remote sensing

MDPI

Article

A New Region-Based Minimal Path Selection Algorithm for Crack Detection and Ground Truth Labeling Exploiting Gabor Filters

Gonzalo de León *, Nicholas Fiorentini, Pietro Leandri and Massimo Losa

Department of Civil and Industrial Engineering (DICI), Largo Lucio Lazzarino 1, University of Pisa, 56122 Pisa, Italy; nicholas.fiorentini@ing.unipi.it (N.F.); pietro.leandri@unipi.it (P.L.); massimo.losa@unipi.it (M.L.)

* Correspondence: gonzalo.deleon@phd.unipi.it

Abstract: Cracks are fractures or breaks that occur in materials such as concrete, metals, rocks, and other solids. Various methods are used to detect and monitor cracks; among many of them, image-based methodologies allow fast identification of the distress and easy quantification of the percentage of cracks in the scene. Two main categories can be identified: classical and deep learning approaches. In the last decade, the tendency has moved towards the use of the latter. Even though they have proven their outstanding predicting performance, they suffer some drawbacks: a "black-box" nature leaves the user blind and without the possibility of modifying any parameters, a huge amount of labeled data is generally needed, a process that requires expert judgment is always required, and, finally, they tend to be time-consuming. Accordingly, the present study details the methodology for a new algorithm for crack segmentation based on the theory of minimal path selection combined with a region-based approach obtained through the segmentation of texture features extracted using Gabor filters. A pre-processing step is described, enabling the equalization of brightness and shadows, which results in better detection of local minima. These local minimal are constrained by a minimum distance between adjacent points, enabling a better coverage of the cracks. Afterward, a region-based segmentation technique is introduced to determine two areas that are used to determine threshold values used for rejection. This step is critical to generalize the algorithm to images presenting close-up scenes or wide cracks. Finally, a geometrical thresholding step is presented, allowing the exclusion of rounded areas and small isolated cracks. The results showed a very competitive *F*1-score (0.839), close to state-of-the-art values achieved with deep learning techniques. The main advantage of this approach is the transparency of the workflow, contrary to what happens with deep learning frameworks. In the proposed approach, no prior information is required; however, the statistical parameters may have to be adjusted to the particular case and requirements of the situation. The proposed algorithm results in a useful tool for researchers and practitioners needing to validate their results against some reference or needing labeled data for their models. Moreover, the current study could establish the grounds to standardize the procedure for crack segmentation with a lower human bias and faster results. The direct application of the methodology to images obtained with any low-cost sensor makes the proposed algorithm an operational support tool for authorities needing crack detection systems in order to monitor and evaluate the current state of the infrastructures, such as roads, tunnels, or bridges.

Keywords: minimal path; Gabor filter; automatic crack detection; ground truth; labeling; deep learning; image processing; image segmentation; image analysis; region-based

Citation: de León, G.; Fiorentini, N.; Leandri, P.; Losa, M. A New Region-Based Minimal Path Selection Algorithm for Crack Detection and Ground Truth Labeling Exploiting Gabor Filters. *Remote Sens.* **2023**, *15*, 2722. https://doi.org/10.3390/rs15112722

Academic Editors: Dong Liu and Pedro Melo-Pinto

Received: 14 April 2023
Revised: 19 May 2023
Accepted: 22 May 2023
Published: 24 May 2023

1. Introduction

Cracks are fractures or breaks that occur in materials such as concrete, metals, rocks, and other solids. They can occur due to various factors, including structural stresses, temperature changes, chemical reactions, or mechanical damage. Cracks can range from

hairline fractures to large, gaping openings, and can significantly affect the strength and stability of materials [1,2]. Various methods are used to detect and monitor cracks, including visual inspection, nondestructive testing, and advanced imaging techniques.

In civil engineering, cracks in buildings and infrastructure can be a serious issue as they can lead to structural instability [3,4], water infiltration, noise performance decrease [5], surface roughness [6], and other problems. Understanding the causes and mechanisms of crack formation is essential for preventing and mitigating their impact [7].

Cracks have an intrinsic randomness that makes them unpredictable and hard to model and assess [8]. Nevertheless, they present a common property; they grow following a linear shape or a set of gridded lines that break the homogeneity of the surface. These breaks generate a fast rate of change in the slope of the surface, provoking an obstacle for the incoming light, which can be seen by the human eye as a dark area.

This aspect makes image-based methodologies very suitable and appropriate for the task of automatic detection systems [9,10], allowing fast identification of the distress and easy quantification of the percentage of cracks in the scene [11,12]. Various image processing and analysis techniques can be used to detect and analyze cracks in images. These techniques include edge detection [13–15], thresholding [16–19], morphological operations [20–22], wavelet-decomposition [23–25], and machine-learning-based approaches among the most diffused ones [26–28].

The tendency in the last decade has moved towards the use of these last ones, especially deep learning techniques [29–34]. Even though they have proven their value in predicting performance, they suffer some drawbacks. Generally, neural networks are referred to as "black-box" systems due to the lack of insight in the intermediate layers, leaving the user only with the final result [35–37]; however, in recent years, great efforts have been made to obtain a more complete view of these intermediate processes [38,39]. When the outcome is not accurate, it is hard to explain the rationales behind it. Moreover, another concern associated with this type of technique lies in the huge amount of labeled data required [7,26]. Usually, this labeling process requires expert judgment and tends to be very tedious. Finally, a major issue often neglected is the human error inherited by the system [40]. Whereas in a more simple task such as classification, i.e., humans, animals, or vehicles in images, this problem does not arise, for the particular case of road cracks segmentation a high accuracy is needed; therefore, human error becomes more noticeable when the results are analyzed in detail. In order to avoid these biases and the tedious task of labeling by hand, some researchers have followed the path of traditional methods.

In [41], the authors developed an image processing toolbox for crack detection and characterization. The toolbox is structured in four main modules: (i) image preprocessing, including algorithms for image smoothing, white lane markings detection, pixel intensity normalization and saturation; (ii) crack detection, based on pattern classification techniques; (iii) crack characterization into types, notably classifying detected cracks as longitudinal, transversal, or miscellaneous, and including a severity-level assignment; (iv) evaluation routines, to compute ROC curves, and standard metrics such as recall, precision, or F-measure.

In [42,43], the authors proposed a minimal path searching (MPS) algorithm by considering solely the darkest points in a window kernel as nodes. Then, the path between adjacent nodes was retrieved using a *Dijkstra* algorithm [44] based on an ad hoc cost function. The algorithm was developed to work for thin cracks, under a fixed view position, and with a fixed scale. Over time, some improvements have been proposed, with the aim of enhancing computational time [45,46] and minimizing the number of local minima [47,48].

In another study [49], a scheme based on the Gabor filter for crack detection was proposed; specifically, a filter bank consisting of multiple oriented Gabor filters was used. Results reported in this paper only showed a couple of pictures from two different acquisition systems, with values of precision and recall up to 95% and 78%, respectively for the LRIS (Laser Road Imaging System) (https://www.pavemetrics.com/wp-content/uploads/2016/03/LRIS_Flyer.pdf) acquisition system and 66% and 90% with a commercial camera.

Generally, these studies working with traditional approaches develop the methodologies to fit their particular exigences and the results are often compared only with instances belonging to their dataset without much variance. In this study, the capacity for generalization of the algorithm was a key desired feature. For this reason, the *DeepCrack* (DC) dataset was chosen to analyze the performance of our results.

The current work details a new methodology to obtain high-precision crack segmentation from images. To perform this, a new algorithm, based on the theory of minimal path selection combined with a region-based approach obtained through segmentation of texture features extracted using Gabor filters, was developed; the given name is *Region-Based Minimal Path Searching* (RB-MPS). Among the intermediate steps in the algorithm, a new pre-processing step defined as *PCA Gaussian* is presented, enabling equalization of brightness and shadows in the scene, which results in better detection of local minima. These local minimal are constrained by a minimum distance between adjacent points, enabling a better coverage of the cracks. Afterward, a region-based segmentation technique is introduced to determine two areas that are used to determine threshold values used for rejection. This step is critical to generalize the algorithm to images presenting close-up scenes or wide cracks. Finally, a thresholding step based on geometrical properties is presented; this step allows for excluding rounded areas and small isolated cracks. This improvements are summarized as follows:

1. Preprocessing: Brightness and shadow equalization using *PCA Gaussian*.
2. Textural features extracted using Gabor filters as input for the detection of local minima.
   - Minimal distance between two adjacent minima.
   - Region-based local minima (RBLM).
3. New region-based rejection strategy.
4. Final thresholding based on area and eccentricity properties.

The main advantage of the proposed approach is the transparency of the workflow, contrary to what happens with deep learning frameworks. In the proposed approach, no prior information is required; however, the statistical parameters may have to be adjusted to the particular case and requirements of the situation. The proposed algorithm results in a useful tool for researchers and practitioners needing to validate their results against some reference or needing labeled data for their models. Moreover, the current study could establish the grounds to standardize the procedure for crack segmentation with a lower human bias and faster results. The direct application of the methodology to images obtained with any low-cost sensor makes the proposed algorithm an operational support tool for authorities needing crack detection systems in order to monitor and evaluate the current state of the infrastructures, such as roads, tunnels, or bridges.

2. Theory Background

In this section, we proceed to briefly describe the theoretical background for the development of the RB-MPS algorithm. The authors believe that including the theory and pseudocode of previous work enlightens readers regarding the development and improvements of the proposed work. The pseudocode can be found in Appendix A. The following subsections expose the concepts behind the original *Dijkstra* algorithm, *Gabor* filters, and the approach for minimal path finding for crack detection.

2.1. Dijkstra Algorithm

In 1959, the computer scientist Edsger W. Dijkstra published, in a three-page article, his work for solving the problem of finding the shortest route between two cities in the Netherlands using a simplified map or, in a more generic form, the problem to find the minimal path between two points given some constraints [44]. The algorithm is based on graph theory, where the graphs are data structures that are used to depict connections between nodes (or vertex) and are connected amid them between edges.

Starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost. It achieves this by maintaining a tree of paths originating at the start node and extending those paths one edge at a time, based on the lowest cost, until its termination criterion is satisfied.

A relevant application of the algorithm that resembles the one applied to cracks is a maze problem, where the start and exit are the source and destination nodes, the walls of the maze are the constraints where the path cannot pass through, and the goal is to obtain the shortest distance to the exit. The general idea of the algorithm is listed in the following steps and further detailed in the pseudocode Algorithm A1.

1. The algorithm starts by setting for all the nodes the **cost** to **infinity** and the attribute **visited** to **false**, meaning that they have not been visited yet. The algorithm will try to improve them step by step.
2. The cost from **source** to itself is set to **zero**. In this way, the algorithm starts from the source node.
3. Create a set **Q** of all the unvisited nodes.
4. Iterate over **Q**.
5. Find the node **u** with the smallest **cost**. In the first iteration, this will be the source.
6. Mark the current node as visited.
7. Remove **u** from the queue in **Q**.
8. For the current node, consider all of its unvisited neighbors and calculate their tentative **cost** through the current node.
9. Compare the newly calculated tentative distance to the one currently assigned to the neighbor and assign the smaller one.
10. Add the current node **u** to the **path**.
11. End when the **target** node has been visited or if every node in the unvisited set is unreachable.

2.2. Gabor Filters for Crack Detection

The Gabor filters, named after Dennis Gabor, are linear filters used in myriad image processing applications for edge detection, texture analysis, feature extraction, etc. They are governed by the basic conception of vision information processing by multiple channel filtration in the mammalian visual structure. These filters have been shown to possess optimal localization properties in both spatial and frequency domains, and, thus, are well-suited for texture segmentation problems.

Gabor filters are special classes of band-pass filters, i.e., they allow a certain "band" of frequencies and reject the others. A Gabor filter can be viewed as a sinusoidal signal of particular frequency and orientation, modulated by a Gaussian wave. The general form of the Gabor filter is

$$g(x,y) = s(x,y)w(x,y) \tag{1}$$

where $s(x,y)$ is the Gaussian envelope and $w(x,y)$ is a complex sinusoidal.

The Gaussian envelope can be termed as

$$s(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}[(\frac{x'}{\sigma_{x'}})^2+(\frac{y'}{\sigma_{y'}})^2]} \tag{2}$$

where $x' = x\cos\theta + y\sin\theta$; $y' = -x\sin\theta + y\cos\theta$, with θ being the angle and σ_x, σ_y the scale factors of the neighborhood. On the other hand, using Euler's identity, the complex sinusoidal can be termed as

$$w(x,y) = e^{j(2\pi\omega_0 x'+\phi)} = \cos(2\pi\omega_o x' + \phi) + j\sin(2\pi\omega_o x' + \phi) \tag{3}$$

where ω is the frequency and ϕ is the phase offset.

From Equations (1)–(3), the Gabor filter consists of a real component and an imaginary component.

The real part is termed as

$$g_r(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}[(\frac{x'}{\sigma_{x'}})^2+(\frac{y'}{\sigma_{y'}})^2]} \{\cos[2\pi\omega_0(x\cos\theta + y\sin\theta) + \phi]\} \quad (4)$$

The imaginary component of the complex Gabor filter is termed as

$$g_i(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}[(\frac{x'}{\sigma_{x'}})^2+(\frac{y'}{\sigma_{y'}})^2]} \{\sin[2\pi\omega_0(x\cos\theta + y\sin\theta) + \phi]\} \quad (5)$$

Equations (4) and (5) can be jointly written in terms of aspect ratio (γ) (i.e., the ellipticity of the Gabor filter) and wavelength (λ) of the sinusoidal factor:

$$g(x,y) = \frac{\gamma}{2\pi\sigma^2} e^{-\frac{1}{2}[\frac{x'^2+\gamma^2y'^2}{\sigma^2}]} e^{j(\frac{2\pi x'}{\lambda}+\phi)} \quad (6)$$

where $\gamma = \frac{\sigma_x}{\sigma_y}$.

Particularly for crack detection, a filter bank is generated using Gabor filters. The filter bank contains a multiorientation filter. The number of orientations depends on the application where the Gabor filter is applied. As the number of orientations increases, the output results are more accurate, but the computational time and complexity of the system increase. All other parameters used to generate Gabor kernels are defined experimentally. The Gabor filter of a given orientation is then convoluted with the input preprocessed image. After the completion of convolution, the real component of the response out of the kernel is thresholded to generate a binary output image.

Finally, the binary images resulting from the differently oriented filters are combined by logical OR operation to produce an output image that contains detected crack segments.

2.3. Minimal Paths for Crack Detection

In this section, we limit ourselves to describe the core steps of the algorithm presented in [42,43] (this is the author's interpretation and the reader is referred to the quoted articles for a deeper understanding). For a deeper understanding, the pseudocode is presented in Algorithm A2. The algorithm consists mainly of the following steps:

1. Definition of the kernel size.
2. Identification of local minima based on the statistic of the whole image.
3. Estimation of potential minimal paths and associated cost using *Dijkstra* algorithm considering the local minima as nodes.
4. Potential paths rejection or acceptance based on a statistic of the costs of potential minimal paths.
5. Skeletonization of the partial result and cleaning of the spurious "branches" and "loops".
6. Width growth by absorbing dark pixels neighboring the currently detected skeleton, according to a threshold test.

3. Materials and Methods

During the present chapter, images from a benchmark dataset named after their work in [50,51], "DeepCrack", were used. Unlike most traditional methods that use their own dataset, this database is composed of 537 labeled images of cracks of different natures, ranging from roads to walls, with a high variance and different scale. This dataset constitutes one of the most diffused datasets in the literature. In the following chapter, different examples drawn from this dataset are used to expose the different steps in the new proposed methodology.

3.1. Preprocessing: Brightness and Shadow Equalization

The first addition proposed by the author is a preprocessing step, namely, *flat-field*, in order to homogenize bright and shadow areas in the image. This preprocessing is required since these artifacts have an important impact on the steps of the methodology, from [43]: "*We suppose that images have no lighting defaults, i.e., halos or a nonuniform lighting are removed by a preprocessing step, if needed*".

Brightness is the perception elicited by the luminance of an object, while shadows are dark areas where light is blocked by an opaque object. Usually, these effects can affect the results or methodologies proposed; therefore, the need to obtain an artifact-free image is markedly significant.

The key concept we apply in this work is a simple but effective approach. The components are firstly modeled by a smoothed version of the image. Subsequently, the image is corrected by a point-wise division between the image and the component. Finally, the corrected image is scaled using the mean of the original image, as can be appreciated in Equation (7):

$$I_{FF} = \bar{I}\left(I \circ S^{-1}\right) \tag{7}$$

where $\bar{I}$ is the mean value of the image, S^{-1} is the inverse of the shading component, and the Hadamard product $\circ$ is used to express the point-wise multiplication.

In order to smooth the image, a Gaussian filter is applied; this filter needs a value σ, which determines the degree of smoothing applied to the image. The determination of σ is far from being trivial since a small value of sigma includes finer detail and yields a shading component too similar to the image, leading to a flat-fielded image that shifts and narrows the histogram too much. On the other hand, a higher value of sigma does not have any influence and yields a result similar to the original input image. Figure 1 elucidates the effect of underestimation of sigma with a value of $\sigma = 2$.

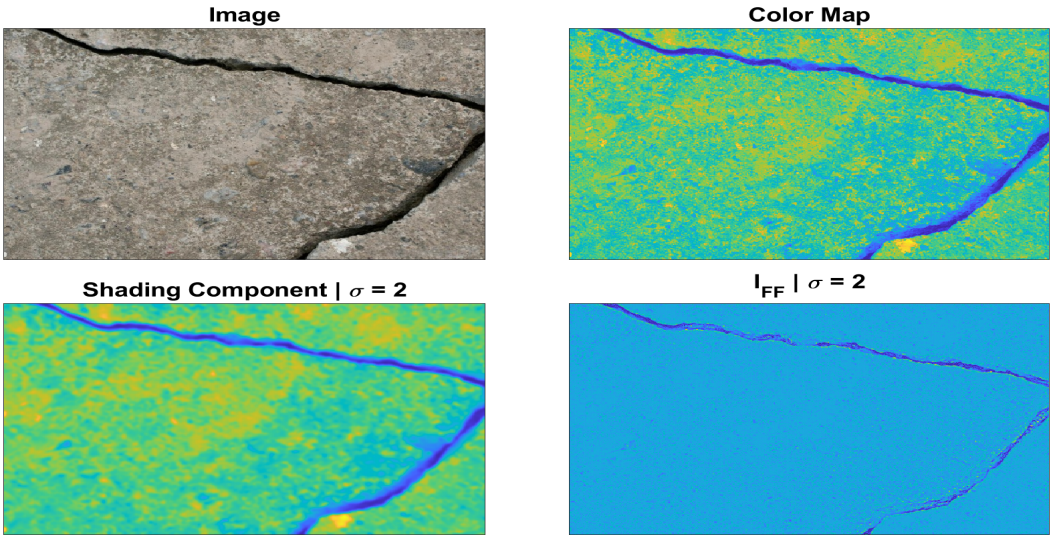

Figure 1. Flat-field underestimation of $\sigma = 2$. **Up left:** image. **Up right:** color map. **Down left:** shading component. **Down right:** flat-fielded image.

It can be noticed how the shading component obtained using a small value of sigma produces a similar image to the original one; therefore, the point-wise multiplication yields the values close to one, which leads to the resulting flat-fielded image being close to the mean original value without much dynamic range.

On the other hand, Figure 2 elucidates the effect of overestimation of sigma with a value of $\sigma = 50$. In this case, it can be noticed how the shading component obtained using a big value of sigma produces uncorrelated values with respect to the position of the

shadow/bright areas. This leads the resulting flat-fielded image to be close to the original image without much correction.

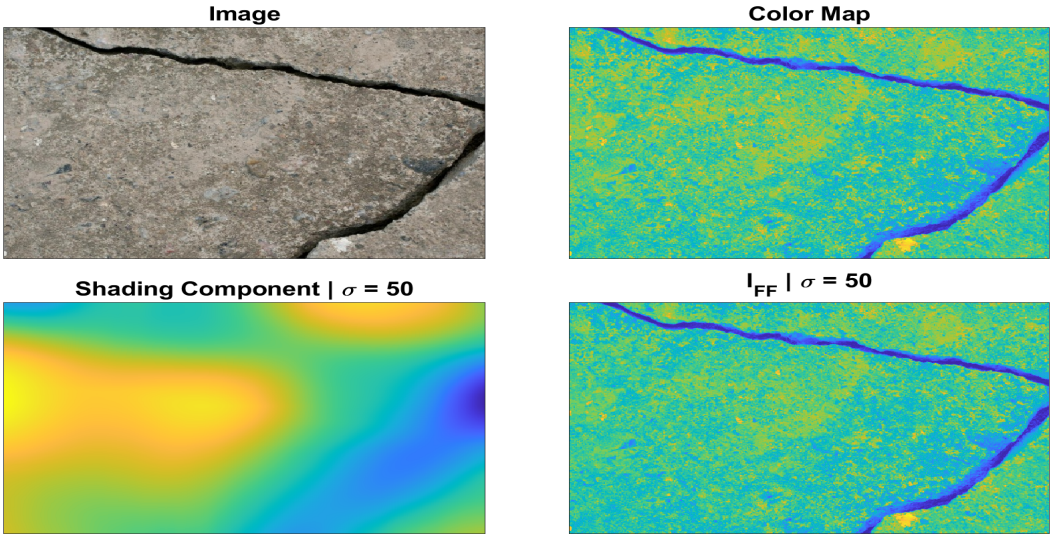

Figure 2. Flat-field overestimation with $\sigma = 50$. **Up left:** image. **Up right:** color map. **Down left:** shading component. **Down right:** flat-fielded image.

In order to solve this issue, the authors propose a new approach and assign the name *PCA Gaussian* to it. The methodology consists of creating a bank of Gaussian's filtered images with different sigma values. Then, a principal component analysis (PCA) analysis allows obtaining the coefficients to combine the different levels into a unique feature, keeping the spatial information. The methodology is analogous to working with a multilevel pyramid scheme, in which the different levels detect different frequencies in the image.

Figure 3 shows the Gaussian bank for the presented example. It can be noticed how the details in the filtered image decrease as the sigma increases.

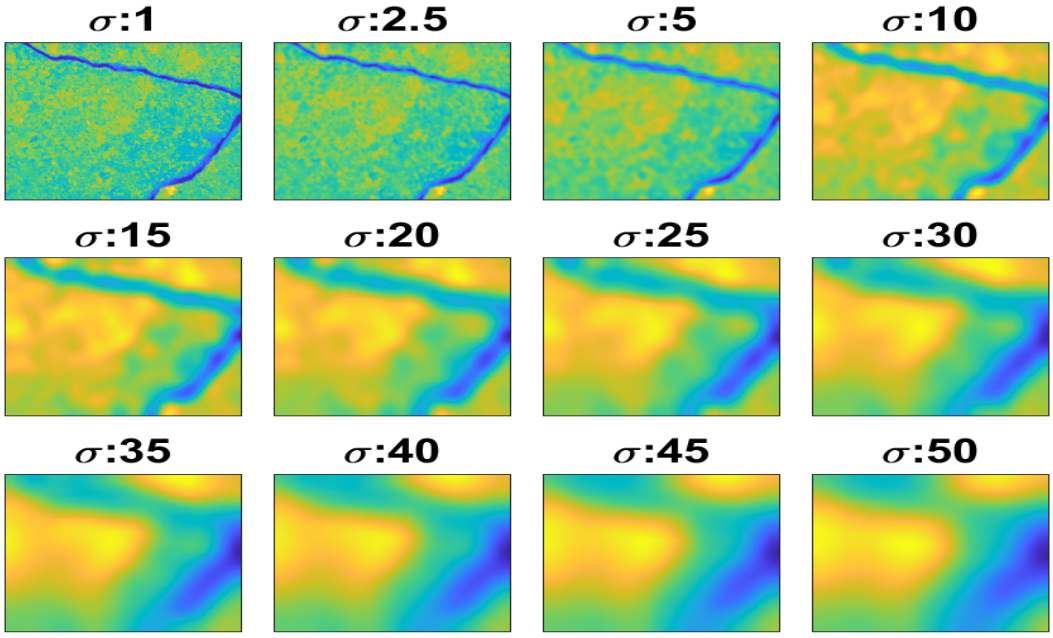

Figure 3. Gaussian bank.

Figure 4 presents the results for this example. In the bottom left part of the figure, the resulting shading component obtained with the *PCA Gaussian* methodology is displayed.

It can be appreciated how the new shading component combines the low- and the high-frequency components, yielding a better description of the brightness/shading phenomena. The bottom right part of the figure presents the corrected flat-fielded image. The result maintains the darkness within the cracks and corrects the brighter parts in the rest of the scene.

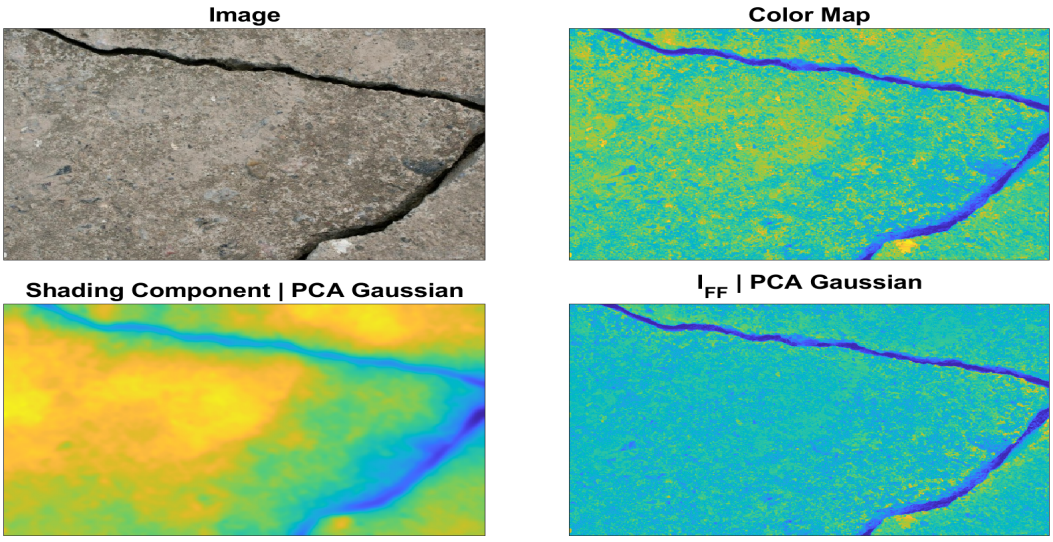

Figure 4. Flat-fielded image with PCA Gaussian. **Up left:** image. **Up right:** color map. **Down left:** shading component. **Down right:** flat-fielded image.

3.2. Textural Features Using Gabor Filters

Texture in the context of image analysis is considered as the regular repetition of an element or pattern on a surface. The use of textural features extracted with Gabor filters yields valuable extra information that can be used for many different purposes.

In this work, the goal is to extract the textural features with the Gabor filters in order to identify regions with different probabilities for cracks to develop. We follow the basic approach described in [52] to perform texture segmentation. Firstly, a bank of Gabor filters is designed using different frequencies (λ) and orientations (θ). Each pair of frequencies and orientations localizes different information in the input image. Regularly sampled orientations between 0° and 150° in steps of 30° are considered, while for wavelength, increasing powers of two starting from $4/\sqrt{2}$ up to the hypotenuse length of the input image are taken into account. These combinations of frequency and orientation are taken from [52]. Only the magnitude response of the filter is considered. The aspect ratio of Gaussian in the spatial domain (γ) defines the ratio of the semimajor and semiminor axes of the Gaussian envelope. This parameter controls the ellipticity of the Gaussian envelope. In this work, a value of $\gamma = 2$ was found to yield better results due to the similarities between the elliptical shape of the filter and the cracks. Figure 5 shows the filter bank.

Each row represents the orientation of the filter and each column represents the wavelength. For a better representation, each kernel was resized, and on the top row for each column, the size of the kernel is shown. Furthermore, it can be noticed how the columns at higher frequencies are richer in detail.

Figure 6 closely shows the 30th filter, for the wavelength $\lambda = 90.51$, orientation $\theta = 120^\circ$, and aspect ratio $\gamma = 2$. On the left side of the figure, the 3D shape of the filter can be appreciated.

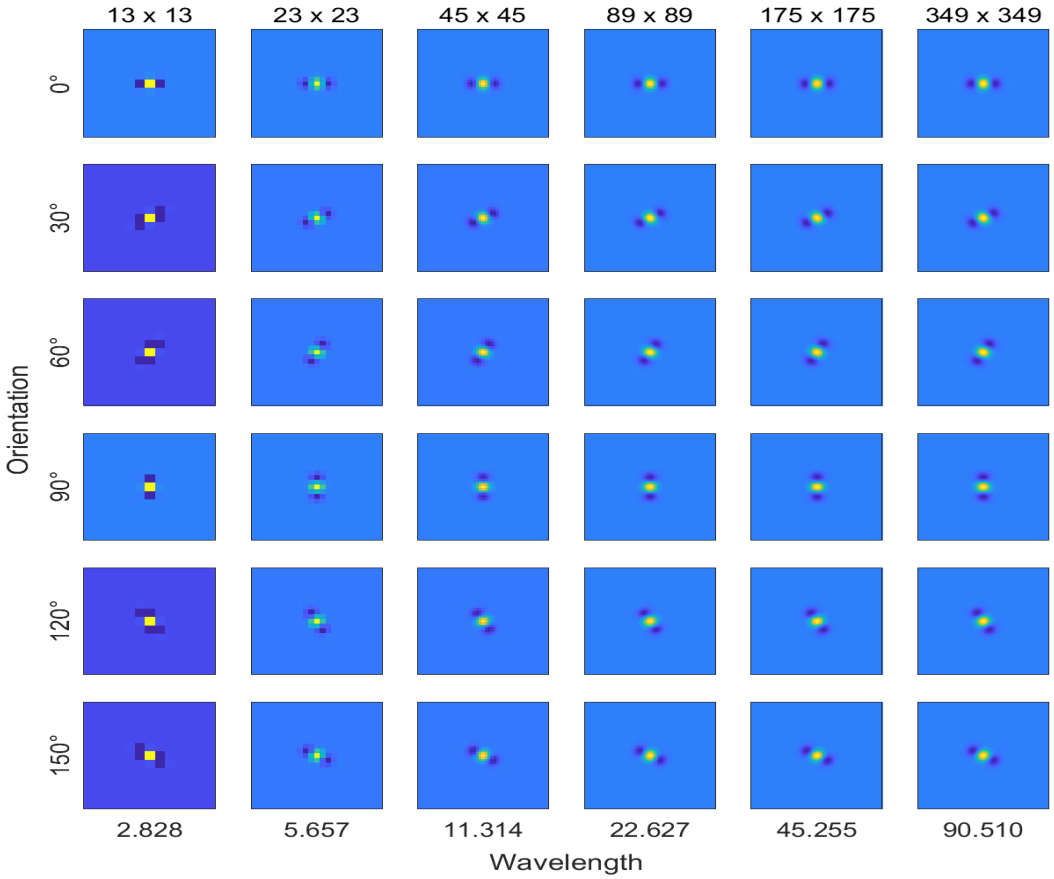

Figure 5. Gabor filter banks. Kernel size at top of each column. X-axis wavelengths vs. Y-axis orientation.

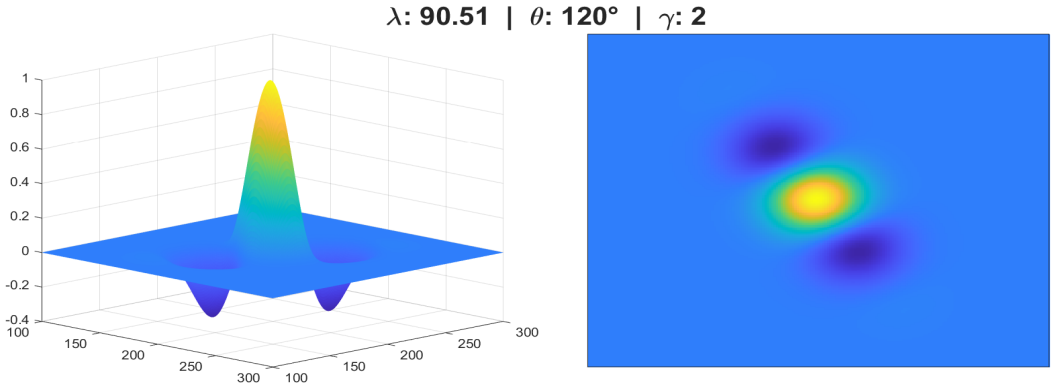

Figure 6. Gabor filter real part at $\lambda = 90.51$, $\theta = 120^\circ$ and $\gamma = 2$.

In order to avoid local variations, each channel is filtered using a Gaussian low-pass filter to smooth the Gabor magnitude information. Sigma is chosen to match the Gabor filter that extracted each feature. A total of 36 features were extracted at different frequencies and orientations. Figure 7 presents one new example and exposes the result of the filtered response.

The resulting filtered magnitudes are concatenated in depth alongside the spatial information. This additional information help to group regions that are spatially closer. PCA is used to move from a 36D representation of each pixel in the input image into a 1D intensity value for each pixel. Figure 8 shows the output obtained for the example being analyzed.

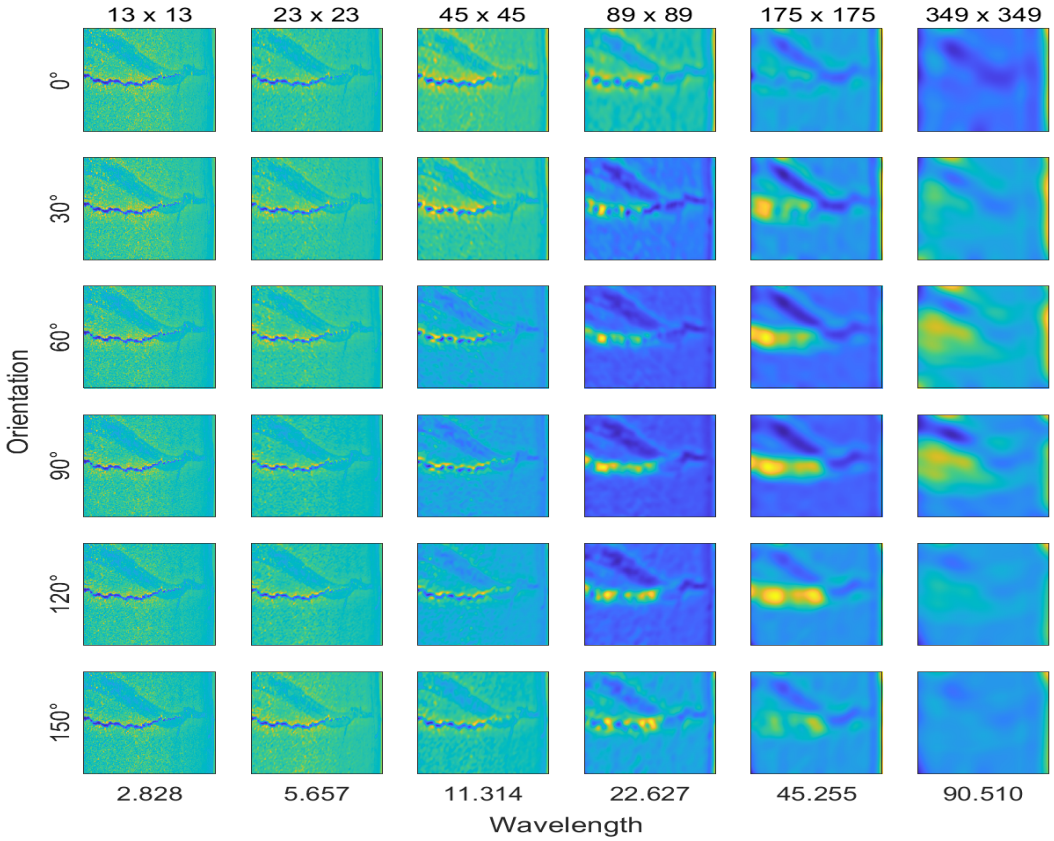

Figure 7. Filtered Gabor magnitude response. Kernel size at top of each column. X-axis wavelengths vs. Y-axis orientation.

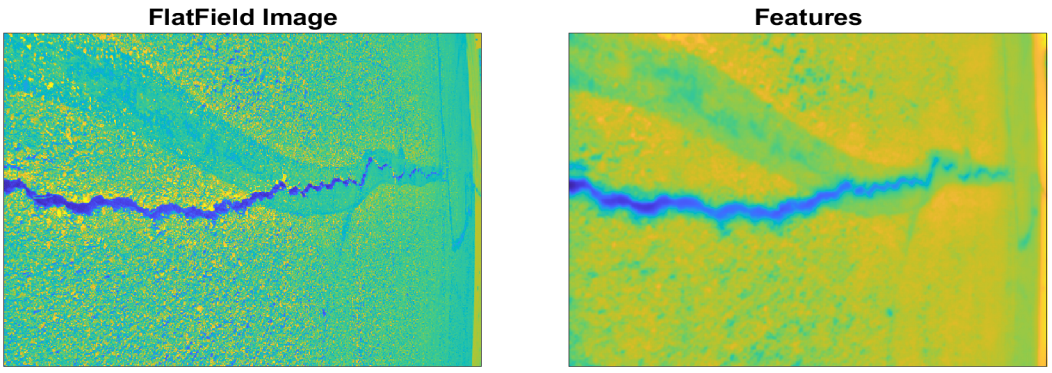

Figure 8. Gabor textural feature. **Left:** flat-field image. **Right:** Gabor texture features.

Finally, we present in Figure 9 the difference between the Gabor features extracted on the original image against the Gabor features extracted from the flat-field corrected image. The longest wavelengths can influence the extraction of features whenever the global gradient of brightness is important, such as in the example presented. It is intuitive from the image to understand the effect the flat-field correction has on the results, yielding more pattern-orientated, rather than intensity-orientated, results. In other words, it gives more weight to the cracks/roughness rather than the gradient of intensities.

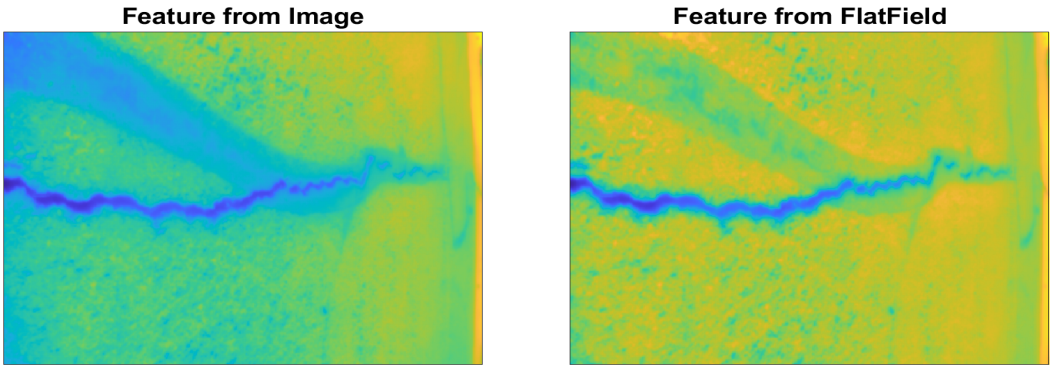

Figure 9. Gabor textural features. **Left:** without flat-field correction. **Right:** with flat-field correction.

3.3. Local Minima

Local minima detection is the process where the lowest values of intensity are found within a small area. The minimum value is compared against a threshold obtained analyzing the statistics of the image. From [43], the window size was chosen to be 8×8, which yields results that balance accuracy and computational time, while the threshold is obtained as follows: $S_a = \mu_a - \sigma_a$, where μ_a is the mean intensity of the image and σ_a is its standard deviation. Local minima detection is a key step in RB-MPS. Indeed, the detected local minima are used as nodes in the path finding algorithm. Therefore, an underestimation of the nodes leads to potential missed paths, and an overestimation leads to a higher concentration of short paths, which overloads the computational cost and biases the statistical estimation used for rejection. Moreover, a high concentration of nodes leads to a generation of closed paths. This issue is not considered in the previous version of the algorithm [42,43]. The new modifications proposed in this step of the study are mainly two:

- Minimal distance between two adjacent minima.
- Detection of local minima from two regions.

3.3.1. Adjacent Local Minima

The procedure for the original local minima is presented in Figure 10 with a new instance from the dataset. As can be appreciated, the windows are represented by a grid in cyan color. The size of the window that has been already mentioned is 8×8 and, inside each window, the minimum is marked with a red point. On the left side of the figure, the entire image is presented; a blue square indicates the zoomed-in area that is presented on the right side of the figure.

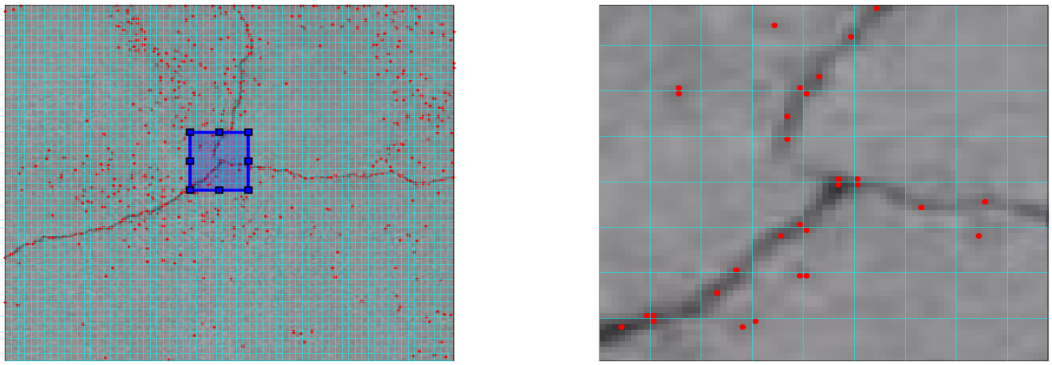

Figure 10. Local minima original. **Left:** full image. **Right:** zoomed-in.

The adjacent local minima, or repulsion of nodes, is a minimal distance defined by the user in which the nodes cannot lay next to each other. To introduce this concept, Figure 11 is presented. The figure shows a synthetic gray image with all its values equal to 0.5. A series

of values lower than 0.5 are presented with different colors, red, green, yellow, and blue, for the corresponding quadrants I, II, IV, and V. At the same time, each color has its own range of values from $0 < I_{color} < 0.5$, with the darkest being the lowest ones. On the right side of the figure, the local minima obtained with the original method are presented. As can be appreciated, the red and the dark green points end up lying next to each other.

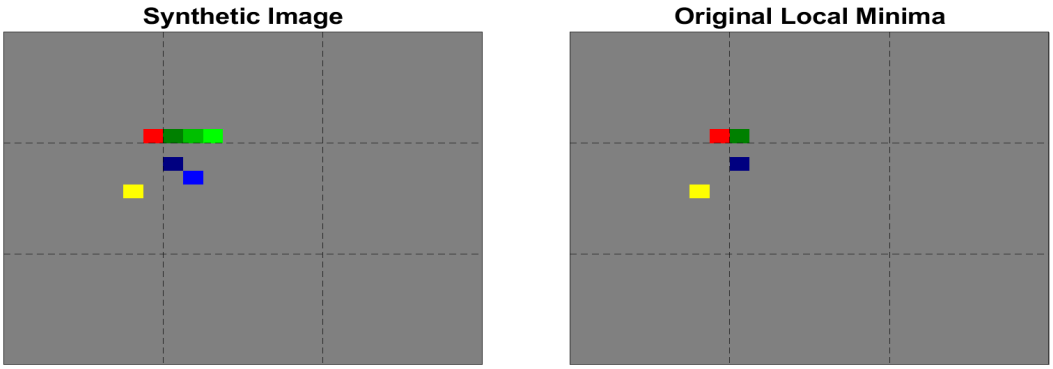

Figure 11. Original local minima. **Left:** synthetic image. **Right:** local minima.

Figure 12 presents the result obtained with the repulsion modification. The center image shows a minimal distance between adjacent points of 1 pixel, while the right image shows a minimal distance of 2 pixels.

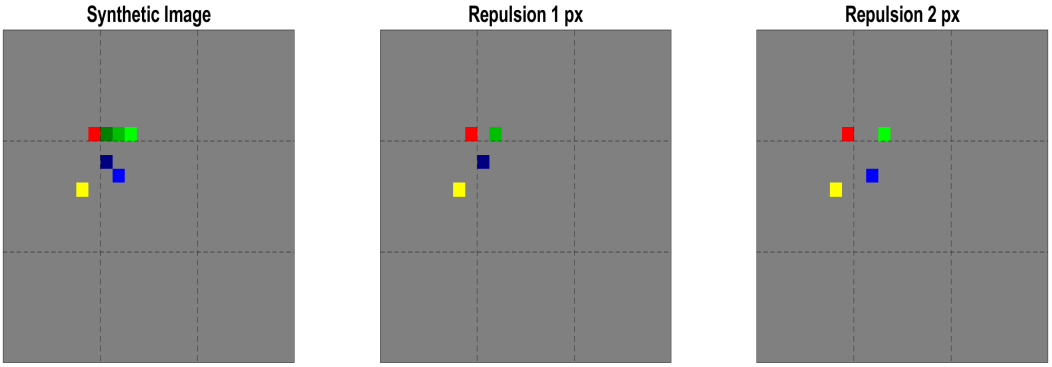

Figure 12. Local minima with repulsion. **Left:** synthetic image. **Center:** $d_{min} = 1$. **Right:** $d_{min} = 2$.

In the present work, a minimal distance of $d_{min} = 2px$ is adopted. Figure 13 compares the same image with the two methods. As can be appreciated, the repulsion method promotes a better alignment between points, as well as better coverage along the crack.

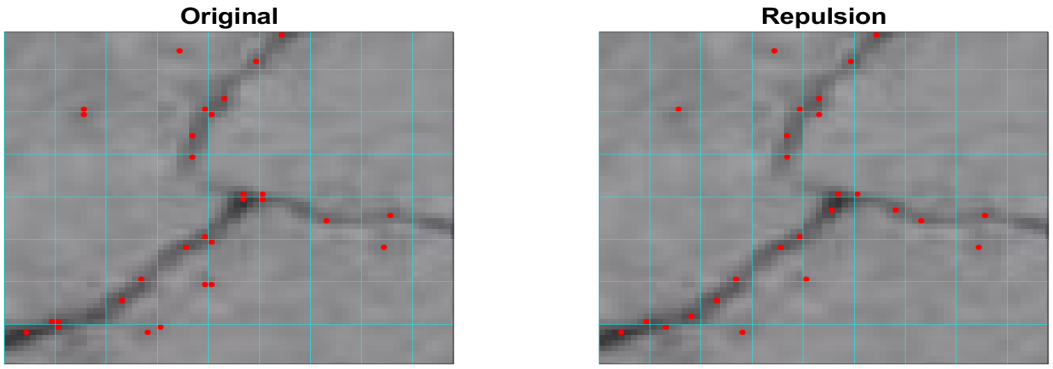

Figure 13. Local minima with repulsion. Zoomed-in. **Left:** original local minima. **Right:** repulsion local minima.

3.3.2. Region-Based Local Minima

The second part of this subsection is the region-based local minima. Using the extracted Gabor features, a segmentation is obtained in order to separate two areas based on their textural content. The local minima are found separately for each region. For the first region, the global local minima are calculated from the entire feature image using a global threshold $T_1 = \mu_{GF} - 1.5\sigma_{GF}$, where μ_{GF} is the mean of Gabor features and σ_{GF} its standard deviation, then only the points falling within the region with the lowest values from the Gabor features (Region 1) are maintained. For the second region, the flat-fielded image is used to determine the local minima; in this case, a regional threshold T_2 is obtained considering the values from Region 2 and is equal to $T_2 = \mu_{FF\cap R2} - 2.5\sigma_{FF\cap R2}$, where $\mu_{FF\cap R2}$ is the mean of the intensity values from the flat-fielded image only in Region 2 and $\sigma_{FF\cap R2}$ its standard deviation. Finally, the union of these two points yields the local minima. Figure 14 shows the decisional flow chart used in this step.

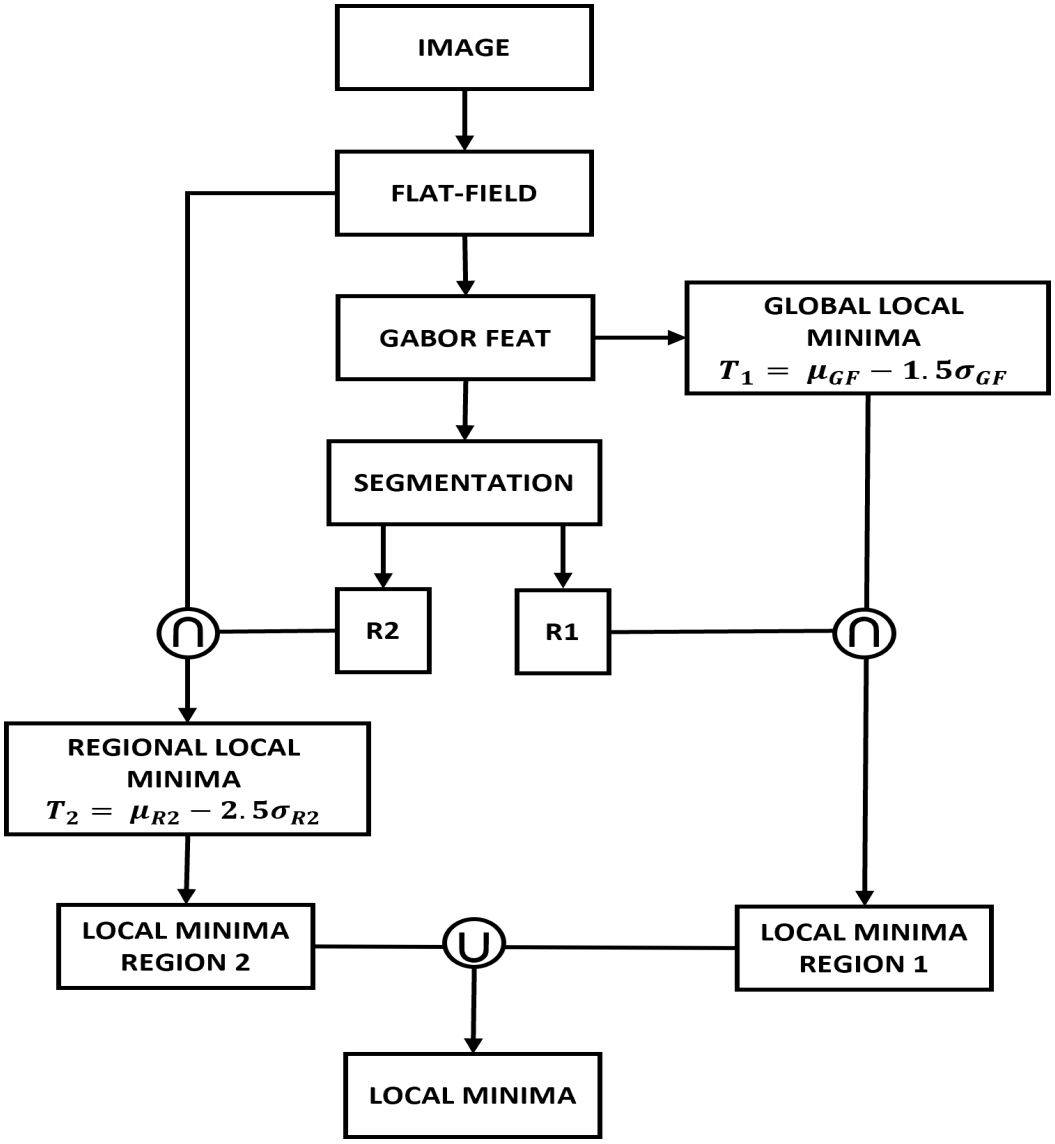

Figure 14. Decisional local minima flow chart.

In the following Figure 15, we present the image used to illustrate the procedure. On the left side, the flat-fielded image is presented, while on the right side, the extracted Gabor features are presented. The Gabor features emphasize the area where the biggest crack is present, but it loses the definition of the finer details.

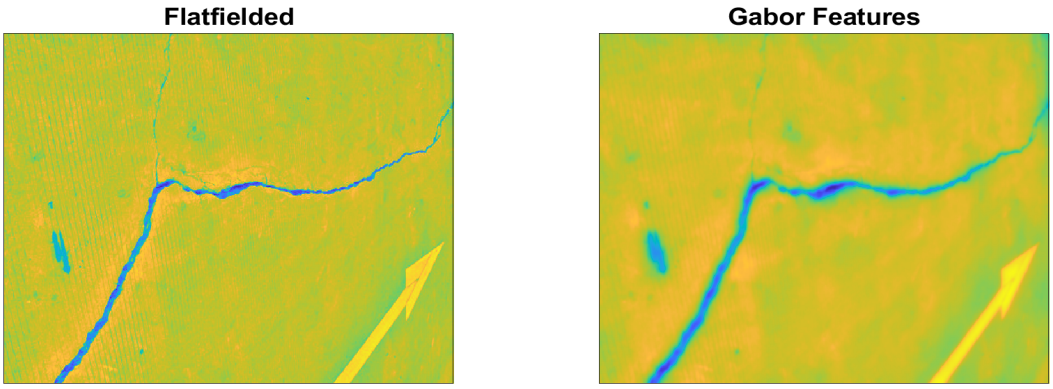

Figure 15. Example. **Left:** flat-fielded image. **Right:** Gabor features.

Based on this fact, the image is segmented into two areas from the textural features, as shown in Figure 16. To perform this, a K-means clustering-based algorithm for image segmentation was performed on the 36D image with all the magnitude responses concatenated in depth as well as the spatial coordinates. K was chosen equal to 3, then the lowest region was defined as Region 1 and the other two as Region 2. It can be appreciated on the right side of the image that the yellow label "Region 1" represents the potential position of the biggest cracks and the violet label "Region 2" represents the background of the scene.

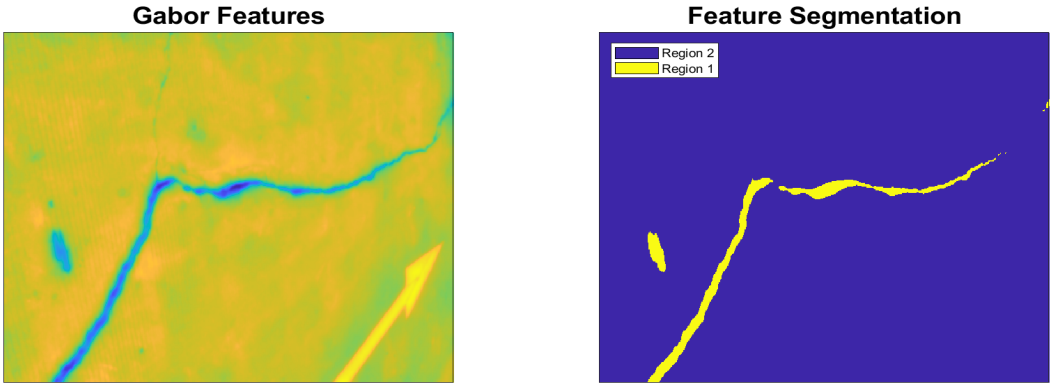

Figure 16. Region segmentation. **Left:** Gabor features. **Right:** segmentation.

Figure 17 shows, separately, the obtained points for each region. On the left side of the figure, the Gabor features extracted alongside the obtained points for Region 1 show the concentration of points falling within the big crack. On the right side, the flat-fielded image alongside the points found for Region 2 represent the background and therefore have a bigger constant equal to 2.5, compared with the 1.5 used for Region 1 in the feature image.

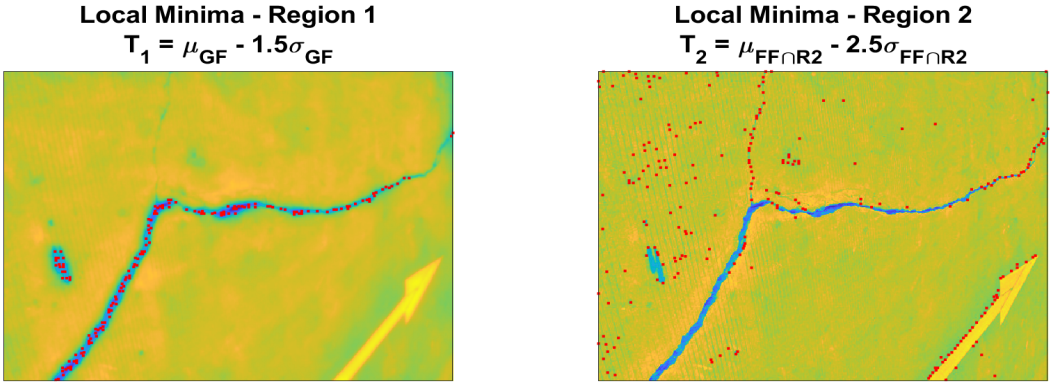

Figure 17. Regional local minima. **Left:** Region 1. **Right:** Region 2.

Subsequently, Figure 18 is presented, displaying a comparison between the original method and the proposed modification. As can be noticed, the proposed new method allows separating the identified local minima into two categories; on the right side of the image, the points coming from Region 1 are marked in red, while the ones coming from Region 2 are in green. Furthermore, the use of different constants for each threshold value gives more flexibility to the user who can adjust them depending on the particular case.

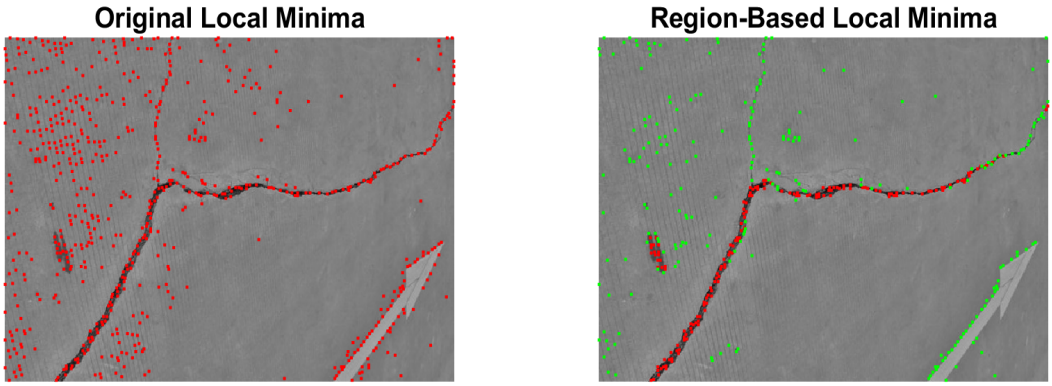

Figure 18. Local minima original vs. region-based. **Left:** original. **Right:** region-based.

To highlight the importance of the flat-field correction when extracting the features, which then are used to segment the regions, the example from Figure 9 is recalled and displayed alongside the segmentation for each case in Figure 19. As can be noticed, on the right side of the image, the segmentation after correcting the brightness guides the segmentation to consider only the major crack.

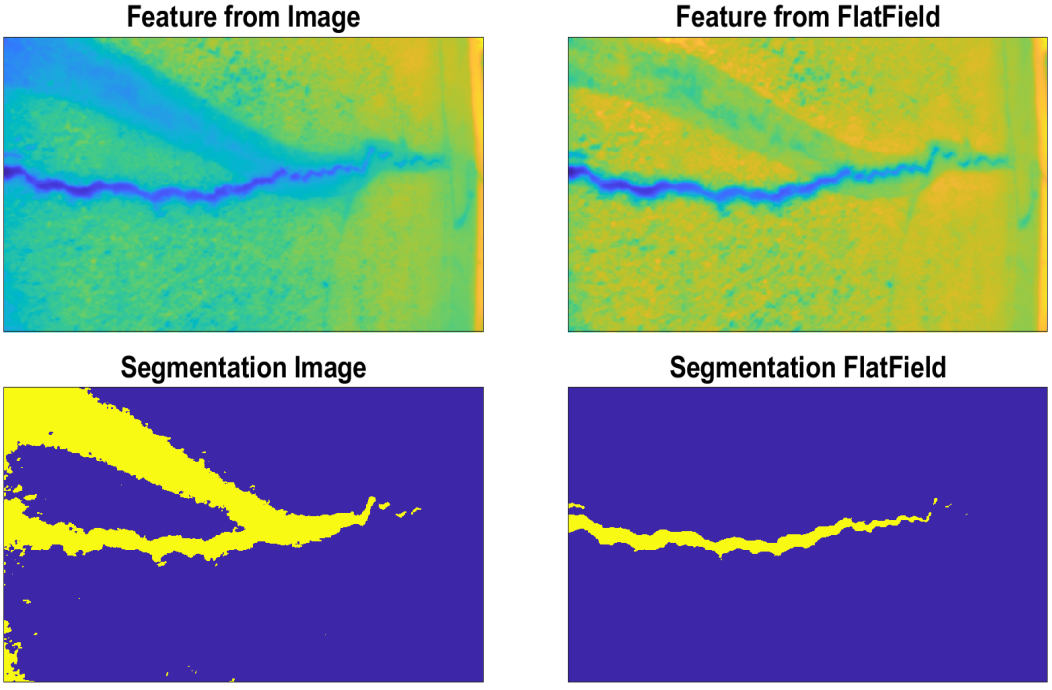

Figure 19. Influence from flat-field on the segmentation.

Finally, Figure 20 is presented below; in this case, a new example is shown to expose the impact of the proposed methodology regarding the total number of points. As can be noticed from the image, the total amount of points is reduced almost two times compared to the original methodology, which mostly lies in the background area. As stated in the introduction of Section 3.3, the total number of points has a huge impact in terms of computational time when calculating the minimal paths between nodes.

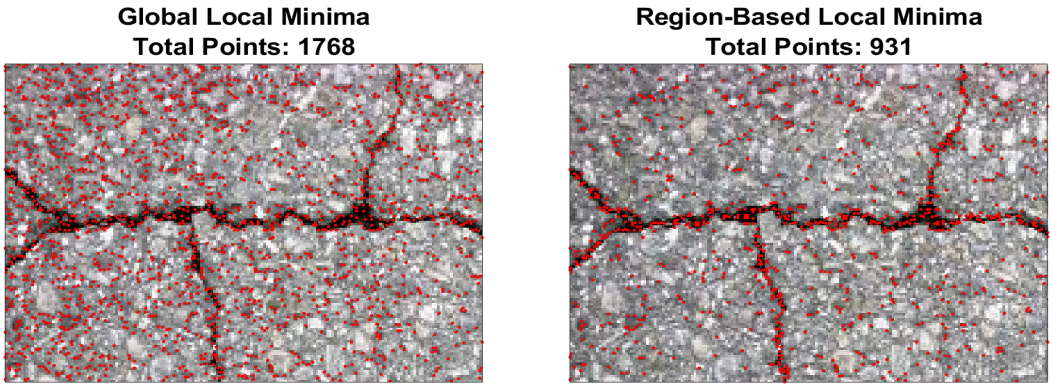

Figure 20. Amount of local minima points. **Left:** global. Total points: 1768. **Right:** region-based. Total points: 931.

3.4. Minimal Paths

The concept behind the minimal path selection is based on the theory of the Dijkstra algorithm already described in Section 2.1. Using the fundamental principles of graph theory, the minimal path between two nodes is found by defining a cost function. As already stated, cracks present a common property, i.e., they are usually formed in a shape of a line or a set of gridded lines that break the homogeneity of the surface. These breaks, having a fast rate of change in the slope of the surface, generate an obstacle for the incoming light which is seen by the human eye as a dark area. In other words, the cracks are generally captured in the images as the darkest areas and they follow a random path. Therefore, in the context of crack detection, the cost function to be minimized was defined, taking into account these considerations. The cost function was defined as follows:

$$c(p_{ij}) = \sum_{m=i}^{j} I(m) \tag{8}$$

where $c(p_{ij})$ is the cost of the path from i to j, x_i is the source point, x_j is the destination point, and m is a pixel within the path; using this cost does not constrain the shape nor the length of the selected paths.

Figure 21 shows the image used to present the modifications in this section. On the right side, a zoomed-in portion is presented with the associated local minima. As can be appreciated, 6 local minima are present in the example.

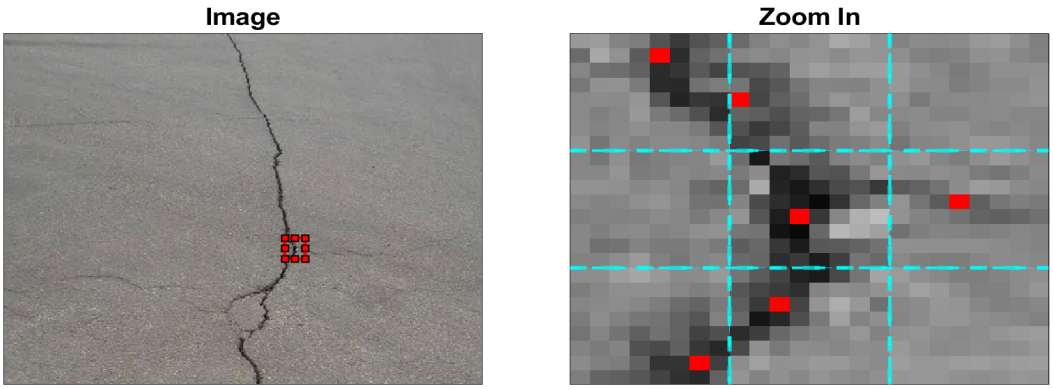

Figure 21. Local minima. **Left:** Full size image. **Right:** Zoomed-in portion.

An important aspect that is not clearly explained in [42,43] is the fact that some paths are overlapped, leading to a biased accumulation of the costs. In order to elucidate this issue, we analyze the total number of combinations. This number is obtained by calculating

the number of possible combinations for each node to all the others. This is performed using Equation (9).

$$ {}_{n}C_{k} = \frac{n!}{(n-k)!k!} \tag{9}$$

where k is the number of combinations and n is the number of nodes. In this particular case, $n = 6$ and $k = 2$, since the nodes can only be set as start or end, and the total number of possible combinations $C = 15$.

Figure 22 presents the same zoomed-in portion. On the left side of the image, the local minima can be appreciated, while on the right side, is the obtained minimal path.

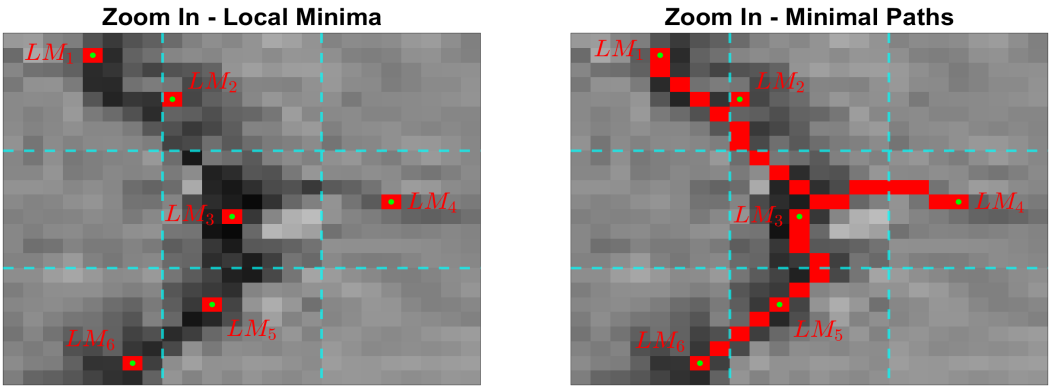

Figure 22. Local minima. **Left:** zoomed-in local minima. **Right:** zoomed-in minimal paths.

Figure 23 presents the number of times a pixel is traversed along the path. As can be appreciated, all the pixels within the paths are at least crossed $n - 1$ times, and the central is crossed 11 times. It is also important to notice that the window we are analyzing at this moment will be shifted; therefore, again, some paths will be considered twice. A solution to these problems is introduced in the next subsection.

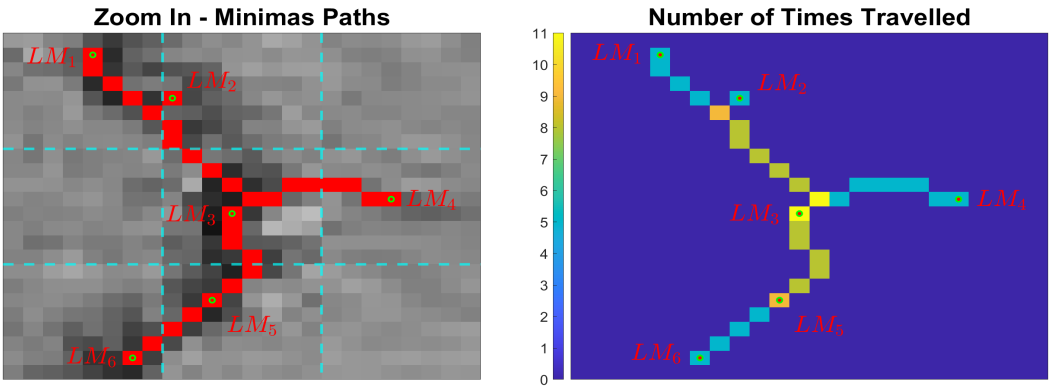

Figure 23. Zoomed-in minimal paths. **Left:** minimal paths. **Right:** number of times traveled.

3.5. Acceptance-Rejection of Paths

The next step in the algorithm foresees the acceptance or the rejection of the paths that satisfy a certain condition based on a defined limit. This threshold is estimated on the statistics of the so-far obtained paths. At this step, the goal is to select paths with the lowest mean intensities and not the shortest ones; therefore, the threshold is estimated with a normalized version of the cost by simply dividing the previous cost in Equation (8) by the length of the path:

$$ c(p_{ij}) = \frac{1}{len(p_{ij})} \sum_{m=i}^{j} I(m) \tag{10}$$

where $len(p_{ij})$ is the length of the path in pixels. Then, the threshold is calculated as follows:

$$T_C = \mu_c - k\sigma_c \tag{11}$$

where μ_c is the mean of the costs, σ_c is their standard deviation, and $k = 1$, which was determined empirically. Figure 24 shows the results obtained with the original method. On the left side of the figure, the image with the paths found is presented, while on the right side, the paths that satisfied the condition are displayed.

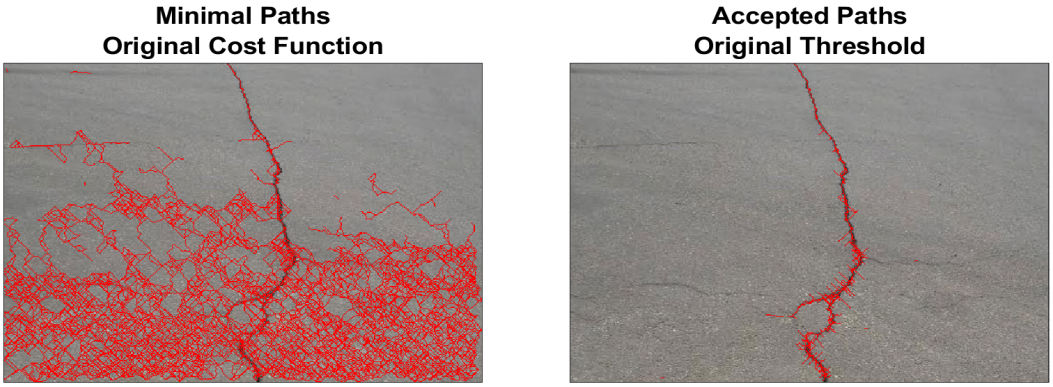

Figure 24. Rejection original. **Left:** all minimal paths. **Right:** accepted minimal paths.

As anticipated in the previous subsection, a specific issue was identified in the procedure since some paths cross the same pixel several times. This fact influences the calculation of this threshold since the estimate is biased towards the deepest cracks that are traveled several times. Therefore, the threshold is overestimated while the found paths are underestimated. Two potential solutions addressing this issue are proposed. First, since the statistics of the cost function follow roughly a bimodal distribution, calculating the T_c with Equation (11) shifts the value towards the darkest points in the image, and the solution is foreseen to determine the threshold based on a Gaussian fit on the upper mode.

Figure 25 shows a comparison between the threshold determined over the statistics of the Costs and the proposed Gaussian fit threshold. As can be noticed by the vertical dashed lines, the threshold in the original methodology is lower than the one proposed.

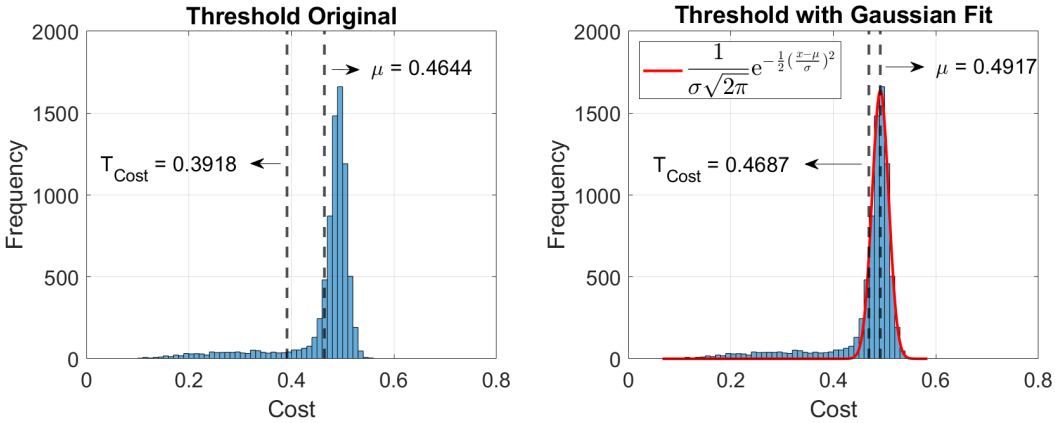

Figure 25. Histograms. **Left:** threshold original. **Right:** threshold with Gaussian fit.

In the following Figure 26, results obtained with the original threshold and with the first solution are introduced. It can be noticed how this threshold covers more secondary cracks that in the first case are omitted.

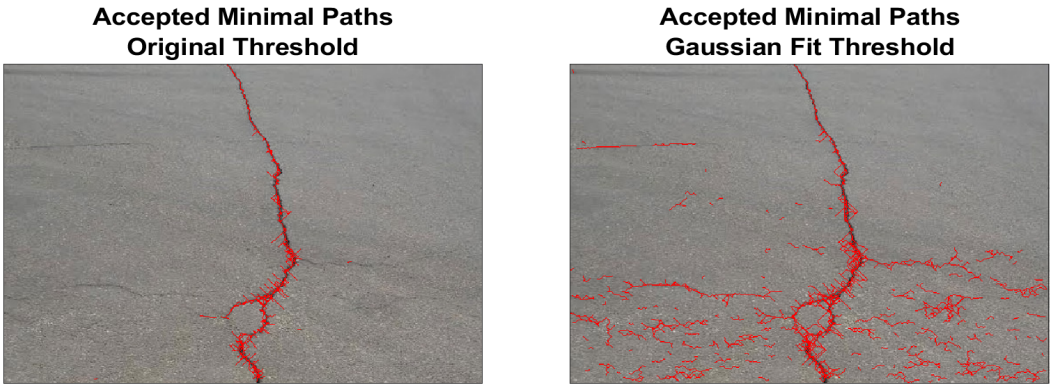

Figure 26. Accepted paths. **Left:** threshold original. **Right:** threshold with Gaussian fit.

The second solution proposes that the threshold can be obtained directly by considering the mean and standard deviation of the intersection between the image and the so-far detected paths, instead of calculating the cost for each path and estimating the threshold from the population of costs. In other words, it considers only the intensity levels of the detected paths. The threshold is named global threshold and is determined as follows:

$$T_{Global} = \mu_{I \cap Paths} - k\sigma_{I \cap Paths} \tag{12}$$

where $\mu_{I \cap Paths}$ is the mean of the intensity levels of the so-far identified paths, σ_c is their standard deviation, and $k = 1$ remains the same as before. Figure 27 shows the histograms for both cases.

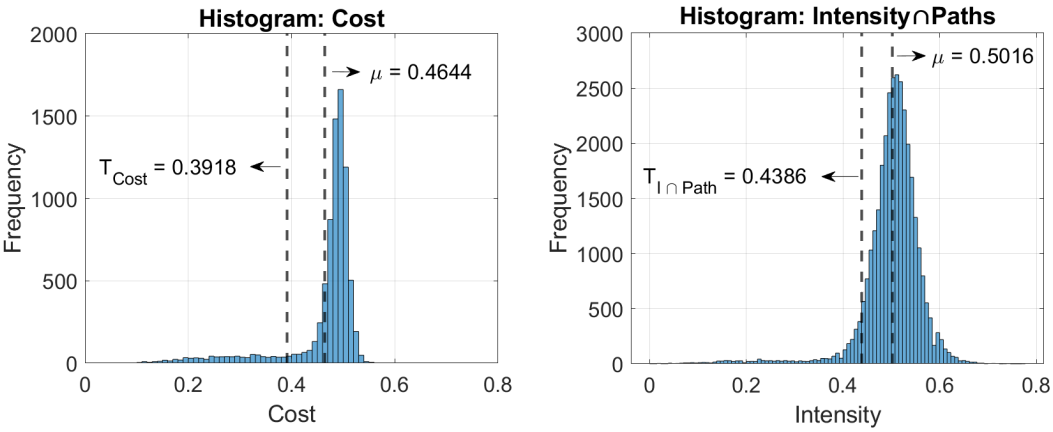

Figure 27. Histograms. **Left:** costs. **Right:** intensities.

On the left side of the figure, the histogram of the costs is presented, while on the right side is the histogram of the intensities. It can be noticed in the latter how the histogram can be better fitted by a normal distribution. Indeed, this solution is the equivalent of considering the paths as traveled only once, since Equation (10) is no other than the mean intensity of the path.

Figure 28 shows a comparison between the paths obtained using the original cost threshold from Equation (11) vs. the global threshold from Equation (12). As can be noticed, the image from the right, corresponding to the global threshold method, yields results in between the original threshold method and the Gaussian fit method.

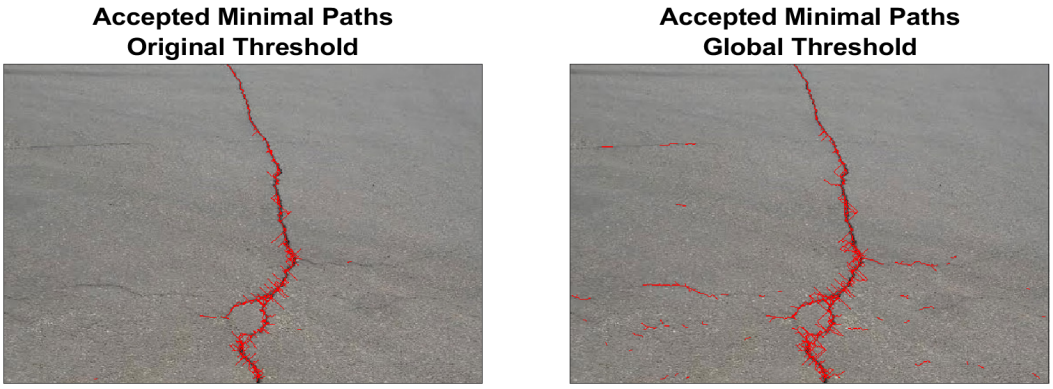

Figure 28. Accepted paths. **Left:** threshold original. **Right:** threshold with global.

In the present new version of this algorithm, the global threshold is used following the methodology proposed in Section 3.3.2 for local minima. The region-based concept is also extrapolated for the calculation of the thresholds. Figure 29 presents the image and the identified regions.

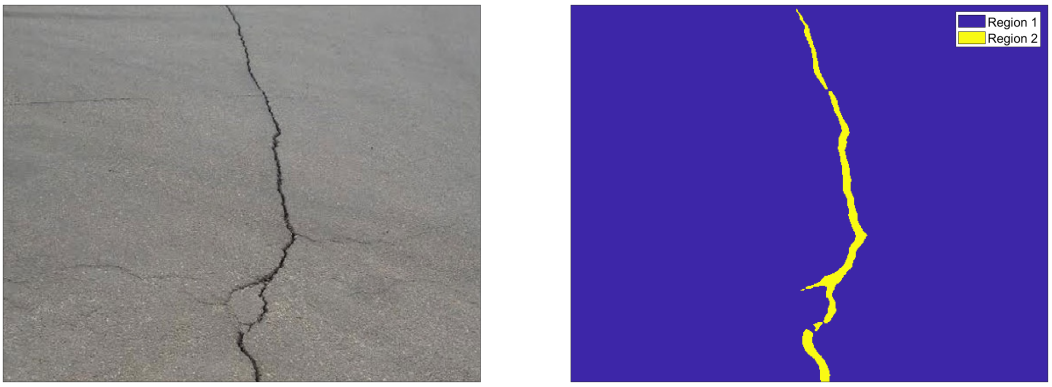

Figure 29. Regions. **Left:** image. **Right:** regions.

For each region found in Section 3.3.2, a different threshold is calculated. The calculation of the thresholds follows the notions introduced in the previous section. This allows better control of the thresholds in the different zones. Particularly, the zone identified with the lowest feature values typically matches the biggest crack of the region. Therefore, the threshold for the paths falling within this region considers the addition of the standard deviation rather than the subtraction of it.

$$\begin{cases} T_1 = \mu_{I_1 \cap Paths_1} - k_1 \sigma_{I_2 \cap Paths_2} \\ T_2 = \mu_{I_2 \cap Paths_2} + k_2 \sigma_{I_2 \cap Paths_2} \end{cases} \tag{13}$$

where $k_1 = 1$ and $k_2 = 0.6$. Figure 30 presents the histograms for both cases. In the figure, the Y-axes are plotted in a logarithmic scale. In the figure, the upper histogram represents the methodology presented in Section 3.5, obtaining the threshold with Equation (11), while the lower graph presents two histograms, one for each region, and the determined thresholds for the particular example presented.

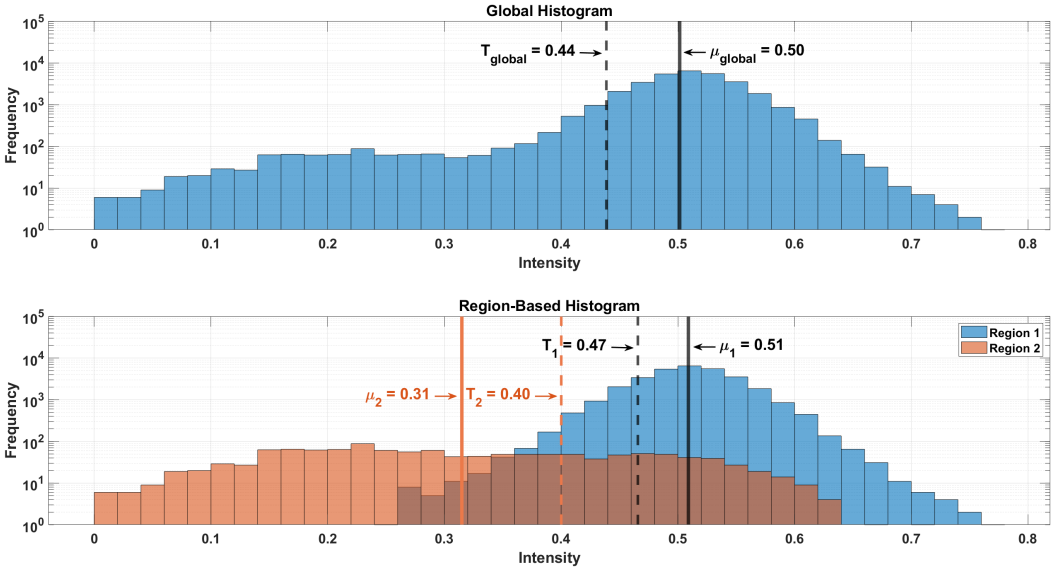

Figure 30. Regions histogram. **Up:** global. **Bottom:** region-based.

As can be noticed, the upper threshold is slightly higher than the global threshold, i.e., $T_{global} \leq T_1$; this allows the detecting of a higher amount of cracks in the foreground. Furthermore, the lower threshold is slightly lower than the global one, i.e., $T_{global} \geq T_2$; this reduction helps to filter some of the paths falling within this area. Figure 31 shows the comparison of the paths found using the original, the global, and the region-based thresholds.

Figure 31. Thresholds. **Left:** original. **Center:** global. **Right:** region-based.

Analyzing the results, it can be appreciated how the proposed improvements yield a higher amount of true positives in the background area compared with the original methodology. Nevertheless, a higher amount of false positives is also present. It seems to indicate that the global methodology yields better results but, after this step, the original algorithm foresees a global skeletonization procedure and a crack length filtering. Therefore, short cracks are disfavored and the critical point in this algorithm obtains connected cracks.

3.6. Skeletonization and Region Growth

This two steps are foreseen in the original algorithm; this was introduced in Section 2.3 as steps 5 and 6, respectively. In this work, these two steps were kept unchanged. For the skeletonization step, two types of nodes are identified: *branch points* and *end points*. To determine them, an internal function from Matlab®, called `bwmorph`, is used, both to skeletonize the paths and to find the points of interest.

The region growth procedure consists of iteratively absorbing dark pixels neighboring the currently detected crack, according to a threshold test. For further details, the reader is referred to [43].

3.7. Geometric Threshold

As a final step, a new filtering procedure on the found regions is proposed. Using an internal function from Matlab®, called bwconncomp, two properties are analyzed for each connected region: area and eccentricity. While the definition of the first is redundant, the latter is a property measured on an equivalent ellipse that has the same second moments as the identified region. The elongation of that ellipse is measured by its eccentricity e, a number ranging from 0 to 1. A value of $e = 0$ represents the limiting case of a circle, while a value of $e = 1$ represents the limiting case of infinite elongation (no longer an ellipse but a parabola), in our case, a line segment. The eccentricity is calculated as follows:

$$e = \sqrt{1 - \frac{b^2}{a^2}} \tag{14}$$

where a is the length of the semimajor axis and b is the length of the semiminor axis.

The combination of these two properties determines the regions to be filtered. The combination is shown in the following Table 1.

Table 1. Properties.

Category	Area	Eccentricity
I	10–25	0.975
II	25–50	0.950
III	50–250	0.900
IV	>250	0.850

4. Results

Figure 32 shows the results obtained for four different examples. For each image in the figure, the main cracks were presented in green, the secondary cracks in blue, and the filtered regions in red.

In all four images, the green cracks represent the main crack belonging to Region 2 defined in the previous sections. Cracks displayed in blue represent secondary cracks belonging to Region 1. In red, the small regions filtered in this last step can be appreciated. In A, the image presents a perspective view, the image presents thin cracks and several secondary cracks, and three small regions were filtered belonging to Category III from Table 1. In B, a wide crack is presented; as can be noticed, this type of crack yields results not as good compared with the other cases, and the detection of secondary cracks is not accurate. Moreover, the image present a watermark, which elevates the complexity of the scene; the images with watermarks were chosen to be left in the dataset since they represent a typical challenge that DL models solve quite well. In C, a crack with medium width and with a short secondary crack is presented, and several small regions belonging to Category I are rejected. Finally, in D, an image with a view from the top is presented; in the scene, a big oil stain can be appreciated together with a main thin crack: the big oil stain belongs to Category IV and is rejected since it does not fulfill the eccentricity criteria.

A and D present a particular situation regarding the blue areas, i.e., the secondary cracks. It can be noticed that some small parts of the blue areas indeed belong to the primary or main crack. In order to adjust this, blue regions are analyzed as to whether they fall within a green crack or not. In the case that they fall within a green crack, they are considered as if they were part of it and they are absorbed. On the other hand, blue regions having free endpoints are retained as secondary cracks together with disjointed areas. Figure 33 highlights the incorporation of blue segments to the green ones. In addition, some secondary cracks growing from the main crack are retained and some other unconnected areas are present.

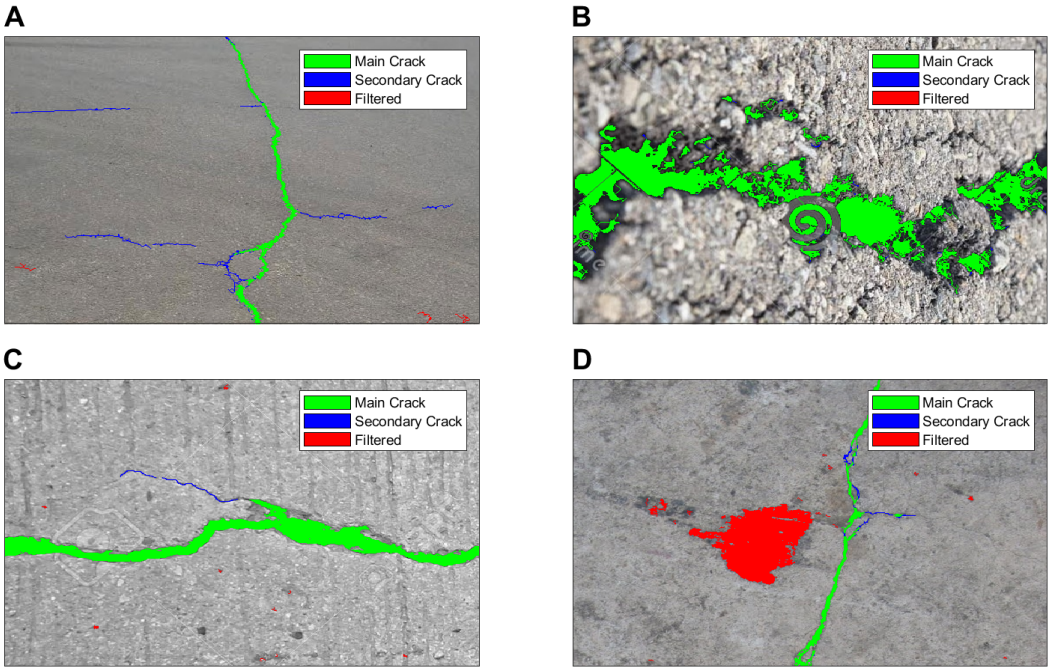

Figure 32. Results. (**A**) Thin cracks in perspective and secondary cracks. (**B**) Wide crack and watermark. (**C**) Medium width crack with secondary cracks. (**D**) View from the top, thin cracks, secondary cracks, and oil stain.

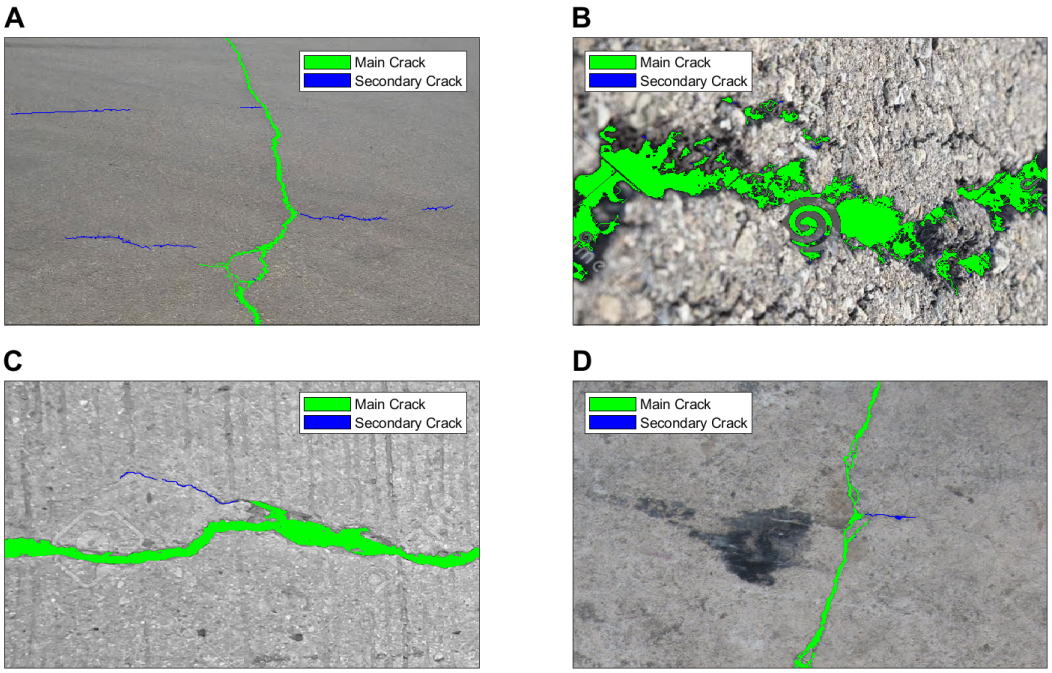

Figure 33. Filtered results. (**A**) Thin cracks in perspective and secondary cracks. (**B**) Wide crack and watermark. (**C**) Medium width crack with secondary cracks. (**D**) View from the top, thin cracks, secondary cracks, and oil stain.

Finally, Figure 34 shows the comparison between the results obtained with our method RB-MPS and the original MPS algorithm. The presented examples are displayed vertically and each method is represented on each column, respectively. Images A, C, E, and G present the results with RB-MPS while B, D, F, and H present the results obtained with MPS.

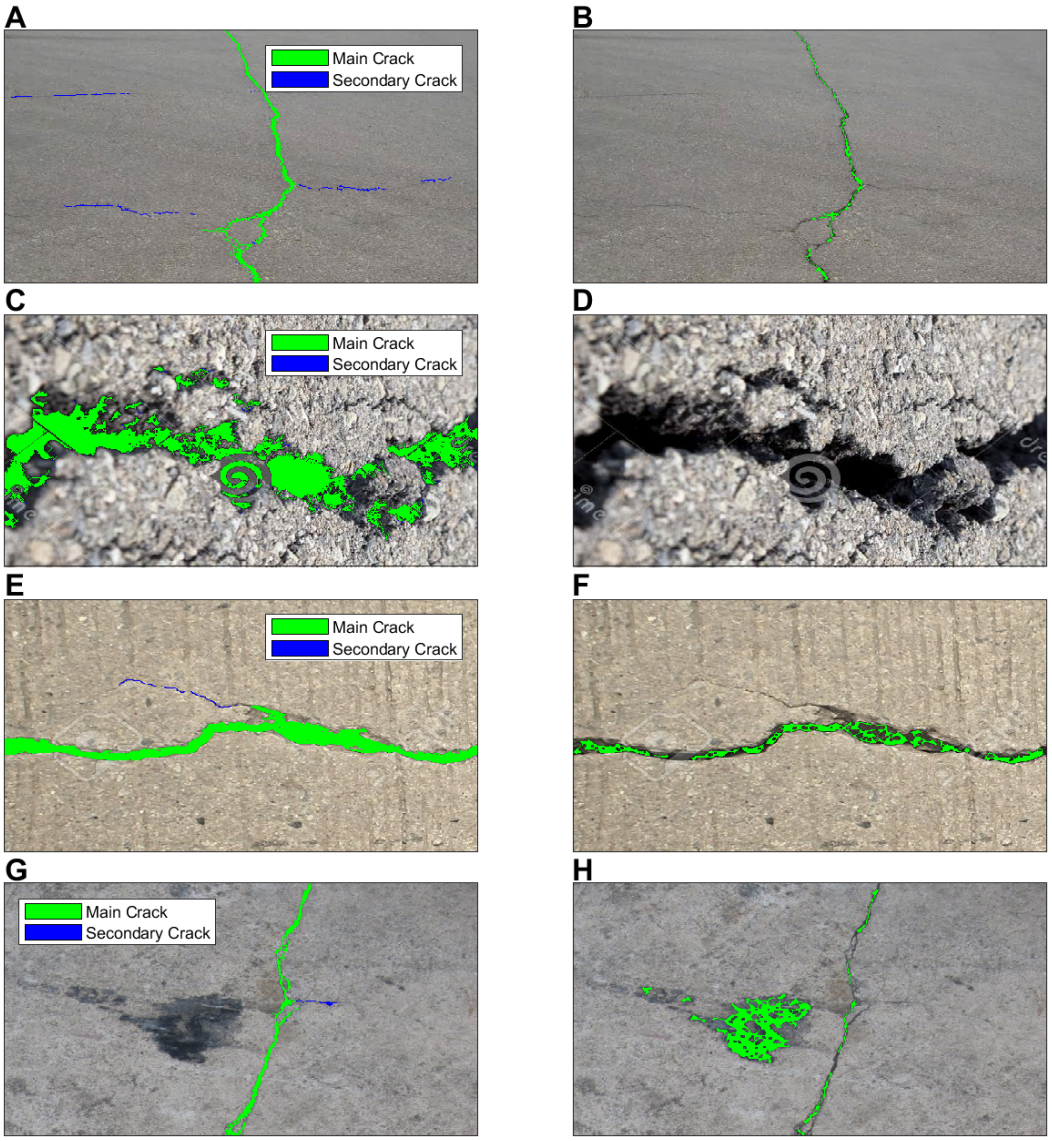

Figure 34. Results for RB-MPS (**A**,**C**,**E**,**G**) vs. MPS (**B**,**D**,**F**,**H**).

In the first row, it can be appreciated how the MPS algorithm has good accuracy regarding the main crack. However, the detected secondary cracks of A are omitted in B. The next row presents an example of a wide crack; as can be seen, the MPS methodology does not detect any crack. This is due to the distribution of pixel intensities since the present image contains a high number of dark pixels belonging to the crack. Therefore, at the moment to accept paths, none of them satisfy the condition and become rejected. On the other hand, the result obtained with RB-MPS is not optimal, but the region-based approach allows them to keep the paths and still finds a good amount of paths inside the crack. Continuing downwards, the following row presents images E and F. In this case, the detection with MPS is not quite accurate since some internal parts within the crack are missing, while the result with RB-MPS shows a really good performance. Finally, images G and H present the last example. In this case, the results in H show a big stain detected as a crack, while, as shown in Figure 33, the stain is filtered in the last step of the procedure.

In order to provide a quantitative comparison, a set of evaluation metrics are used to assess the performance of the results. These metrics measure the similarity between the predicted and the ground truth regions based on the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) generated by the model.

Precision, recall, and *F*1-score are commonly used metrics for binary classification tasks and are widely used in crack segmentation.

- **Precision:** Precision measures the proportion of true positives among the predicted positive samples. Precision is a useful metric when the cost of false positives is high, such as in medical diagnosis. It is calculated as follows:

$$Prec = \frac{TP}{TP + FP} \tag{15}$$

- **Recall:** Recall measures the proportion of true positives among the actual positive samples. Recall is a useful metric when the cost of false negatives is high, such as in fraud detection. It is calculated as follows:

$$Rec = \frac{TP}{TP + FN} \tag{16}$$

- ***F*1-Score:** *F*1 is the harmonic mean of precision and recall and is used to balance the trade-off between them. *F*1-score is a useful metric when both false positives and false negatives are equally important, and there is an imbalance in the class distribution. It is calculated as follows:

$$F1 = \frac{2 \cdot Prec \cdot Rec}{Prec + Rec} \tag{17}$$

A distance of 2 pixels between the detection and the reference segmentation for the calculation of the TP rate is accepted in the same manner as in [43].

Table 2 shows the mean of the metrics obtained at each step of the procedure for both methodologies. As can be noticed in every single step, our RB-MPS outperforms the MPS. It is important to note that the biggest differences can be seen in the first two steps of the procedure. Indeed, this is a combined consequence of the lower number of local minima and the region-based rejection strategy.

Table 2. Metrics RB-MPS vs. MPS.

	Precision		**Recall**		***F*1**	
	RB-MPS	**MPS**	**RB-MPS**	**MPS**	**RB-MPS**	**MPS**
Find Paths	0.3133	0.1346	0.9541	0.8785	0.4255	0.1505
Clean Paths	0.7019	0.4695	0.9105	0.7455	0.7626	0.3162
Skeleton	0.7692	0.5412	0.9472	0.8263	0.8100	0.4944
Final	0.8144	0.5694	0.9407	0.8102	0.8386	0.5536

Table 3 displays the results compared against the results presented in *DeepCrack*. It can be noticed how the deep learning approach yields better results in terms of precision, while our approach has a higher recall value. A high recall value means that the number of FN is quite low, while a higher value of precision translates into a lower number of FP. It can be noticed in the case of the DC approach how the recall and precision values are more balanced, while in our approach this difference is slightly more acute. The explanation behind this difference is mainly due to the incorporation of the secondary cracks in our approach, which yield a higher number of FP, impacting the precision rate value.

Table 3. Metrics MPS vs. RB-MPS vs. DC.

	Precision	**Recall**	**F1**
MPS [42,43]	0.5694	0.8102	0.5536
RB-MPS	0.8144	**0.9407**	0.8386
DC [50]	0.**8610**	0.8690	**0.8650**

In order to give a visual demonstration, Figure 35 is presented. The figure shows a comparison of the results obtained with the different methods. The first row presents the original images extracted from the dataset. Then, the second row named B, presents the ground truth data which were labeled by hand by experts. The third row, named C, shows the results obtained with the DeepCrack network. Row D presents the results obtained with the MPS algorithm. Finally, row E shows the results obtained with the RB-MPS. The results obtained with RB-MPS perform slightly worse in situations where the crack width is considerable, such as the first column case. On the other hand, the MPS algorithm completely fails to identify this type of crack. The second and third column examples show how the DC has a continuous estimation map, i.e., nonbinary, yielding an effect of degrading. Regarding MPS vs. RB-MPS, the results are quite similar, with a slightly higher rate of false detection coming from the MPS. Finally, the last example in the fourth column shows an exogenous flower in the middle of the crack. In this case, our algorithm outperforms DC and MPS. Further discussion can be found in the next section.

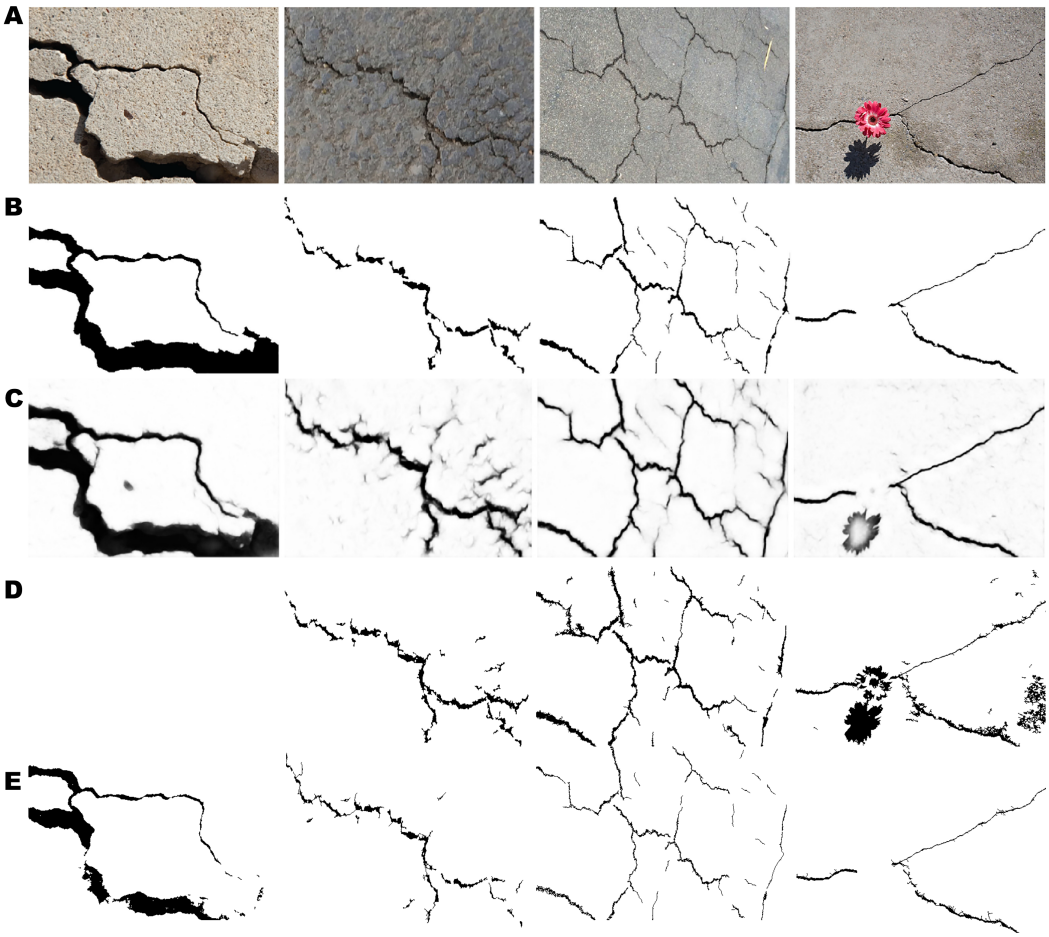

Figure 35. Results. (**A**) Original images. (**B**) GT. (**C**) DC. (**D**) MPS. (**E**) RB-MPS.

5. Discussion and Future Works

Even though the Results section showed a couple of examples of the final results, through Section 3, different steps from the proposed new version of this algorithm were introduced along with some results. Table 4 resumes the differences in the steps, comparing the original algorithm and the proposed one.

Table 4. Steps MPS vs. RB-MPS.

	MPS [42,43]	RB-MPS	Upgrade
Preprocessing	No Details	Flat-Field	New
Kernel	8×8	8×8	Same
Local Minima	Global	Adjacent Gabor-Region-Based	New
Paths Finding	Dijsktra	Dijsktra	Same
Rejection	Global	Region-Based	New
Skeletonization	Branch and Loops	Branch and Loops	Same
Width Grow	Absorption	Absorption	Same
Geometric Threshold	No	Yes	New

Starting from the top, the first row highlights how the original algorithm does not present any details regarding the preprocessing step used. In this study, a new flat-fielded methodology was proposed. The new *Gaussian PCA* was proven to improve the accuracy of the new algorithm through the workflow and in the final outcomes. This step is rarely made explicit or detailed; even with deep learning approaches, this preprocessing step is taken for granted, even though it has a big impact on the following steps.

Secondly, the kernel dimensions influence the computational time of the *Dijkstra* minimal path search, since the distance between nodes increases. Moreover, too-small window sizes can yield unconnected paths, which is undesired since small paths are removed. Therefore, the kernel size was left unchanged based on the results shown from [43].

Third, the local minima present two new improvements. First, the adjacent repulsion allows the local minima not to be placed side by side, and, thus, they better cover the crack length. This improves the path connections which otherwise are at risk of becoming detached. Secondly, the segmentation obtained by using Gabor filters allows two regions to be distinguished. These regions give rise to the new region-based approach. In this particular step, the new approach yields a lower number of local minima, i.e., nodes in the algorithm; therefore, this number has a huge impact on the computational time. The lower quantity of points is more visible in the background, meaning that the new approach keeps the points laying in the main crack while reducing the amount of them in the background zone.

Next, for the rejection/acceptance of the paths, an important problem was detected. Indeed, the original version of the algorithm overestimated the threshold since the statistic calculated over the cost considered paths traversing some pixels several times, biasing the threshold towards the darkest values; this resulted in some neglected secondary cracks. The region-based approach was incorporated to overcome these problems, and this aspect became one of the main advantages of our work since the algorithm is able to better identify secondary cracks. Furthermore, another big issue was identified in cases where the crack represented a high proportion of the scene, such as the wide cracks or close-up images. The problem was corrected by considering the global thresholding approach for each region; this is a key concept that allowed us to overcome these problems and better generalize the algorithm.

Concerning the minimal path finding algorithm, i.e., *Dijkstra*, the skeletonization and the width growth, no further modifications were introduced, and the same concepts described in the original algorithm hold. As a final step, the geometric threshold step was incorporated; this step is very important since the overestimated regions are filtered, such as in the case of rounded shape objects or very short thin cracks.

Finally, the results showed close to state-of-the-art results using deep learning approaches. From a quantitative point of view, the metrics from the DC showed a slightly higher $F1$-score, yielding more balanced precision and recall metrics. In the case of MPS, the final $F1$-score is quite low; this is mainly explained because the algorithm was developed for thin cracks captured with a zenithal camera. Therefore, in this case, where the dataset was complex and had a lot of variance, several cases, such as the one presented in

the first column in Figure 35, failed to detect any crack. On the other hand, our RB-MPS algorithm was developed to overcome this problem and to better generalize the results. This is clear from the metrics results; however, the in-balance that can be seen in between the precision and recall indicates that the algorithm tends to yield a higher value of FN than FP. Particularly, a higher number of FPs influence the precision, yielding lower values, but as already stated, the ground truth labeling often neglects the secondary cracks, therefore our RB-MPS algorithm yields lower values of precision compared with the DC. This problem is recurrent in other crack databases since the labeling procedure is carried out manually with a high subjectivity from the expert. The authors believe that this issue must be deepened and studied in order to obtain a more standardized procedure for crack labeling.

A clear visual demonstration is presented in Figure 35, and the results obtained with RB-MPS perform slightly worse in situation where the crack width is considerable, such as in the first column case. The reason behind this behavior is the kernel size and the region-growth procedure. This problem open the possibilities for future work and can be easily tackled using a multilevel tackled scheme to estimate cracks at different scales. Following the discussion of Figure 35, the results presented for the DC showed a continuous estimation map of the crack; this can be explained due to the choice of the weighted cross-entropy loss, in terms of probability, that combines the outputs at each side with the final fused estimation. This concept is interesting and could be exploited in order to fuse both approaches, opening the field to interesting new results.

On the other hand, still referring to Figure 35, it was interesting to notice how the new addition, regarding the geometric threshold to determine the regions to be filtered, allowed the exclusion of exogenous elements in the images, such as the flower in the corresponding figure, or the oil stain in Figure 32. This is an excellent improvement, but must be calibrated carefully since it is scale-dependent. The values presented in this work were calibrated to yield the best results with the current dataset.

The main downside of the current algorithm is the computational speed. Indeed, the current algorithm speed is dependent on the total number of local minima found. On average for the dataset, the total time for each image sized 544×384 is about 65 s using the suggested parameters. In addition, in the case of not-satisfactory results, the calibration for the optimal parameters requires the user to supervise the algorithm for each instance. In the case of a large amount of data, an advised workflow is the following: run the entire dataset with the suggested parameters, check for unsatisfactory results, and adjust the parameters in order to obtain the desired results.

Here, the typical discussion between traditional and deep learning approaches opens up once again. Indeed, both methodologies have their advantages and drawbacks. For this particular application, DL models yield more balanced results and with higher metrics values ($F1$). However, they are usually solved using supervised learning, meaning that a lot of labeled data are needed, which is a tedious, prone-to-error, and time-consuming task. Furthermore, the "black-box" nature of the DL makes them indecipherable and impossible to correct when the results are erroneous. Conversely, the proposed "traditional" approach yields slightly lower metric values for the particular dataset. Nevertheless, these metrics could be biased since the current algorithm does not consider secondary cracks; hence, the values are lower. However, the main advantage of the proposed methodology is its transparency and its parametric nature, meaning that it could be adapted for each particular case and corrected in case the results are not satisfactory.

6. Conclusions

Cracks are fractures or breaks that occur in materials such as concrete, metals, rocks, and other solids. Cracks have an intrinsic randomness that makes them unpredictable and hard to model and assess. This aspect makes image-based methodologies very suitable and appropriate for the task of automatic detection systems. Although machine and deep learning algorithms have demonstrated their potential in the last decade, their "black-box" nature remains a gap to be filled in order to clearly explain the underlying principles of

the phenomena under investigation or to assess counterintuitive outcomes. At the cost of losing computational efficiency, traditional image processing can track intermediate processes and can serve as a valuable tool for crack segmentation.

The present study proposed a framework for a new algorithm for crack segmentation based on the theory of minimal path selection combined with a region-based approach obtained through the segmentation of texture features extracted using Gabor filters. Starting from the concept introduced in [43], several modification were proposed in order to generalize the algorithm. Firstly, a preprocessing step was incorporated in order to homogenize the scene in terms of light and shadows. This preprocessing step improved the false rate detection of local minima, yielding a lower amount of points, which translates into lower computational burden at the time to calculate the path between nodes. Moreover, a slight modification in the local minima identification forced adjacent points to respect a minima distance, breaking the local accumulation of points in certain areas, thus homogenizing the distribution of detected points in the scene. Afterward, a region-based segmentation technique was introduced to determine two regions in the image: one with the potential position of the crack, and the second region considering the background. This region-based approach was a key improvement that enhanced the detection of secondary cracks and improved the rejection of false paths and the generalization of the algorithm, particularly for close-up scenes or wide cracks. Finally, a thresholding step based on geometrical properties was presented, allowing for excluding rounded areas and small isolated cracks.

The main advantage of this study is the transparency of the workflow, contrary to what happens with deep learning approaches. This transparency allows understanding and correcting outcomes in cases where the algorithm does not work correctly. Moreover, in order to use the deep learning frameworks, a huge amount of data is needed and each instance needs to be labeled by hand. In the proposed approach, prior information is not required. However, the statistical parameters may have to be adjusted to the particular case and requirements of the situation. The proposed algorithm results in a useful tool for researchers and practitioners needing to validate their results against some reference or needing labeled data for their models. Moreover, the current study could establish the grounds to standardize the procedure for crack segmentation with a lower human bias and to speed up the procedure. The direct application of the methodology to images obtained with any low-cost sensor makes the proposed algorithm an operational support tool for authorities needing crack detection systems in order to monitor and evaluate the current state of infrastructures such as roads, tunnels, or bridges.

Author Contributions: Conceptualization, G.d.L., N.F. and P.L.; methodology, G.d.L.; software, G.d.L.; validation, G.d.L.; formal analysis, G.d.L.; investigation, G.d.L.; resources, G.d.L., P.L. and M.L.; data curation, G.d.L.; writing—original draft preparation, G.d.L.; writing—review and editing, G.d.L. and N.F.; visualization, G.d.L.; supervision, N.F., P.L. and M.L.; project administration, P.L. and M.L.; funding acquisition, P.L. and M.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the PRIN (Projects of Relevant National Interest) n. 20179BP4SM promoted by the Italian Ministry of University and Research (MUR).

Data Availability Statement: The dataset used in this work comes from [50]. The dataset containing the images and the labeled ground truth can be found in https://github.com/yhlleo/DeepCrack/blob/master/dataset/DeepCrack.zip.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

DL	Deep Learning
MPS	Minimal Path Searching
LRIS	Laser Road Imaging System
DC	DeepCrack
RB-MPS	Region-Based Minimal Path Searching
PCA	Principal Component Analysis
RBLM	Region-Based Local Minima
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

Appendix A

Algorithm A1 Dijkstra algorithm.

Input: *graph, source, target, costfun*
Output: *path*

for each *node* in *graph* **do** ▷ *initialization*
    $node.cost \leftarrow Inf$
    $node.visited \leftarrow false$ ▷ *mark as unvisited*
$source.cost \leftarrow 0$
$Q \leftarrow all(graph.nodes)$ ▷ *nodes to be checked*

while Q **is not** empty **do**
    $u.cost \leftarrow min(costfun(Q))$ ▷ *node in Q with smallest cost*
    $u.visited = true$ ▷ *mark as visited*
    remove u *from* Q
    for each neighbor v of u **do** ▷ *check current node neighbors*
        if $v.visited == false$ **then** ▷ *it has not already been visited*
            $newCost \leftarrow u.cost + costfun(u, v)$ ▷ *calculate cost*
            if $newCost < v.cost$ **then**
                $v.cost \leftarrow newCost$
                $path \leftarrow u$ ▷ *add the current node to the path*
    if $u ==$ *target* **then** ▷ *current node reached the target*
        $path.cost = costfun(path)$ ▷ *get the path cost*
        $Q \leftarrow empty$ ▷ *empty Q to end while*
    if $costfun(Q) == Inf$ **then** ▷ *remaining nodes are unreachable*
        $Q \leftarrow empty$ ▷ *empty Q to end while*
        $path \leftarrow empty$ ▷ *path not found*
        return

return *path*

Algorithm A2 Minimal paths for cracks.

Input: I, K
Output: *Cracks*

$T_I \leftarrow \mu_I - \sigma_I$ ▷ *Intensity Threshold*
function LOCALMINIMA(K, I, T_I)
    for each K in I **do** ▷ *Convolve Image*
        $A_k \leftarrow min(I_k)$ ▷ *Local Minima*
        if $A_k \leq T_I$ **then**
            keep
        else
            delete
    return A

$A \leftarrow LocalMinima(K,\ I,\ T_I)$

function POTENTIALPATHS(K, I, A)
    for each $3K$ in I **do** ▷ *Convolve Image with kernel size* $\times$ *3*
        $Q \leftarrow A_{3k}$ ▷ *Nodes to be Checked*
        for each pair (u, v) from Q **do**
            $P \leftarrow dijkstra(Q_u,\ Q_v)$ OR $astar(Q_u,\ Q_v)$ ▷ *Path from u to v*
            $C \leftarrow P.cost$ ▷ *Cost from u to v*
    return P, C

$P,\ C \leftarrow PotentialPaths(K,\ I,\ A)$

$T_C \leftarrow \mu_C - \sigma_C$ ▷ *Cost Threshold*
function CLEANPATHS(P, C)
    for each p in P **do**
        if $C_p \geq T_C$ **then**
            delete P_p ▷ *Remove Path*
    return P

$P_{clean} \leftarrow CleanPaths(P,\ C)$

$T_S \leftarrow 60\ px$ ▷ *Length Threshold* $1\ px \leftarrow 1\ mm$
function SKELETON(P_{clean}, I)
    $S \leftarrow Skeleton(P_{clean})$ ▷ *Reduce to connected skeleton of 1px thickness*
    if $any(len(S) \leq T_S)$ **then**
        delete ▷ *Remove Isolated Path*
    for each s in S **do** ▷ *the skeleton can result in several smaller parts*
        $b,\ l \leftarrow branchs,\ loops$
        $C_b,\ C_l \leftarrow \mu_{I_b},\ \mu_{I_l}$ ▷ *Cost branch and loop*
        if $C_b \geq T_C$ **then**
            delete s_b ▷ *Remove branch*
        if $C_l \geq T_C$ **then**
            delete s_l ▷ *Remove loop*
    return S

$S \leftarrow Skeleton(P_{clean},\ I)$

$T_w \leftarrow \mu_{I_S} + 0.6\sigma_{I_s}$
function WIDTHGROWTH(S, I)
    while S is unchanged **do**
        if $\mathcal{N}(S) \leq T_w$ **then** ▷ *Neighbors*
            $S_{Final} \leftarrow S \cup \mathcal{N}(S)$
    return S_{Final}

$Cracks \leftarrow WidthGrowth(S,\ I)$

References

1. Walsh, J.B. The effect of cracks on the compressibility of rock. *J. Geophys. Res.* **1965**, *70*, 381–389. [CrossRef]
2. Ostachowicz, W.; Krawczuk, M. Analysis of the effect of cracks on the natural frequencies of a cantilever beam. *J. Sound Vib.* **1991**, *150*, 191–201. [CrossRef]
3. Abraham, F.F.; Brodbeck, D.; Rafey, R.; Rudge, W. Instability dynamics of fracture: A computer simulation investigation. *Phys. Rev. Lett.* **1994**, *73*, 272. [CrossRef] [PubMed]
4. Sharon, E.; Gross, S.P.; Fineberg, J. Local crack branching as a mechanism for instability in dynamic fracture. *Phys. Rev. Lett.* **1995**, *74*, 5096. [CrossRef] [PubMed]
5. De Leon, G.; Fidecaro, F.; Cerchiai, M.; Reggiani, M.; Ascari, E.; Licitra, G. Implementation of CNOSSOS-EU Method for Road Noise in Italy. In Proceedings of the 23rd International Congress on Acoustics, Integrating 4th EAA Euroregio, Aachen, Germany, 9–13 September 2019.
6. Fiorentini, N.; Leandri, P.; Losa, M. Predicting international roughness index by deep neural networks with Levenberg-Marquardt backpropagation learning algorithm. In Proceedings of the Earth Resources and Environmental Remote Sensing/GIS Applications XII, Online, 13–18 September 2021; Volume 11863, pp. 194–207.
7. Oliveira, H.; Correia, P.L. Automatic road crack detection and characterization. *IEEE Trans. Intell. Transp. Syst.* **2012**, *14*, 155–168. [CrossRef]
8. Lapetra, C.; Mayo, J.; Dominguez, J. The randomness of fatigue crack growth under constant-amplitude loads. *Fatigue Fract. Eng. Mater. Struct.* **1996**, *19*, 589–600. [CrossRef]
9. Munawar, H.S.; Hammad, A.W.; Haddad, A.; Soares, C.A.P.; Waller, S.T. Image-based crack detection methods: A review. *Infrastructures* **2021**, *6*, 115. [CrossRef]
10. Mohan, A.; Poobal, S. Crack detection using image processing: A critical review and analysis. *Alex. Eng. J.* **2018**, *57*, 787–798. [CrossRef]
11. Zakeri, H.; Nejad, F.M.; Fahimifar, A. Image based techniques for crack detection, classification and quantification in asphalt pavement: A review. *Arch. Comput. Methods Eng.* **2017**, *24*, 935–977. [CrossRef]
12. Chen, Z.; Hutchinson, T.C. Image-based framework for concrete surface crack monitoring and quantification. *Adv. Civ. Eng.* **2010**, *2010*, 215295. [CrossRef]
13. 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]
14. Abdel-Qader, I.; Abudayyeh, O.; Kelly, M.E. Analysis of edge-detection techniques for crack identification in bridges. *J. Comput. Civ. Eng.* **2003**, *17*, 255–263. [CrossRef]
15. Nhat-Duc, H.; Nguyen, Q.L.; Tran, V.D. Automatic recognition of asphalt pavement cracks using metaheuristic optimized edge detection algorithms and convolution neural network. *Autom. Constr.* **2018**, *94*, 203–213. [CrossRef]
16. Fan, R.; Bocus, M.J.; Zhu, Y.; Jiao, J.; Wang, L.; Ma, F.; Cheng, S.; Liu, M. Road crack detection using deep convolutional neural network and adaptive thresholding. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 474–479.
17. Oliveira, H.; Correia, P.L. Automatic road crack segmentation using entropy and image dynamic thresholding. In Proceedings of the 2009 17th European Signal Processing Conference, Glasgow, UK, 24–28 August 2009; pp. 622–626.
18. Hoang, N.D. Detection of surface crack in building structures using image processing technique with an improved Otsu method for image thresholding. *Adv. Civ. Eng.* **2018**, *2018*, 3924120. [CrossRef]
19. Akagic, A.; Buza, E.; Omanovic, S.; Karabegovic, A. Pavement crack detection using Otsu thresholding for image segmentation. In Proceedings of the 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 21–25 May 2018; pp. 1092–1097.
20. Dorafshan, S.; Maguire, M.; Qi, X. *Automatic Surface Crack Detection in Concrete Structures Using OTSU Thresholding and Morphological Operations*; Civil and Environmental Engineering Faculty Publications: Logan, UT, USA, 2016.
21. Sun, B.C.; Qiu, Y.j. Automatic identification of pavement cracks using mathematic morphology. In Proceedings of the International Conference on Transportation Engineering 2007, Chengdu, China, 22–24 July 2007; pp. 1783–1788.
22. Abas, F.S.; Martinez, K. Classification of painting cracks for content-based analysis. In Proceedings of the Machine Vision Applications in Industrial Inspection XI, Santa Clara, CA, USA, 20–24 January 2003; Volume 5011, pp. 149–160.
23. Xiang, J.; Liang, M. Wavelet-based detection of beam cracks using modal shape and frequency measurements. *Comput.-Aided Civ. Infrastruct. Eng.* **2012**, *27*, 439–454. [CrossRef]
24. Nigam, R.; Singh, S.K. Crack detection in a beam using wavelet transform and photographic measurements. *Structures* **2020**, *25*, 436–447. [CrossRef]
25. Chambon, S.; Subirats, P.; Dumoulin, J. Introduction of a wavelet transform based on 2D matched filter in a Markov random field for fine structure extraction: Application on road crack detection. In Proceedings of the Image Processing: Machine Vision Applications II, San Jose, CA, USA, 18–22 January 2009; Volume 7251, pp. 87–98.
26. Hsieh, Y.A.; Tsai, Y.J. Machine learning for crack detection: Review and model performance comparison. *J. Comput. Civ. Eng.* **2020**, *34*, 04020038. [CrossRef]
27. Yokoyama, S.; Matsumoto, T. Development of an automatic detector of cracks in concrete using machine learning. *Procedia Eng.* **2017**, *171*, 1250–1255. [CrossRef]

28. Kim, H.; Ahn, E.; Shin, M.; Sim, S.H. Crack and noncrack classification from concrete surface images using machine learning. *Struct. Health Monit.* **2019**, *18*, 725–738. [CrossRef]
29. Silva, W.R.L.d.; Lucena, D.S.d. Concrete cracks detection based on deep learning image classification. *Proceedings* **2018**, *2*, 489.
30. Flah, M.; Suleiman, A.R.; Nehdi, M.L. Classification and quantification of cracks in concrete structures using deep learning image-based techniques. *Cem. Concr. Compos.* **2020**, *114*, 103781. [CrossRef]
31. de León, G.; Cesbron, J.; Klein, P.; Leandri, P.; Losa, M. Novel methodology to recover road surface height maps from illuminated scene through convolutional neural networks. *Sensors* **2022**, *22*, 6603. [CrossRef]
32. Zheng, M.; Lei, Z.; Zhang, K. Intelligent detection of building cracks based on deep learning. *Image Vis. Comput.* **2020**, *103*, 103987. [CrossRef]
33. Kim, B.; Cho, S. Automated vision-based detection of cracks on concrete surfaces using a deep learning technique. *Sensors* **2018**, *18*, 3452. [CrossRef] [PubMed]
34. Fiorentini, N.; Pellegrini, D.; Losa, M. Overfitting prevention in accident prediction models: Bayesian regularization of artificial neural networks. *Transp. Res. Rec.* **2023**, *2677*, 1455–1470. [CrossRef]
35. Castelvecchi, D. Can we open the black box of AI? *Nat. News* **2016**, *538*, 20. [CrossRef]
36. Holm, E.A. In defense of the black box. *Science* **2019**, *364*, 26–27. [CrossRef]
37. Rai, A. Explainable AI: From black box to glass box. *J. Acad. Mark. Sci.* **2020**, *48*, 137–141. [CrossRef]
38. Tang, Y.; Huang, Z.; Chen, Z.; Chen, M.; Zhou, H.; Zhang, H.; Sun, J. Novel visual crack width measurement based on backbone double-scale features for improved detection automation. *Eng. Struct.* **2023**, *274*, 115158. [CrossRef]
39. Que, Y.; Dai, Y.; Ji, X.; Leung, A.K.; Chen, Z.; Tang, Y.; Jiang, Z. Automatic classification of asphalt pavement cracks using a novel integrated generative adversarial networks and improved VGG model. *Eng. Struct.* **2023**, *277*, 115406. [CrossRef]
40. Sengupta, E.; Garg, D.; Choudhury, T.; Aggarwal, A. Techniques to elimenate human bias in machine learning. In Proceedings of the 2018 International Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, India, 23–24 November 2018; pp. 226–230.
41. Oliveira, H.; Correia, P.L. CrackIT—An image processing toolbox for crack detection and characterization. In Proceedings of the 2014 IEEE international conference on image processing (ICIP), Paris, France, 27–30 October 2014; pp. 798–802.
42. Amhaz, R.; Chambon, S.; Idier, J.; Baltazart, V. A new minimal path selection algorithm for automatic crack detection on pavement images. In Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014; pp. 788–792.
43. Amhaz, R.; Chambon, S.; Idier, J.; Baltazart, V. Automatic Crack Detection on Two-Dimensional Pavement Images: An Algorithm Based on Minimal Path Selection. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 2718–2729. [CrossRef]
44. Dijkstra, E.W. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*; ACM Books: New York, NY, USA, 2022; pp. 287–290.
45. Yang, L.; Baltazart, V.; Amhaz, R.; Jiang, P. A new A-star algorithm adapted to the semi-automatic detection of cracks within grey level pavement images. In Proceedings of the Eighth International Conference on Digital Image Processing (ICDIP 2016), Chengu, China, 20–22 May 2016; Volume 10033, pp. 796–800.
46. Baltazart, V.; Nicolle, P.; Yang, L. Ongoing tests and improvements of the MPS algorithm for the automatic crack detection within grey level pavement images. In Proceedings of the 2017 25th European Signal Processing Conference (EUSIPCO), Kos, Greece, 28 August–2 September 2017; pp. 2016–2020.
47. Kaddah, W.; Elbouz, M.; Ouerhani, Y.; Baltazart, V.; Desthieux, M.; Alfalou, A. Optimized minimal path selection (OMPS) method for automatic and unsupervised crack segmentation within two-dimensional pavement images. *Vis. Comput.* **2019**, *35*, 1293–1309. [CrossRef]
48. Chen, Y.; Liang, J.; Gu, X.; Zhang, Q.; Deng, H.; Li, S. An improved minimal path selection approach with new strategies for pavement crack segmentation. *Measurement* **2021**, *184*, 109877. [CrossRef]
49. Salman, M.; Mathavan, S.; Kamal, K.; Rahman, M. Pavement crack detection using the Gabor filter. In Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), The Hague, The Netherlands, 6–9 October 2013; pp. 2039–2044.
50. Liu, Y.; Yao, J.; Lu, X.; Xie, R.; Li, L. DeepCrack: A Deep Hierarchical Feature Learning Architecture for Crack Segmentation. *Neurocomputing* **2019**, *338*, 139–153. [CrossRef]
51. Zou, Q.; Zhang, Z.; Li, Q.; Qi, X.; Wang, Q.; Wang, S. Deepcrack: Learning Hierarchical Convolutional Features for Crack Detection. *IEEE Trans. Image Process.* **2019**, *28*, 1498–1512. [CrossRef] [PubMed]
52. Jain, A.K.; Farrokhnia, F. Unsupervised texture segmentation using Gabor filters. *Pattern Recognit.* **1991**, *24*, 1167–1186. [CrossRef]