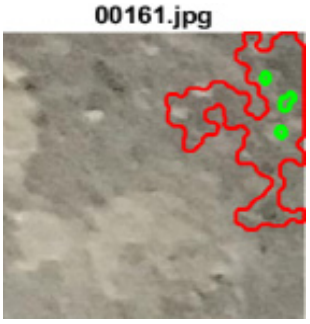
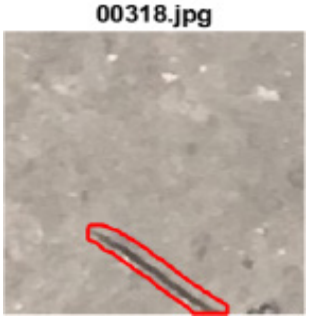
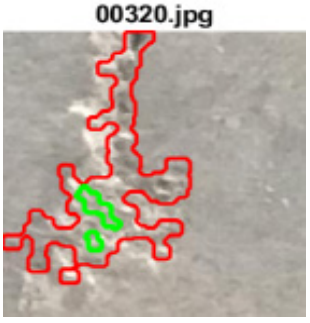
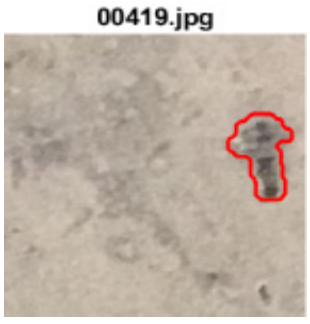
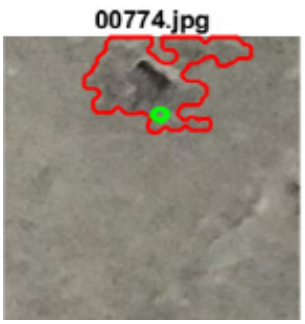
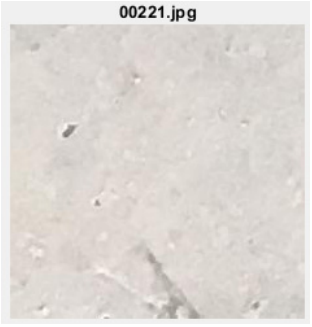
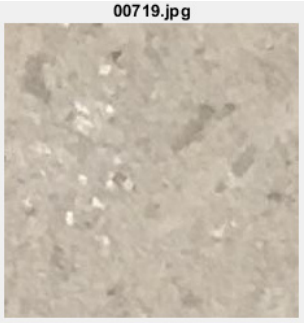
Type C1 (Not bad results) [List and sample images]	34, 35, 55, 58, 100, 101, 103, 144, 145, 269, 272, 305, 311, 328, 349, 393, 395, 410, 414, 422, 452, 482, 497, 504, 512, 527, 533, 538, 561, 564, 587, 592, 605, 615, 625, 657, 680, 704, 707, 719, 784, 808, 824, 835, 909, 918, 929 [No. of Type C1 results: 47 (4.7%)]	00034.jpg  00100.jpg  00145.jpg  00527.jpg 
Type C2 (Good results) [List and sample images]	36, 43, 79, 86, 87, 88, 105, 106, 107, 148, 171, 183, 215, 228, 245, 297, 303, 335, 339, 345, 353, 392, 408, 418, 419, 421, 426, 429, 449, 475, 498, 509, 513, 542, 565, 621, 624, 633, 641, 655, 664, 670, 693, 756, 770, 781, 782, 787, 814, 820, 821, 823, 825, 826, 830, 850, 868, 926, 932, 935, 937, 940, 948, 954, 967, 968, 973, 990 [No. of Type C2 results: 68 (6.8%)]	00079.jpg  00086.jpg  00107.jpg  00418.jpg 
Type C3 (Excellent results) [List and sample images]	All except Type C1 and Type C2 [No. of Type C3 results: 885 (88.5%)]	00326.jpg  00331.jpg  00851.jpg 

Table 6. Detection types for non-crack images (Type N1 and N2).

Type N1 [List and sample images]	47, 101, 102, 161, 242, 251, 268, 283, 290, 318, 320, 324, 361, 375, 376, 412, 419, 432, 472, 558, 609, 708, 710, 770, 774, 781, 807, 840, 871, 953, 964 [No. of bad results: 31 (3.1%)]	         
Type N2 [List and sample images]	All except Type N1 [No. of Type N2 results: 969 (96.9%)]	  

For the 1000 crack images, Table 7 shows detection results in the first 500 images, and Table 8 shows detection results for the last 500. It can be magnified to some extent and viewed as large. Based on the results presented in Table 7, it is expected that the proposed algorithms can effectively detect cracks in concrete images.

Table 7. The results for crack images (00001.jpg~00500.jpg).

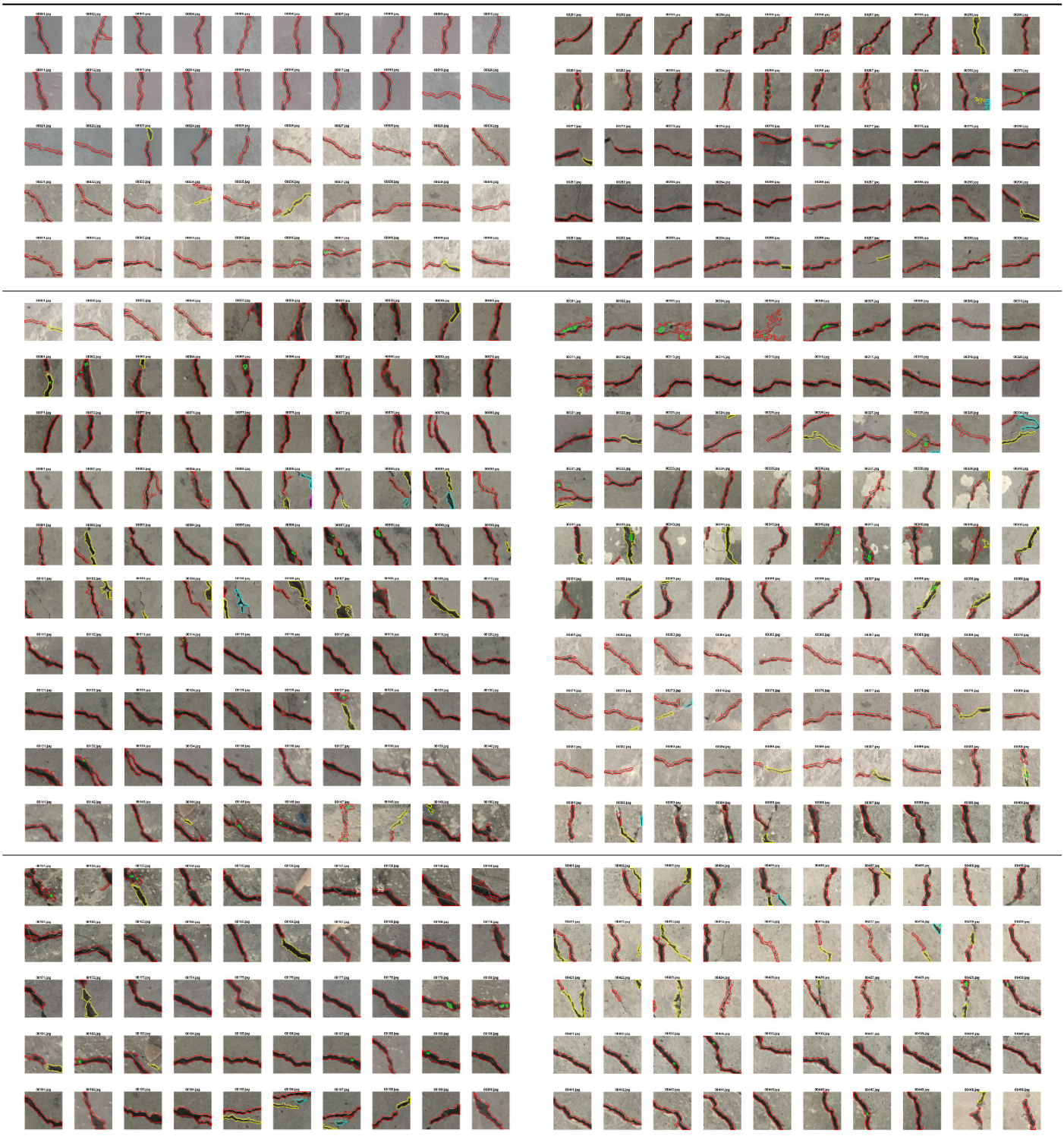


Table 7. *Cont.*

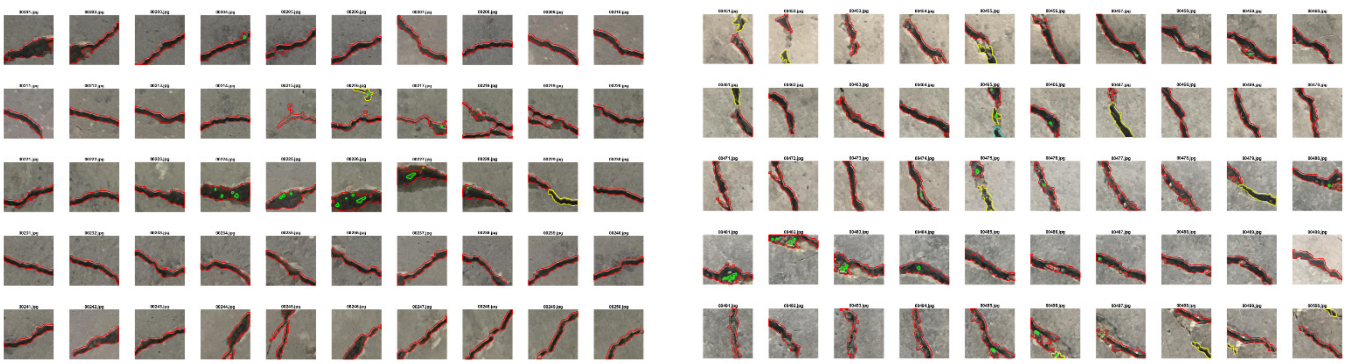


Table 8. The results for crack images (00501.jpg~01000.jpg).

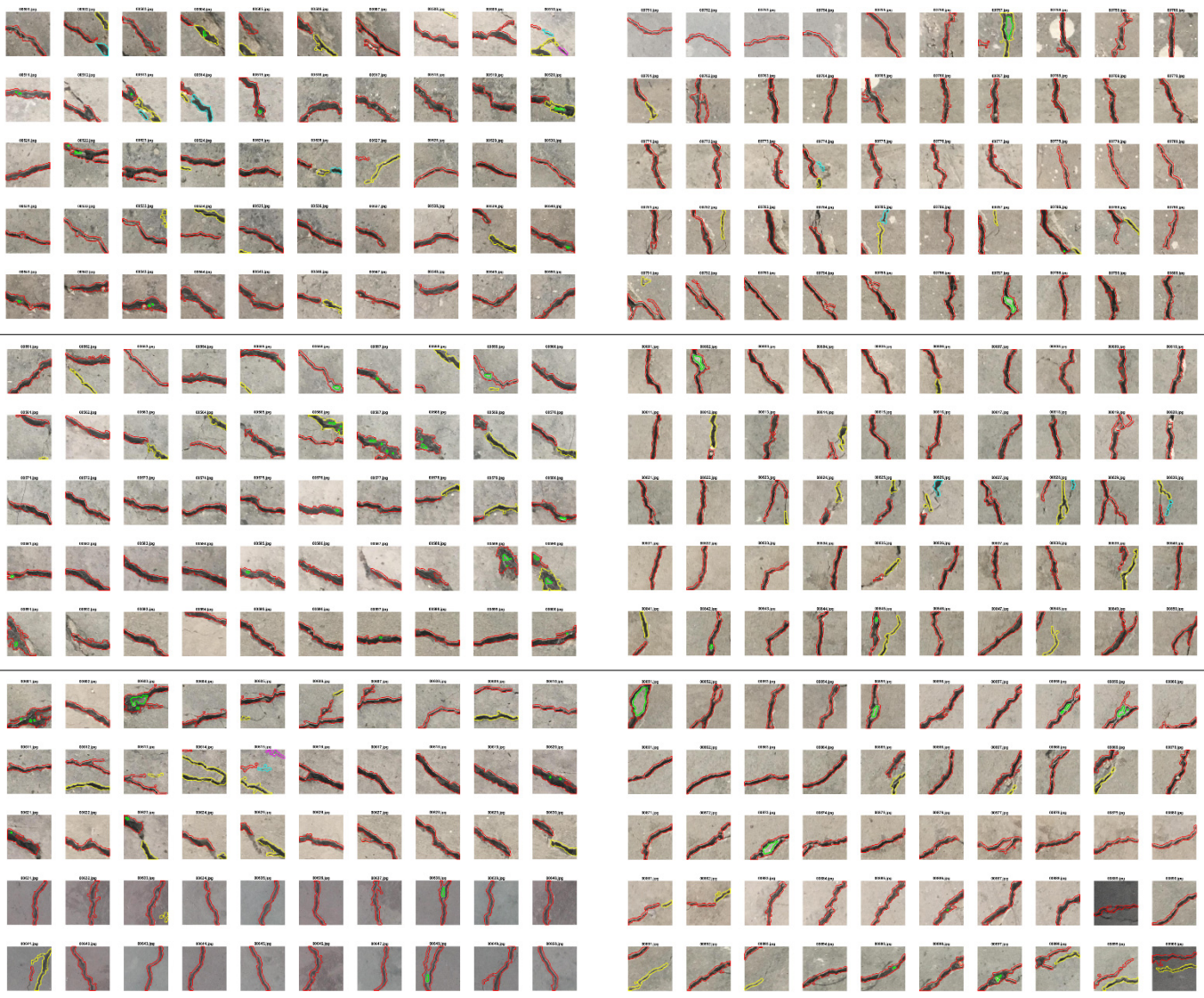
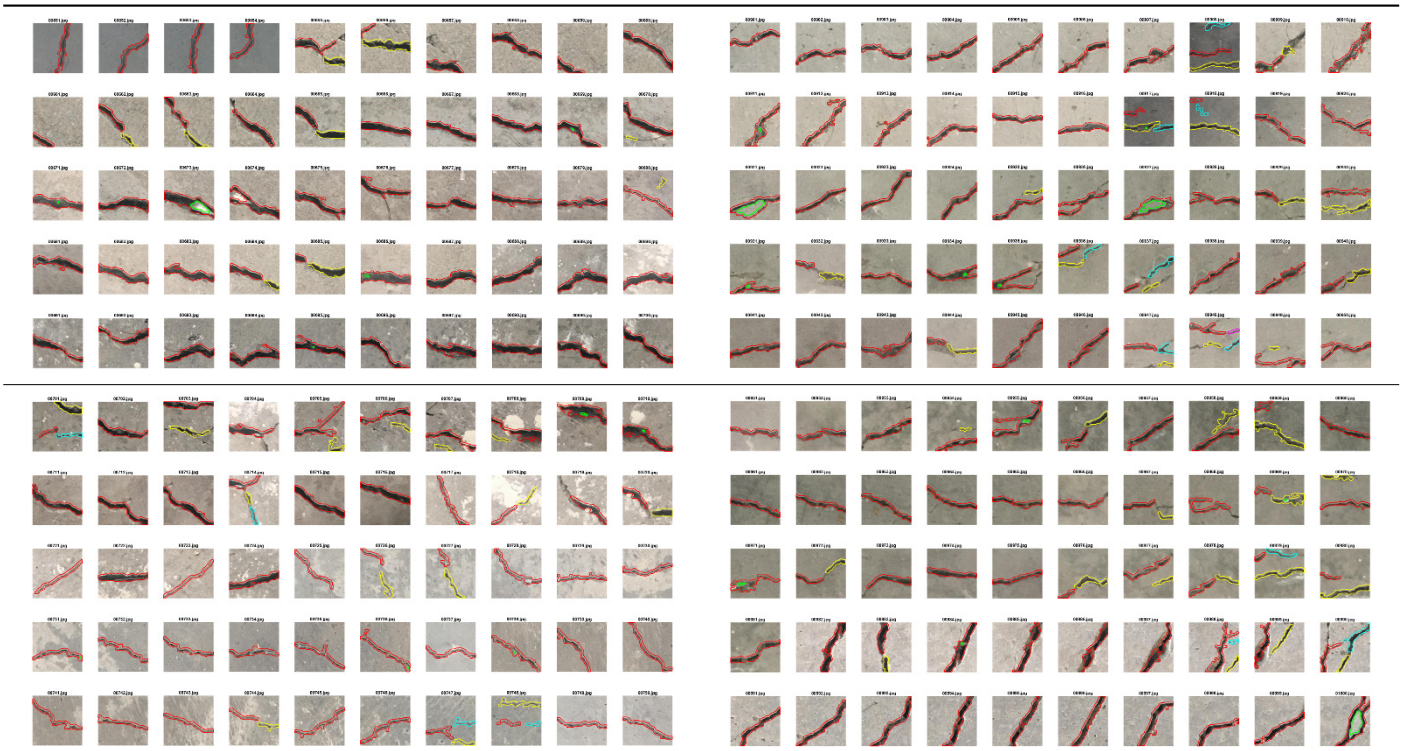


Table 8. *Cont.*



5. Conclusions

This study developed algorithms to detect the size and location of cracks in concrete images. Cracks were automatically detected using a total of 10 steps, and the accuracy of crack detection was improved by adding two user algorithms in steps 5 and 9. A total of 1000 crack images and non-crack images were used, respectively.

Cracks were detected in 100% of the crack images. Even excluding the not bad results (Type C1), it demonstrated outstanding performance with a 95.3% accuracy rate. The accuracy of non-crack images was also remarkably excellent (96.9%).

These results indicate that crack detection is highly effective and accurate. Therefore, it is expected that the crack detection process presented in this study can be used to accurately detect the location and size of cracks in concrete.

Author Contributions: Conceptualization, B.-C.K. and B.-J.S.; methodology, B.-C.K. and B.-J.S.; software, B.-J.S.; validation, B.-C.K.; formal analysis, B.-J.S.; investigation, B.-C.K. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by the Konyang University Research Fund in the second half of 2021 and the National Research Foundation of Korea (NRF-2021R1F1A1061650).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rahai, M.; Esfandiari, A.; Bakhshi, A. Detection of structural damages by model updating based on singular value decomposition of transfer function subsets. *Struct. Control Health Monit.* **2020**, *27*, e2622. [CrossRef]
2. Yu, Y.; Rashidi, M.; Samali, B.; Mohammadi, M.; Nguyen, T.N.; Zhou, X. Crack detection of concrete structures using deep convolutional neural networks optimized by enhanced chicken swarm algorithm. *Struct. Health Monit.* **2022**, *21*, 2244–2263. [CrossRef]
3. Wang, P.; Hu, Y.; Dai, Y.; Tian, M. Asphalt Pavement Pothole Detection and Segmentation Based on Wavelet Energy Field. *Math. Probl. Eng.* **2017**, *2017*, 1604130. [CrossRef]
4. Liu, Y.; Cho, S.; Spencer, B.F., Jr.; Fan, J. Automated assessment of cracks on concrete surfaces using adaptive digital image processing. *Smart Struct. Syst.* **2014**, *14*, 719–741. [CrossRef]
5. Ebrahimkhanlou, A.; Farhidzadeh, A.; Salamone, S. Multifractal analysis of crack patterns in reinforced concrete shear walls. *Struct. Health Monit.* **2016**, *15*, 81–92. [CrossRef]
6. Dow, H.; Perry, M.; McAlorum, J.; Pennada, S.; Dobie, G. Skeleton-based noise removal algorithm for binary concrete crack image segmentation. *Autom. Constr.* **2023**, *151*, 104867. [CrossRef]
7. Chan, B.; Guan, H.; Jo, J.; Blumenstein, M. Towards UAV-based bridge inspection systems: A review and an application perspective. *Struct. Monit. Maint.* **2015**, *2*, 283–300. [CrossRef]
8. Chen, S.; Laefer, D.F.; Mangina, E.; Zolanvari, S.I.; Byrne, J. UAV bridge inspection through evaluated 3D reconstructions. *J. Bridge Eng.* **2019**, *24*, 05019001. [CrossRef]
9. Chow, J.K.; Liu, K.F.; Tan, P.S.; Su, Z.; Wu, J.; Li, Z.; Wang, Y.H. Automated defect inspection of concrete structures. *Autom. Constr.* **2021**, *132*, 103959. [CrossRef]
10. Ali, L.; Harous, S.; Zaki, N.; Khan, W.; Alnajjar, F.; Al Jassmi, H. Performance evaluation of different algorithms for crack detection in concrete structures. In Proceedings of the 2021 2nd International Conference on Computation, Automation and Knowledge Management, ICCAKM, Dubai, United Arab Emirates, 19–21 January 2021; pp. 53–58. [CrossRef]
11. Hsieh, Y.-A.; Tsai, Y.J. Machine learning for crack detection: Review and model performance comparison. *J. Comput. Civ. Eng.* **2020**, *34*, 04020038. [CrossRef]
12. Dorafshan, S.; Thomas, R.J.; Maguire, M. Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. *Constr. Build. Mater.* **2018**, *186*, 1031–1045. [CrossRef]
13. Kheradmandi, N.; Mehranfar, V. A critical review and comparative study on image segmentation-based techniques for pavement crack detection. *Constr. Build. Mater.* **2022**, *321*, 126162. [CrossRef]
14. Zhang, L.; Yang, F.; Zhang, Y.D.; Zhu, Y.J. Road Crack Detection Using Deep Convolutional Neural Network. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016. [CrossRef]
15. Özgenel, Ç.F.; Gönenç Sorguç, A. Performance Comparison of Pretrained Convolutional Neural Networks on Crack Detection in Buildings. In Proceedings of the 2018 Proceedings of the 35th ISARC, Berlin, Germany, 20–25 July 2018.
16. Mathworks. *MATLAB Program. Image Processing Toolbox (R2023a)*; Mathworks: Natick, MA, USA, 2023.
17. Dong, X.; Wang, G.; Pang, Y.; Li, W.; Wen, J.; Meng, W.; Lu, Y. Fast efficient algorithm for enhancement of low lighting video. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), Barcelona, Spain, 11–15 July 2011; pp. 1–6.
18. He, K. Single Image Haze Removal Using Dark Channel Prior. Ph.D. Thesis, The Chinese University of Hong Kong, Hong Kong, China, 2011.
19. Park, D.; Park, H.; Han, D.K.; Ko, H. Single Image Dehazing with Image Entropy and Information Fidelity. In Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.